Point-to-Manifold Geometry: Flexibly Mitigating the Curse of Dimensionality in Neural Computational Units

Rohan Ghosh and Mehul Motani
Department of Electrical and Computer Engineering
National University of Singapore
rghosh92@gmail.com, motani@nus.edu.sg

Abstract

The ability of neural networks to generalize is fundamentally shaped by their computational units. While the perceptron provides a first-order, linear inductive bias and the radial basis function (RBF) unit offers an isotropic curved bias, we identify a critical opportunity for a structured extension: the Generative Matching Unit (GMU). Each individual GMU captures complex dependencies by treating the forward pass as an inference problem; its internal generative model optimizes instance-specific latent parameters to compute a reconstruction error, effectively measuring a point-to-manifold distance as opposed to the point-to-point distance in RBFs. We focus on linear GMUs, which yield closed-form analytical expressions for fast computation and extend naturally to convolutional architectures. Our theoretical analysis demonstrates that linear GMUs are highly flexible universal approximators capable of recovering structured class posteriors. We prove that while a single GMU can exactly emulate an RBF kernel and two can emulate the decision boundary of a perceptron, emulating a single k-order GMU's decision boundary requires a hidden layer of standard units that scales polynomially or exponentially with input dimensionality. Furthermore, we prove that the GMU's point-to-manifold distance can be significantly more discriminative than Euclidean distance when data has high ambient and low intrinsic dimension, offering unique robustness to the curse of dimensionality. Finally, we show that GMUs offer superior flexibility and efficiency in learning smooth manifold decision boundaries compared to MLPs and RBFs. Motivated by the theoretical results, we place GMUs in the first network layer like RBFs, where their role is to enhance linear separability for subsequent layers. This avoids the typical gradient collapse problem with stacking RBF-like units while ensuring the generalization benefits remain. We find that across extensive experiments involving 27 tabular datasets, five vision datasets evaluated across 34 test-time corruption settings, and 30 synthetic data scenarios, GMU networks demonstrate statistically significant improvements in generalization and robustness.

1 Introduction

Computational units serve as the fundamental building blocks of deep learning, defining the inductive bias that directly influences the generalization behavior of a trained model. While universal approximation theorems guarantee sufficient representational capacity for neural architectures, the behavior of individual units dictates how a model adapts to new inputs. Despite the success of the perceptron's first-order linear bias and the isotropic curved bias of Radial Basis Functions (RBFs, Broomhead and Lowe [1988]), a fundamental question remains: what are the inherent probabilistic biases of these units, and do they align with the complexity of real-world data?

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

In recent years, architectural innovation has focused on stacking layers or dynamically directing computation via mechanisms like Self-Attention, Hypernetworks [Ha et al., 2016], or Kolmogorov-Arnold Networks [Liu et al., 2025]. We take a different approach by rethinking the computational unit through a dual *Bayesian* and *Geometric* lens. While the Universal Approximation Theorem ensures that networks can fit any function in the mathematical limit, we wish to further geometrically explore the efficiency and robustness of that representation. Furthermore, we argue that to move beyond mere curve-fitting, a unit must be aligned with the probabilistic structure of the data it intends to model.

Bayes-Optimality. In the supervised classification problem $X \to Y$, by characterizing the distributions $P(X, Y)$ under which a single-layer network achieves *Bayes-optimal* classification, we note that: the perceptron is optimal only when data factorizes into conditionally independent distributions. Most natural data aren't conditionally factorizable, however, and thus in order to use more sophisticated models, we introduce the *Class-wise Latent Structural Causal Model* (CL-SCM). In this generative model, each class y is governed by a distinct generative mechanism, containing its own set of weights W_y. We separately test the alignment of the CL-SCM model with natural data (Appendix 4.4), and find that there is positive alignment that can be directly exploited by using carefully constructed computational units. Subsequently, we ask, "If the perceptron is the Bayes-Optimal unit for conditionally factorizable distributions, what is the Bayes-Optimal Unit for distributions which follow a CL-SCM process?". The answer to this leads us to the foundation of the proposed computational unit, called the Generative Matching Unit (GMU).

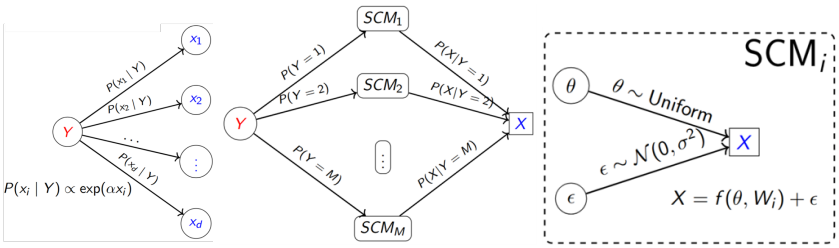
Flexible Geometry. Unlike traditional units that compute direct feature mappings, a GMU treats each forward pass as a backward internal inference problem, optimizing latent parameters that minimize a *point-to-manifold* reconstruction error. It subsequently outputs a non-linear function of that error that is monotonically decreasing w.r.t the error magnitude (i.e., higher error $\to$ lower activation). Readers will note that this philosophy is somewhat aligned with Radial Basis Function (RBF) units, where every unit computes a distance between the input and another point (i.e. its 'center') and the output is a monotonically decreasing function of this distance. Within GMU's framing, RBF's *manifold* can be thought of as zero-dimensional (i.e. a fixed singular point). This represents a philosophical shift in neural computation: moving from point-to-point (RBF) or point-on-plane (Perceptron) geometry to a point-to-manifold geometry. Throughout this work, we demonstrate the benefits of point-to-manifold geometry from multiple perspectives. Geometrically, the decision boundary that emerges from point-to-manifold geometry lies in between the flat hyperplaned perceptron and the isotropically curved spherical boundaries of an RBF: it represents flexible hyper-cylinders. This enables more *efficient* universal approximation overall and better adherence to smooth manifold decision boundaries.

Combining the Bayesian and geometric perspectives, we find GMU offers more flexible approximation while inherently catering to structured distributions in high-dimensional, complex data.

2 Contributions

In this work, we study both the geometric and Bayesian side of computational units, identifying a critical gap between first-order linear inductive biases (Perceptrons) and isotropic curved biases (RBFs). We summarize our contributions below:

- **Bayesian Exploration of Computational Units (Section 3):** We characterize the correspondence between computational units and Bayes Optimality. We prove that while Perceptrons are optimal under conditionally independent sampling, GMUs emerge as the Bayes-optimal unit for structured sampling frameworks (i.e. *Class-wise Latent Structural Causal Models* (CL-SCMs)).
- **The GMU Framework (Section 4):** We introduce the *Generative Matching Unit* (GMU), which treats the forward pass as an internal inference problem to compute a **point-to-manifold** distance. We derive a closed-form *Linear GMU* for analytical efficiency.
- **Geometric Analysis and Universal Approximation (Section 5):** We prove that GMUs are universal approximators that maintain discriminative power in high dimensions ($d \to \infty$) where standard Euclidean distances fail. We further establish that emulating a single k-order GMU requires a hidden layer of standard units that scales **polynomially (RBF)** or **exponentially (MLP)**.
- **General Manifold Approximation Efficiency (Section 6):** We derive scaling laws for approximation of smooth manifold decision boundaries among units, showing GMUs are exponentially more efficient than MLPs/RBFs at approximating anisotropic C^3-smooth hypersurfaces.
- **Extensive Empirical Scale:**



(a) ECI Sampling (b) CL-SCM sampling (c) Proposed SCM Block

Figure 1: Comparing ground truth data generation models (a) Exponential Conditionally Independent (ECI) Sampling, which samples every input dimension x_i independently given the label Y and (b) Class-wise latent structural causal model based sampling (CL-SCM), which maintains a separate SCM for each class. (c) outlines the SCM in the CL-SCM approach, which can emulate more complex structural dependencies in the data.

  - **Tabular & Vision (Section 8):** Statistically significant gains across 27 OpenML datasets and robustness to 34 test-time corruptions on 5 vision datasets.
  - **Controlled Synthetic Studies (Appendix 3):** A suite of 7 controlled experimental scenarios isolating GMU advantages in learning Fourier series, Tree-structured generative models, and Sparse Neural generation, among others.
- **Mechanistic Insights & Ablations (Appendix 4):** Through a linear probe analysis, we demonstrate that a single GMU layer provides superior linear separability. We also ablate the effect of GMU order k and show that stacking GMU layers improves generalization and reduces result deviation. Lastly, we note that a general tradeoff is the increased computational cost of GMUs (Appendix 7.5).

3 Bayes Optimality of Computational Units

In this section, we study when computational units can recover the true conditional probabilities, thereby achieving Bayes optimal classification (i.e. recover true $P(Y \mid X)$). We begin with a simple conditionally independent setting for perceptrons, and then extend to more structured cases, including class-wise latent models that can be viewed as instances of switching causal models [Willig et al., 2025], which motivates our new generative computational unit. Given random variables X and Y denoting inputs and labels, we consider samples $S = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ from $P(X, Y)$. The sampling models are outlined in Figure 1.

Definition 3.1 (Exponential Conditionally Independent (ECI) Sampling)**.** Let $X = [x_1, x_2, \ldots, x_d] \in \mathbb{R}^d$ and class label $Y \in \{1, \ldots, M\}$. The data is sampled such that $P(Y|X) \propto P(Y) \prod_{j=1}^{d} P(x_j|Y)$, where each feature x_j is conditionally independent given Y, and follows the exponential form $P(x_j|Y = i) = \frac{1}{Z_{ij}} \exp(\alpha_{ij} x_j)$, where Z_{ij} is a normalization factor.

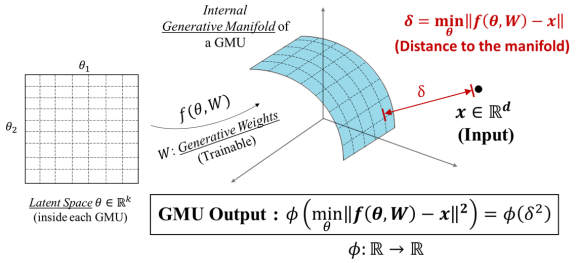
A single-layer perceptron trained with cross-entropy recovers $P(Y \mid X)$ under the ECI assumption.

Proposition 3.2. *Let (X, Y) follow the ECI sampling process, and let $S = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ be i.i.d. samples from this distribution. Consider a single-layer neural network with computational units $\mathrm{pu}_i(X, \mathbf{W}, \mathbf{b}) := \sum_{j=1}^{d} W_{ij} x_j + b_i, \quad i \in \{1, \ldots, M\}$, followed by a softmax layer producing output $P_{\mathrm{soft}}(Y|X, \mathbf{W}, \mathbf{b})$. Let $\mathcal{L}_{\mathrm{CE}}$ denote empirical cross-entropy loss assumed uniformly bounded on the support of $P(Y|X)$. Then, for every $(\mathbf{W}^*, \mathbf{b}^*) \in \arg\min_{\mathbf{W}, \mathbf{b}} \mathcal{L}_{\mathrm{CE}}(S)$, $\lim_{n \to \infty} \mathrm{KL}\left(P(Y|X) \,\|\, P_{\mathrm{soft}}(Y|X, \mathbf{W}^*, \mathbf{b}^*)\right) = 0$ almost surely.*

While the ECI sampling assumes conditional independence between input dimensions, real-world data often has structured patterns. To address this, we introduce a structured sampling process that encodes these inter-dimensional relationships, called Class-wise Latent Structural Causal Model (SCM) sampling.

Definition 3.3 (Class-wise Latent SCM (CL-SCM) Sampling)**.** Let $X = [x_1, x_2, \ldots, x_d] \in \mathbb{R}^d$ be generated by one of M latent structural causal models (SCMs) corresponding to each class $Y \in \{1, \ldots, M\}$:

$$X = f(\theta, W_Y) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2 I), \quad \theta \sim \mathcal{U}(\Omega),$$



(a)

Parameter Choices	
θ	$\mathbb{R}^k$
b	$\mathbb{R}^d$; $[0, 0, .., 0]$
W	$\mathbb{R}^{k \times d}$;
	$[\mathbb{R}^{k-1 \times d}; J_{1,d}]$;
	$[w, w^2, ..w^k], w : \mathbb{R}^d$
f	$W^T\theta$
$\phi(z)$	e^{-z}; $\sqrt{1-z}$; $-\log z$
η	1; σ_{X-b}; $\sqrt{\sum_{i=1}^d x_i^2/d}$
σ^2	d

(b)

Figure 2: (a) A conceptual depiction of a single generative matching unit. A GMU has an internal generative manifold $\mathcal{M} = f(\theta, W)$ parameterized by trainable weights W. Given an input x, it outputs a function (ϕ) of the distance between x and $\mathcal{M}$. We choose ϕ such that larger distances yield lower outputs. (b) Table showing potential parameter choices for the GMU function in (1). Note that we focus on linear manifolds only ($f : W^T\theta$), and that there are many ways to parameterize the internal weights, even beyond those explored here. We set $\sigma^2 = d$ throughout, which helps standardize the exponent.

where $\theta \in \mathbb{R}^k$ represents latent confounders influencing X, W_Y corresponds to the structural parameters of the causal model, and $\Omega \subset \mathbb{R}^k$ is any compact support set for the latent variable θ with finite, non-zero volume. Here, $\mathcal{U}$ denotes the uniform distribution.

The class-wise latent SCM (CL-SCM) can be viewed as a special case of switching SCMs [Willig et al., 2025], where different causal mechanisms are activated depending on context activated by Y. Rather than operating under a single invariant graph, the data-generating process dynamically switches between class-specific regimes. In the CL-SCM case, we show a new computational unit emerges that recovers the true posterior, but only under a MAP approximation.

Proposition 3.4. *Let the random variables (X, Y) follow the CL-SCM sampling process, and let $S = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ be i.i.d. samples from this distribution. Consider a single-layer neural network with computational units $g_i(X, \mathbf{W}, \sigma) := b_i - \frac{1}{\sigma^2} \min_{\theta \in \Omega} \mathbb{E}\left[\|f(\theta, W_i) - X\|^2\right]$, $i \in \{1, \ldots, M\}$, followed by a softmax layer producing output $P_{\text{soft}}(Y \mid X, \mathbf{W}, \sigma)$. b_i are bias terms. Let $\mathcal{L}_{\text{CE}}$ denote the empirical cross-entropy loss and assume it is uniformly bounded over the support of $P(Y \mid X)$. Then, for every $(\mathbf{W}^*, \sigma^*) \in \arg\min_{\mathbf{W}, \sigma} \mathcal{L}_{\text{CE}}(S)$, we have $\lim_{n \to \infty} \text{KL}(P(Y \mid X) \,\|\, P_{\text{soft}}(Y \mid X, \mathbf{W}^*, \sigma^*)) = 0$ almost surely.*

The computational unit g_i explicitly performs an internal optimization for every input instance X, using an internal generative model $f(\theta, W_i)$ to reconstruct X. This directly motivates the need for generative computational units, which we outline in the following sections.

Remark 3.5. The unit g_i may not be directly adaptable to real architectures. When active, its output remains close to zero, whereas when inactive, it can become significantly negative. To yield a better range, the final form of the GMU incorporates an additional activation function ϕ, ensuring that the unit maintains a positive activation when it effectively predicts the input and approaches zero otherwise.

Remark 3.6. The CL-SCM assumption itself, which is intuitive from a simple generative framing of any natural input distribution, still needs verification to support our proposed unit. In Appendix 4.4, we demonstrate whether this assumption aligns with natural datasets, in two different ways. Overall, we find positive alignment of natural data (vision and tabular) with the CL-SCM assumption (assuming linear f), that highlights the potential use of such a computational unit. We suggest that readers refer to this after going through the details of our proposed unit in the next section.

4 Generative Matching Units: Variants and Theory

We define a Generative Matching Unit (GMU) as a function that estimates a function of the distance between an input and an internal generative manifold.

Definition 4.1 (Generative Matching Unit). Let the input be $X = [x_1, \ldots, x_d] \in \mathbb{R}^d$. A GMU of order k is parameterized by generative weights $W \in \mathbb{R}^{k \times d}$, bias $b \in \mathbb{R}^d$, and latent variables $\theta \in \mathbb{R}^k$. Consider $\mathcal{F}$ such that $f(\theta, W) : \mathbb{R}^k \to \mathbb{R}^d$ for $f \in \mathcal{F}$ for input θ and fixed weights W.

A GMU computes

$$\text{gmu}(X) = \phi\left(\frac{1}{\sigma^2}\min_{\theta\in\mathbb{R}^k}\left\|\frac{X-b}{\eta} - f(\theta, W)\right\|^2\right) \tag{1}$$

where $\phi : \mathbb{R} \to \mathbb{R}$ is the activation function, η is an optional normalization factor ensuring boundedness of the distance term, and σ is an optional smoothing parameter.

Conceptually, a GMU computes a function ϕ of the distance from X to its internal manifold (Figure 2); typically, ϕ is chosen so that smaller distances yield higher activations. A special case is the **μ-RBF unit**, where $k = 0, \eta = 1, \sigma^2 = d, f = 0$, and $\phi(z) = e^{-z^2}$, reducing (1) to a dimension-averaged Radial Basis Function.

4.1 Linear GMUs

By restricting f to linear functions ($W^T\theta$), the GMU models a linear subspace of dimensionality k. This formulation allows for a closed-form solution via least-squares:

$$\text{gmu}(X) = \phi\left(\frac{1}{\sigma^2}\left\|\frac{X-b}{\eta} - W^T(W^T)^\dagger\left(\frac{X-b}{\eta}\right)\right\|^2\right) \tag{2}$$

where $(W^T)^\dagger$ is the pseudo-inverse. In our experiments, we set $\sigma^2 = d$ as the $1/d$ term sufficiently stabilizes gradients and leads to superior performance. Technical parameter choices and tensor dimensions for W, b, and η are detailed in Figure 2(b).

5 GMUs and the Curse of Dimensionality

First, we study the universal approximation abilities of two layered GMU-MLPs, which consists of a layer containing multiple GMUs followed by a linear perceptron layer. First, let us define the set of functions $L^p(\mathbb{R}^d)$ such that every $f \in L^p(\mathbb{R}^d)$, where $f : \mathbb{R}^d \to \mathbb{R}$, is p^{th} power integrable, bounded, and continuous with compact support. This leads to the universal approximation theorem for GMUs, as an extension of the result in Park and Sandberg [1991].

Proposition 5.1. *(from Park and Sandberg [1991]) We are given a GMU-MLP, with GMU units in* (1) *of arbtrary k specified as: $\eta = 1$ and $\exists W s.t. f(\theta, W) = C \forall \theta$ (C is any constant) and $\phi(z)$ is any integrable bounded function such that $\int \phi(x)dx \neq 0$. Then, this GMU-MLP can approximate any function $f \in L^p(\mathbb{R}^d)$.*

Note that linear GMUs satisfy the constraints of Proposition 5.1, and thus are universal approximators.

5.1 Comparing Linear GMUs with RBFs

Linear GMUs merit a direct comparison to RBFs in some ways, because both GMUs and RBFs compute distances internally, and return a function of this distance. For RBFs, it is the Euclidean distance $||X_1 - X_2||$, whereas for linear GMUs of order k, it is the distance between a k-dimensional linear subspace and a point. We call this the k-subspace distance, and it is formally defined as follows:

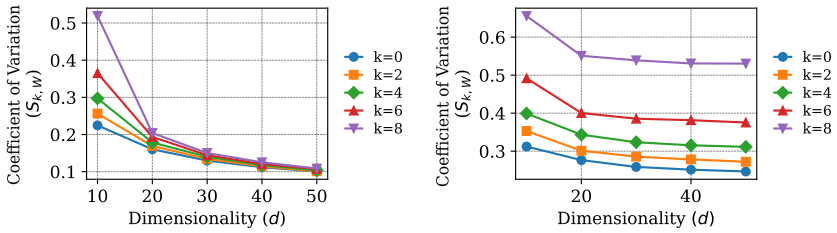
Definition 5.2 (k-Subspace Distance)**.** The k-subspace distance between two points $X_1, X_2 \in \mathbb{R}^d$ with respect to a generative weight matrix $W \in \mathbb{R}^{k\times d}$ is given by:

$$S_{k,W}(X_1, X_2) = \min_{\theta\in\mathbb{R}^k}\left\|X_2 - (W^T\theta + X_1)\right\|.$$

This measures the closest distance between X_2 and the k-dimensional affine subspace generated by W which passes through X_1.

We note that the linear GMU computes a k-subspace distance prior to its activation function. When $k = 0$, this expression reduces to the standard Euclidean distance used in RBFs. Since the activation functions considered in our work are monotonic (Figure 2(b)), the behavior of GMUs is fundamentally governed by this distance metric. In what follows, we analyze the k-subspace distance through the lens of the curse of dimensionality.

Curse of Dimensionality: As dimensionality increases, distance-based metrics tend to lose their discriminative power. In high-dimensional spaces, most points become nearly equidistant from one another, causing similarity measures to collapse [Xia et al., 2015]. The effectiveness of Euclidean distance diminishes as variability in measured distances decreases, making tasks like nearest-neighbor



(a) Full Dimensionality (ID $= d$) (b) Low Intrinsic Dimensionality (ID $= 10$)

Figure 3: Coefficient of variation (CV) versus dimensionality for various distance measures. We compare the k-subspace distance used by GMUs against Euclidean distance ($k = 0$), which is standard for RBF units, under both (a) full-dimension data and (b) low intrinsic dimensionality settings.

classification or kernel methods more challenging. This naturally raises the question: is the k-subspace distance inherently more robust to the curse of dimensionality than Euclidean distance? To address this, we introduce the concept of the *coefficient of Variation* (CV), which quantifies how informative a similarity measure remains in high-dimensional spaces.

Definition 5.3. (**Coefficient of Variation**) We are given a similarity measure $S(X_1, X_2)$, where $X_1, X_2 \in \mathbb{R}^d$. Let I_d represent the identity matrix of size $d \times d$. Then, the coefficient of variation (CV) $\gamma_S(d)$ of S in d-dimensional space is given by:

$$\gamma_S(d) = \frac{\sigma_{X_1,X_2\sim\mathcal{N}(0,I_d)}[S(X_1, X_2)]}{\mu_{X_1,X_2\sim\mathcal{N}(0,I_d)}[S(X_1, X_2)]}, \tag{3}$$

where $\sigma[.]$ and $\mu[.]$ denote the standard deviation and the mean of the random variables within, given $X_1, X_2 \sim \mathcal{N}(0, I_d)$. Maintaining a high CV is desirable, as it implies robustness to increasing dimensionality and prevents the collapse of meaningful differences between points.

Theoretical Results: To investigate whether k-subspace distance is more resistant to the curse of dimensionality, we compare the CV for Euclidean distance (γ_E) and k-subspace distance ($\gamma_{S_{k,W}}$) theoretically. We have the following results.

Theorem 5.4. *Let $E(X_1, X_2) = \|X_1 - X_2\|$, then, we can show that $\gamma_E(d + 1) < \gamma_E(d)$.*

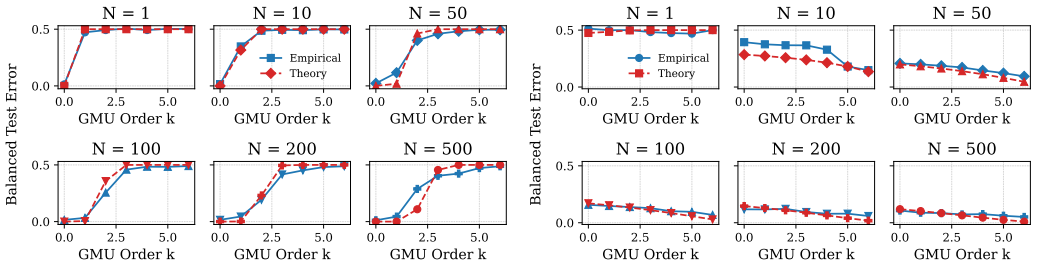
Corollary 5.5. *We are given the k-subspace distance $S_{k,W}(X_1, X_2)$ from X_1 to X_2. With this, first, we note that $\gamma_{S_{0,W}}(d) = \gamma_E(d)$, where E denotes the Euclidean distance. Then, we have that $\gamma_{S_{k+1,W}}(d) > \gamma_{S_{k,W}}(d)$, and thus $\gamma_{S_{k,W}}(d) > \gamma_E(d)$.*

Proposition 5.6. *We consider the case where the RV X has low intrinsic dimensionality. Let $X \in \mathbb{R}^d$ be generated as: $X = \sum_{i=1}^{ID} a_i \boldsymbol{v_i}, \quad a_i \sim \mathcal{N}(0, 1)$, where $\{\boldsymbol{v_i}\}_{i=1}^{ID}$ are orthonormal vectors in $\mathbb{R}^d$, and ID $< d$ is the intrinsic dimensionality of X. Let us set $W = [\boldsymbol{v_1}, \ldots, \boldsymbol{v_k}]$ to be the first k orthonormal vectors. Then, we have that $\gamma_E(d) = \gamma_E(ID)$ and $\gamma_{S_{k,W}}(d) = \gamma_E(ID - k)$.*

Remark 5.7. Proposition 5.4 reinforces the curse of dimensionality, showing that Euclidean distance loses information as dimensionality increases. In contrast, Proposition 5.5 demonstrates that k-subspace distance retains a strictly greater CV, enabling more robust differentiation in high-dimensional spaces. Moreover, higher-order GMUs extract richer structural information as k increases. Lastly, Proposition 5.6 shows that when high-dimensional data has low intrinsic dimension, which is common in real-world datasets, the k-subspace distance retains a high CV.

Empirical verification: We conduct a series of experiments to estimate the CV of both Euclidean distance and k-subspace distance, verifying the above theoretical results. First, we generate $X_1, X_2 \sim \mathcal{N}(0, I_d)$ and compute the CVs $\gamma_E(d)$ and $\gamma_{S_{k,W}}(d)$ as functions of dimensionality d. The results, summarized in Figure 3(a), confirm that k-subspace distance consistently exhibits higher CV compared to Euclidean distance ($k = 0$), highlighting that GMUs can capture greater structural variability in high-dimensional settings.

Next, we test data distributions with low intrinsic dimensionality (ID), common in real datasets. We hypothesize that when data lies in a low-dimensional subspace of $\mathbb{R}^d$, suitable weights W can enhance the CV of k-subspace distance. We generate inputs constrained to 10-dimensional linear subspaces within ambient spaces where $10 \leq d \leq 50$, setting the first k rows of W to match the



(a) RBF Network $\rightarrow$ GMU ($d = 50, R = 0.25$). **(b)** MLP Network $\rightarrow$ GMU ($d = 10, R = 0.25$).

Figure 4: Empirical and theoretical balanced error for RBF and MLP networks emulating a single GMU.

subspace bases. Figure 3(b) shows substantial CV gains for larger k, which persist even as ambient dimensionality d grows high.

5.2 Representational Asymmetry: GMUs vs. RBFs vs. MLPs

While GMUs, RBFs, and MLPs are all universal approximators, they exhibit a striking *representational asymmetry* in their ability to emulate one another. In particular, GMUs can efficiently represent the computations performed by RBFs and MLPs, but the reverse direction is highly inefficient. We show that a single GMU can exactly emulate an RBF unit, whereas both RBF networks and MLPs require rapidly growing numbers of hidden units to approximate even a *single* GMU. In what follows, we consider GMUs with no normalization $\eta = 1$ and non-zero bias b. Here, we focus on the ability of these units to emulate decision boundaries of each other.

GMU $\rightarrow$ RBF: $O(1)$ scaling. We begin by noting the trivial fact that a single GMU unit can exactly emulate an RBF kernel, by setting the GMU weights $W = 0$.

RBF $\rightarrow$ GMU: Polynomial scaling. We next consider the reverse direction: approximating a k-order GMU using a single hidden layer RBF network. The decision boundary of a linear GMU with radius R is a k-aligned hypercylinder in $\mathbb{R}^d$. The following theorem characterizes the number of RBF units required to approximate this boundary.

Theorem 5.8. *Consider an input $X \sim \mathcal{U}[-0.5, 0.5]^d$. We consider a k-order GMU be defined with zero bias, no normalization, and weight axes aligned with k of the d dimensions. The balanced classification error ϵ of an RBF network with N hidden units attempting to emulate the decision boundary of this GMU emulated by the k-subspace contour of distance R:*

$$\epsilon \lessapprox \frac{1}{2}\left[1 - \left(1 - \frac{1}{R^2(NV_k(1))^{2/k}}\right)^{\frac{d-k}{2}}\right] \tag{4}$$

where $V_k(1)$ is the volume of a unit k-sphere. To maintain a constant error $\epsilon < 0.5$ as dimensionality d increases, the number of RBF units N must scale as $N = O(d^{k/2})$.

Proposition 5.9 (GMU $\rightarrow$ MLP.)**.** *Two GMUs (of any order k) in a GMU–linear network can exactly emulate the decision boundary of a single perceptron.*

MLP $\rightarrow$ GMU: Exponential scaling. We now analyze the ability of a single-hidden-layer MLP to emulate a GMU. Unlike RBFs, whose difficulty arises from covering a k-dimensional cross-section, MLPs struggle because they approximate the GMU's curved $(d-k)$-dimensional boundary using flat hyperplane facets. The following theorem quantifies this limitation.

Theorem 5.10. *Consider an input $X \sim \mathcal{U}[-0.5, 0.5]^d$. We consider a k-order GMU defined with zero bias, no normalization, and weight axes aligned with k of the d dimensions. Let $m = d - k$. The balanced classification error ϵ of a single-hidden-layer MLP with N hidden units attempting to emulate the decision boundary of this GMU is approximately lower bounded by:*

$$\epsilon \gtrapprox \left[\frac{m(m-1)}{(m+1)^2}\right]\left(\frac{m-1}{m+1}\right)^{\frac{m-1}{2}}\left(\frac{mV_m}{V_{m-1}}\right)^{\frac{2}{m-1}} N^{-\frac{2}{m-1}} \tag{5}$$

where $V_m = \frac{\pi^{m/2}}{\Gamma(m/2+1)}$ is the volume of the m-dimensional unit ball. To maintain a low error $\epsilon \ll 0.5$ as dimensionality d increases, the number of MLP units must scale exponentially with d, specifically $N = \Theta\Big(\big(\frac{1}{e\,\epsilon}\big)^{\frac{m-1}{2}}\sqrt{m}\Big)$.

Empirical Validation. Figure 4 compares the theoretical predictions of Theorems 5.8 and 5.10 with empirical results for both RBF and MLP networks attempting to emulate a single GMU. For RBFs (Fig. 4a, $d = 50$), the observed error closely follows the polynomial scaling law: performance deteriorates rapidly with increasing GMU order k, and even $N = 500$ units fail to approximate GMUs of order $k \geq 3$. For MLPs (Fig. 4b, $d = 10$), the degradation is even more severe: the empirical error is significantly higher which is why a smaller d is chosen for analysis, and even in this setting the network requires hundreds of units to approximate GMUs. Interestingly though, the trend for MLP is reversed: larger order k is easier to fit, and the curves very closely follow the predicted theoretical trends. Together, these results confirm a strong representational asymmetry: GMUs can compactly express structures that require polynomially many RBF units or exponentially many MLP units to approximate.

6 Representational Efficiency for Smooth Manifold Boundaries

While universal approximation theorems guarantee asymptotic convergence, we explore the *efficiency* of the approximation. To evaluate how different architectures scale, we analyze their capacity to approximate a smooth, $(d-1)$-dimensional decision boundary $\mathcal{S}$ via local ϵ-covering. Specifically, each hidden unit defines a local decision boundary $\mathcal{B}_i$ tangent to the manifold $\mathcal{S}$ at a point $p_i \in \mathcal{S}$, covering a local patch $\mathcal{P}_i \subset \mathcal{S}$ of radius $h(p_i)$. The patch radius is defined as the maximum extent over which the Hausdorff distance between the unit boundary and the manifold satisfies $d_H(\mathcal{S} \cap \mathcal{P}_i, \mathcal{B}_i \cap \mathcal{P}_i) \leq \epsilon$. The network complexity $N(\epsilon, \mathcal{F})$ is then given by the cardinality of the minimal ϵ-cover required to tile $\mathcal{S}$ (i.e., $\mathcal{S} \subseteq \bigcup_{i=1}^{N} \mathcal{P}_i$).

Theorem 6.1. *Let $\mathcal{S} \subset \mathbb{R}^d$ be a compact, C^3-smooth, $(d-1)$-dimensional Riemannian submanifold with finite $(d-1)$-dimensional volume $\mathcal{V}_\mathcal{S}$. We assume the manifold is locally well-approximated by its second-order Taylor expansion. Let v_{d-1} denote the volume of the $(d-1)$-dimensional unit ball. To achieve a global Hausdorff error ϵ, let $N(\epsilon, \mathcal{F})$ denote the expected minimum number of units required to form an ϵ-cover of $\mathcal{S}$ under a uniform data distribution $p \sim U(\mathcal{S})$. Let $\{\kappa_i(p)\}_{i=1}^{d-1}$ be the ordered principal curvatures at $p \in \mathcal{S}$ such that $|\kappa_1| \geq |\kappa_2| \geq \cdots \geq |\kappa_{d-1}|$.*

The expected unit complexity for different computational architectures is:

$$N(\epsilon, \mathcal{F}) \approx \frac{\mathcal{V}_\mathcal{S}}{v_{d-1}(2\epsilon)^{\frac{d-1}{2}}} \mathbb{E}_{p\sim\mathcal{S}}\left[\Delta\kappa_\mathcal{F}(p)^{\frac{d-1}{2}}\right] \tag{6}$$

where the residual curvature $\Delta\kappa_\mathcal{F}$ is defined as:
MLP: $\Delta\kappa_{MLP} = |\kappa_1(p)|$, ***RBF:*** $\Delta\kappa_{RBF} = (|\kappa_1(p) - \kappa_{d-1}(p)|)/2$ *and*
k-GMU: $\Delta\kappa_{GMU} = \min\Big(\min_{k'\in\{1,\ldots,k\}} \max\Big(|\kappa_{d-k'}(p)|, \frac{|\kappa_1(p)-\kappa_{d-k'-1}(p)|}{2}\Big), \frac{|\kappa_1(p)-\kappa_{d-1}(p)|}{2}\Big)$.

The result in Theorem 6.1 reveals that the ambient dimension d is involved in the exponent, while the architecture's geometric fit that determines the residual curvature $\Delta\kappa_\mathcal{F}$ determines the base of that exponent.

Corollary 6.2 (Exponential Efficiency Gap)**.** *Assume the manifold $\mathcal{S}$ is strictly anisotropic such that $|\kappa_1(p)| > |\kappa_2(p)| > \cdots > |\kappa_{d-1}(p)| > 0$ for all $p \in \mathcal{S}$. Let $n = \frac{d-1}{2}$ be the complexity exponent.*

For any two function spaces $\mathcal{F}_1, \mathcal{F}_2$, the relative unit efficiency scales as: $\frac{N(\epsilon,\mathcal{F}_1)}{N(\epsilon,\mathcal{F}_2)} \approx \left[\frac{\|\Delta\kappa_1\|_{L_n(\mathcal{S})}}{\|\Delta\kappa_2\|_{L_n(\mathcal{S})}}\right]^n$,
where $\|f\|_{L_n(\mathcal{S})} = (\mathbb{E}_{p\sim\mathcal{S}}[|f(p)|^n])^{1/n}$.

In particular, under strict anisotropy, there exist geometric constants $\gamma_1, \gamma_2, \gamma_3 > 1$ such that:

$$\frac{N(\epsilon, \mathcal{F}_{MLP})}{N(\epsilon, \mathcal{F}_{GMU})} = \Omega(\gamma_1^n), \quad \frac{N(\epsilon, \mathcal{F}_{RBF})}{N(\epsilon, \mathcal{F}_{GMU})} = \Omega(\gamma_2^n), \quad \frac{N(\epsilon, \mathcal{F}_{MLP})}{N(\epsilon, \mathcal{F}_{RBF})} = \Omega(\gamma_3^n). \tag{7}$$

Remark 6.3. GMUs are more efficient here because their residual curvature is lower than that of RBFs and significantly lower than MLPs. Once raised to $(d-1)/2$, these differences grow rapidly in high dimensions, so GMUs can require far fewer units to reach comparable accuracy.

Remark 6.4. This explains our focus on first-layer GMUs: unlike classical RBFs that compute point-to-point distances using spherically symmetric receptive fields, GMUs compute point-to-manifold distances using learned subspaces. While both share an $\exp(-\text{distance}^2)$ activation core, just like RBFs, sequentially stacking GMUs creates nested exponential compositions (exponentials over exponentials) that lead to gradient collapse and training instability [Wurzberger and Schwenker, 2024]. Our preliminary experiment (Appendix 4.2) bypasses this limitation by using a negation activation ($\phi(x) = -x$) in the second layer; however, developing robust training schemes for deeply stacked GMU networks remains an important avenue for future work.

7 Summary of Related Work

Detailed discussions on other work on alternative architectures and computational units proposed over the years are provided in Appendix 5. Overall, we note that while there exist many alternative computational strategies explored in recent years, they are mainly exploring very different aspects of computation than GMUs, and even supporting GMU-enhanced variants in theory (e.g. Capsule Networks Sabour et al. [2017], Hypernetworks Ha et al. [2016] or even multi-headed attention Nielsen et al. [2024]). In the case of Kolmogorov Arnold Networks [Liu et al., 2025], we find that while they can emulate a GMU layer exactly in theory, there are many strict restrictions on activation choices, custom connectivity, weight structures with a $\mathcal{O}(md^2)$ parameter blow-up that make it substantially impractical in practice. All of this and more is explored in significant detail in Appendix 5.

8 Experiments

We conduct extensive experiments on both synthetic and real datasets. The synthetic experiments are designed to isolate and test specific structural properties of GMUs, while the real-data experiments demonstrate their utility in practical supervised learning tasks. Note that due to limited space, we have shifted all of the synthetic experiments to Appendix 3. Proofs of theoretical results (Appendix 8), extended experiments (Appendices 2 and 3), in-depth analysis of GMUs (Appendix 4), and experiment details (Appendix 7) are provided in the Appendices. Code is provided here.

8.1 Tabular Datasets (Real)

Outline: We test and compare performance on 27 OpenML datasets, selecting a diverse subset from Kadra et al. [2021], ranging from small-scale (800 samples, 5 dimensions) to medium-scale (170k samples, 1k dimensions). For the Resnet-512-[1,1] architecture, which performed well in our synthetic experiments overall, we replace the first layer with four types of GMU units: $k = 0, 1, 2, 3$. We use the polynomial expanded form of the weights to construct the higher-order GMUs ($W = [w, w^2, ..w^k]$ in Table 2(b)). Each configuration maintains 128 units in the first layer, yielding 512 output units. This is denoted as the GMU-Resnet-P configuration. We test one more configuration (GMU-Resnet-S) in Appendix 7.2. Since GMU-Resnets are slightly more parameterized than Resnets, we scaled the Resnets (530 units per layer) to ensure equal parameter counts. Results are provided in Table 1.

Takeaways: We find that overall, in 21 out of 27 cases, GMU-Resnet-P showcases better or on-par balanced accuracy. Furthermore, GMU-Resnet-P performs competitively or favorably against other state-of-the-art approaches in Kadra et al. [2021] when compared via the Wins/Losses/Ties criterion (please see Table 20). Ties are set via balanced accuracy levels being within 0.2% of each other.

8.2 Vision Datasets (Real)

Outline: We construct convolutional GMUs (please see Appendix 6.1), where we replace every linear operation in convolution with a GMU, using which we create GMU-CNN architectures. Like before, only the first layer is replaced with convolutional GMUs. We focus on two aspects: whether the GMUs show better generalization compared to CNNs with the same architecture and whether they show better out-of-distribution performance to test-time corrupions. We report our findings across four datasets: MNIST, Fashion-MNIST, CIFAR-10, smallNORB and their corrupted versions. Results are summarized in Table 2. Additionally, to also showcase that GMUs can be combined with benchmark approaches for out-of-distribution settings such as test time adaptation [You et al., 2021, Lim et al., 2023], we provide additional experiments in Appendix 2. In total, GMU-CNNs and CNNs of similar parametric complexity (see Table 7) are evaluated on 38 dataset–corruption configurations, which expands to 78 when test-time adaptation is included.

Takeaways: We find that GMU-CNNs show substantial improvements in terms of robustness to test-time corruptions in all cases. In all cases, we see that GMU-CNNs show improvements

Models	Datasets													
	credit-g	anneal	kr-vs-kp	mfeat	vehicle	kc1	phoneme	cnae-9	blood	Australian	car	segment	jasmine	sylvine
Resnet	0.6972	0.8546	0.9969	**0.9850**	0.8446	0.7104	**0.8959**	0.9398	0.6304	0.8479	1.0000	**0.9254**	0.7438	**0.9250**
GMU-P	**0.7071**	**0.8596**	0.9954	0.9775	**0.8666**	0.7090	0.8930	**0.9537**	**0.6813**	**0.8743**	1.0000	0.9221	**0.7604**	0.9229
	adult	nomao	bank	jungle	volkert	helena	connect-4	higgs	numerai	walking	ldpa	aloi	skin-seg	
Resnet	0.7652	0.9582	0.7261	0.9686	**0.6766**	0.2213	0.7369	0.6684	0.5066	0.6203	**0.6991**	0.9617	0.9997	
GMU-P	**0.7812**	**0.9604**	**0.7370**	**0.9868**	0.6667	0.2211	**0.7495**	**0.6853**	0.5075	**0.6297**	0.6816	**0.9646**	0.9997	

Table 1: Balanced Accuracy on 27 Tabular datasets from OpenML.

Dataset: MNIST												
Network	Standard	brightness	canny	dotted	fog	glass	impulse	motion	shot	spatter	zigzag	Average
CNN	0.9949	0.2274	0.6149	0.9791	0.1188	0.541	0.4529	**0.9675**	0.9226	**0.9834**	0.7826	0.6895
GMU-CNN	0.9954	**0.9913**	**0.8998**	**0.9896**	**0.9234**	**0.8141**	**0.9377**	0.9614	**0.9449**	0.9759	**0.9447**	**0.9434**
Dataset: Fashion-MNIST												
CNN	0.9329	0.4535	0.3709	0.8786	0.2712	**0.6518**	0.2056	**0.7188**	**0.5959**	0.8835	0.8131	0.615982
GMU-CNN	0.9356	**0.8250**	**0.7058**	**0.9118**	**0.7442**	0.5817	**0.6703**	0.6831	0.48964	**0.8868**	**0.8806**	**0.755867**

(a)

Dataset: smallNORB		
Model	Test Accuracy (Uncorrupted)	Test Accuracy (Corrupted)
CNN	0.954321	0.436872
GMU-CNN (1S-1D)	**0.97477**	0.546502
GMU-CNN (2D-1S)	0.96642	0.516132
GMU-CNN (3S-1D)	0.96	**0.56329**

(b)

Dataset: CIFAR-10															
Network	Standard	brightn	contrast	defocus	elastic	fog	gauss_blur	glass	impulse	motion	pixelate	saturate	shot_noise	spatter	Average
VGG-16	0.8594	0.83642	0.5952	0.68148	0.71436	0.74422	0.5929	0.44146	0.59338	0.6211	0.6826	0.8147	0.6366	0.7440	0.669109
GMU(3)-VGG	**0.8645**	0.8540	0.7489	0.7811	0.7493	0.8176	0.741	**0.4665**	**0.6003**	0.7142	0.7192	0.8211	**0.6372**	0.7861	0.725885
GMU(8)-VGG	**0.8734**	**0.86188**	**0.75478**	**0.7977**	**0.7534**	**0.8311**	**0.7620**	0.4383	0.5608	**0.7325**	**0.7273**	**0.8298**	0.6163	**0.7874**	**0.727174**

(c)

Table 2: Test Accuracy on standard and corrupted data on (a) MNIST and Fashion-MNIST (b) smallNORB, and (c) CIFAR-10 datasets.

when compared with their native CNN baselines. On smallNORB, GMU-CNNs reach performance competitive to other benchmarks such as first generation capsule networks (2.5 vs 2.7% error rate in Patrick et al. [2022]). We don't use data-augmentation or any other regularization approach in these experiments, to focus on the intrinsic generalization performance.

Computation times: We find that GMU-CNNs and GMU-MLPs still maintain instantaneous inference times per sample, in spite of inverse operations. This is because the size of the matrix being inverted in (2) is $k \times k$, where k is the order of the GMU. In all our experiments $k \leq 8$, which makes this operation efficient. Overall, in the tabular data experiments, the average per-sample inference time of GMU-Resnets was 0.017 ms compared to 0.004 ms for Resnets, and for the vision experiments, GMU-CNNs took 0.0265 ms compared to 0.0083 ms for CNNs.

8.3 Cross-Dataset Testing (Real)

In this section, we train GMU-ResNets and ResNets (with the ResNet-18 backbone) on the Street View House Numbers dataset (SVHN, Netzer et al. [2011]) and then test the trained model directly on MNIST. Similarly, we use trained GMU-CNNs and CNNs on MNIST and test on SVHN. The results are shown in Table 3. We not only see GMU-Resnet-18 significantly improve over Resnet-18 on native SVHN, but we also see significant gains in accuracy when directly transferring between datasets.

Table 3: Cross-Dataset Testing

Test Setting	GMU	Standard
SVHN Only	**96.58**	95.60
SVHN $\rightarrow$ MNIST	**83.99**	67.28
MNIST $\rightarrow$ SVHN	**54.50**	19.95

9 Conclusion and Limitations

By rethinking the forward pass as an internal inference problem, we transition neural computation from rigid point-to-point mappings toward a dynamic point-to-manifold geometry. Our theoretical framework establishes that the GMU is the Bayes-optimal unit for Class-wise Latent Structural Causal Models, formally aligning the unit's internal logic with the generative nature of structured data. We further prove that emulating the decision boundary of a single k-order GMU requires an exponential number of MLP units or a polynomial number of RBF units, representing a fundamental efficiency gap in manifold approximation that grows with dimensionality. Beyond these geometric and Bayesian proofs, our extensive empirical evaluation across tabular and vision benchmarks confirms that replacing standard units with GMUs significantly enhances linear separability and robustness to test-time corruptions. By providing a closed-form, analytically efficient implementation, the linear GMU offers a practical path toward architectures using computational units with internal inference.

Limitations: Most of our experiments place GMUs only in the first layer, where their geometry most naturally parallels radial basis functions and yields immediate benefits. Extending GMUs deeper is conceptually possible and we include a two-layer example, but stacking many layers introduces challenges: exponential activations can cause gradient collapse, and the per-unit FLOPs grow with $2k^2d$, raising compute costs at for large-scale data. Thus, while first-layer GMUs already provide substantial gains, exploring deeper integration remains an important direction for future work.

Acknowledgments and Disclosure of Funding

This research is supported by A*STAR, CISCO Systems (USA) Pte. Ltd and the National University of Singapore under its Cisco-NUS Accelerated Digital Economy Corporate Laboratory (Award I21001E0002), and the Ministry of Education, Singapore, under its Academic Research Fund Tier 2 (Award No.: MOE-T2EP20125-0013). We also thank the Kent-Ridge AI research group at the National University of Singapore for helpful discussions. The authors declare no competing financial interests or third-party funding relationships outside the supported work over the past 36 months.

References

David S Broomhead and David Lowe. Multivariable functional interpolation and adaptive networks. *Complex systems*, 2(3), 1988.

Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11027–11036, 2020.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *International conference on machine learning*, pages 1437–1446. PMLR, 2018.

David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. Matrix capsules with em routing. In *International conference on learning representations*, 2018.

Arlind Kadra, Marius Lindauer, Frank Hutter, and Josif Grabocka. Well-tuned simple nets excel on tabular datasets. *Advances in neural information processing systems*, 34:23928–23941, 2021.

Andrei Nikolaevich Kolmogorov. On the representations of continuous functions of many variables by superposition of continuous functions of one variable and addition. In *Dokl. Akad. Nauk USSR*, volume 114, pages 953–956, 1957.

Patrick Mensah Kwabena, Benjamin Asubam Weyori, and Ayidzoe Abra Mighty. Exploring the performance of lbp-capsule networks with k-means routing on complex images. *Journal of King Saud University-Computer and Information Sciences*, 34(6):2574–2588, 2022.

Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Ng. Efficient sparse coding algorithms. *Advances in neural information processing systems*, 19, 2006.

Hyesu Lim, Byeonggeun Kim, Jaegul Choo, and Sungha Choi. TTN: A domain-shift aware batch normalization in test-time adaptation. In *The Eleventh International Conference on Learning Representations*, 2023.

Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljacic, Thomas Y. Hou, and Max Tegmark. KAN: Kolmogorov–arnold networks. In *The Thirteenth International Conference on Learning Representations*, 2025.

Alfredo Nazabal, Nikolaos Tsagkas, and Christopher KI Williams. Inference and learning for generative capsule models. *Neural Computation*, 35(4):727–761, 2023.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 4. Granada, 2011.

Stefan Nielsen, Laziz Abdullaev, Rachel Teo, and Tan Nguyen. Elliptical attention. *Advances in Neural Information Processing Systems*, 37:109748–109789, 2024.

Jooyoung Park and Irwin W Sandberg. Universal approximation using radial-basis-function networks. *Neural computation*, 3(2):246–257, 1991.

Mensah Kwabena Patrick, Adebayo Felix Adekoya, Ayidzoe Abra Mighty, and Baagyire Y Edward. Capsule networks–a survey. *Journal of King Saud University-computer and information sciences*, 34(1):1295–1310, 2022.

Hao Ren and Hong Lu. Compositional coding capsule network with k-means routing for text classification. *Pattern Recognition Letters*, 160:1–8, 2022. ISSN 0167-8655.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. *Advances in neural information processing systems*, 30, 2017.

Jing-Feng Tian and Zhenhang Yang. Asymptotic expansions of gurland's ratio and sharp bounds for their remainders. *Journal of Mathematical Analysis and Applications*, 493(2):124545, 2021.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Moritz Willig, Tim Tobiasch, Florian Busch, Jonas Seng, Devendra Singh Dhami, and Kristian Kersting. Systems with switching causal relations: A meta-causal perspective. In *The Thirteenth International Conference on Learning Representations*, 2025.

Fabian Wurzberger and Friedhelm Schwenker. Learning in deep radial basis function networks. *Entropy*, 26(5), 2024.

Shuyin Xia, Zhongyang Xiong, Yueguo Luo, and Guanghua Zhang. Effectiveness of the euclidean distance in high dimensional spaces. *Optik*, 126(24):5614–5619, 2015.

Fuming You, Jingjing Li, and Zhou Zhao. Test-time batch statistics calibration for covariate shift. *arXiv preprint arXiv:2110.04065*, 2021.

Appendix

Table of Contents

1. **Summary of Appendices** 13
2. **Vision: Comparing with Test-Time Adaptation** 14
3. **Experiments on Synthetic Data** 15
    - **Sparse Linear Structure Prediction** 15
    - **Fourier Series v/s Noise** 15
    - **k-subspace distance thresholding** 17
    - **Sparse Neural Structure Sampling** 18
    - **Tree Structured Sampling** 19
    - **Polynomial Boundary Sampling** 21
    - **Conditional Gaussian Sampling** 23
4. **Ablations and Further Insights** 24
    - **Isolating Effects of Patch Normalization** 24
    - **Stacked GMU Architectures** 24
    - **Analysis of GMU Order k** 25
    - **Verifying the CL-SCM Assumption** 26
    - **Mechanistic Explanation of GMU Robustness** 28
    - **Activation Function Ablations** 29
5. **Discussions: Contextualizing GMUs in Literature** 30
6. **Details on GMU Function and Integration** 33
7. **Additional Empirical Details** 35
8. **Proofs of Theoretical Results** 38

1 Summary of Appendices

In this appendix we share additional experiments and analyses, empirical details and theoretical proofs and any further clarifications. Experiments cover multiple synthetic studies, mechanistic ablations and insights on GMU behaviour and additional test-time adaptation based vision experiments. We also provide detailed discussions of related works.

2 Vision: Comparing with Test-Time Adaptation

Corruption Type	Standard (CNN)	TTA (CNN)	Standard (GMU-CNN)	TTA (GMU-CNN)
Brightness	0.2274	0.9887	0.9913	**0.9933**
Canny Edges	0.6149	0.9431	0.8998	**0.9813**
Dotted Line	0.9791	0.9873	0.9896	**0.9902**
Fog	0.1188	0.6074	0.9234	**0.9609**
Impulse Noise	0.4529	0.9098	0.9377	**0.9683**
Motion Blur	0.9675	**0.9858**	0.9614	**0.9666**
Shot Noise	0.9226	0.9677	0.9449	0.9757
Spatter	0.9834	**0.9857**	0.9759	0.9755
Zigzag	0.7826	0.8707	0.9447	**0.9608**
Average	0.67213333	0.916244	0.952078	**0.974733**

Table 4: Accuracy comparison of standard (no TTA) and Test-Time Adaptation (TTA) approaches across various corruption types on the MNIST dataset.

Corruption Type	Standard (CNN)	TTA (CNN)	Standard (GMU-CNN)	TTA (GMU-CNN)
Brightness	0.4535	0.8281	0.8250	**0.8550**
Canny Edges	0.3709	0.5322	0.7058	**0.7575**
Dotted Line	0.8786	0.8953	0.9118	**0.9161**
Fog	0.2712	0.6060	0.7442	**0.8373**
Impulse Noise	0.2056	0.6614	0.6703	**0.7837**
Motion Blur	0.7188	**0.7865**	0.6831	0.7830
Shot Noise	0.5959	**0.7059**	0.4896	0.6870
Spatter	0.8835	0.8902	0.8868	**0.8908**
Zigzag	0.8131	0.8606	0.8806	**0.8941**
Average	0.6159	0.7518	0.7558	**0.8227**

Table 5: Accuracy comparison of standard (no TTA) and Test-Time Adaptation (TTA) approaches across various corruption types on the Fashion-MNIST dataset.

Corruption Type	Standard (VGG)	TTA (VGG)	Standard (GMU(3))	TTA (GMU(3))	Standard (GMU(8))	TTA (GMU(8))
Brightness	0.83642	0.82376	0.8540	0.85266	0.86188	**0.86362**
Contrast	0.5952	0.74014	0.7489	0.8016	0.75478	**0.81086**
Defocus Blur	0.68148	0.79334	0.7811	0.82878	0.7977	**0.83792**
Elastic Transform	0.71436	0.74232	0.7493	0.7808	0.7534	**0.78318**
Fog	0.74422	0.77798	0.8176	0.8283	0.8311	**0.8393**
Frost	0.7386	0.7386	0.7410	0.77508	0.7620	**0.78332**
Gaussian Blur	0.5929	0.77528	0.7410	0.81844	0.7620	**0.82662**
Impulse Noise	0.59338	0.65226	0.6003	**0.75382**	0.5608	0.73398
Motion Blur	0.6211	0.74056	0.7142	0.79238	0.7325	**0.79896**
Pixelate	0.6826	0.7591	0.7192	0.80666	0.7273	**0.81328**
Saturate	0.8147	0.81246	0.8211	0.8283	0.8298	**0.83882**
Shot Noise	0.6366	0.72318	0.6372	**0.77136**	0.6163	0.76632
Spatter	0.7440	0.73732	0.7861	0.79432	0.7874	**0.7989**
Average	0.691966	0.7551	0.747	0.8025	0.752074	**0.807314**

Table 6: Accuracy comparison of standard (no TTA) and Test-Time Adaptation (TTA) approaches across various corruption types on the CIFAR-10 dataset.

We extend our analysis to include test-time adaptation (TTA) and summarize the statistical comparisons in Table 7. The table reports wins, losses, and ties across corruption types, along with Wilcoxon signed-rank p-values for GMU-CNN versus standard CNN (or VGG in the case of CIFAR-10), both under standard evaluation and with TTA. Several clear trends emerge. First, under standard evaluation, GMU-CNN significantly outperforms the baseline on MNIST and CIFAR-10, and shows a positive but not statistically significant trend on Fashion-MNIST. Second, when TTA is applied, both standard and GMU-based models benefit substantially, confirming that adaptive batch normalization improves robustness to distribution shifts. Crucially, GMU-based architectures continue to hold an advantage even after TTA is applied to both models: on Fashion-MNIST and CIFAR-10, the improvements are statistically significant ($p = 0.01953$ and $p = 0.00012$, respectively), while on MNIST the trend remains favorable though not significant ($p = 0.06445$). Finally, higher-order GMU variants (e.g., GMU(8) on CIFAR-10) show that the benefits of TTA and GMU are complementary, leading to consistent wins across all corruptions. These results demonstrate that GMU-based architectures not only integrate seamlessly with existing TTA methods but also preserve their relative advantage over conventional networks under test-time distribution shifts.

Dataset	Standard			With TTA		
	Wins	Losses	Wilcoxon p	Wins	Losses	Wilcoxon p
MNIST	9	2	0.00488	7	2	0.06445
Fashion-MNIST	8	3	0.08740	7	2	0.01953
CIFAR-10	11	3	0.00336	13	0	0.00012

Table 7: Comparison of GMU-CNN vs CNN across datasets. Left: standard evaluation. Right: test-time adaptation (TTA). Wins/losses are counted per corruption type; p-values are from the one-sided Wilcoxon signed-rank test (GMU-CNN $>$ CNN).

3 Experiments on Synthetic Data

We design seven synthetic sampling scenarios to evaluate GMUs under controlled conditions, each highlighting a different structural or functional property of the data. These experiments test not only whether networks can exploit class-specific structural causal models (SCMs) and generalize to out-of-distribution shifts, but also whether they can capture functional and geometric structure such as Fourier expansions, k-subspace thresholds, polynomial boundaries, and Gaussian mixtures. Due to space constraints, we report in the main paper only the *Sparse Linear Structure Prediction* experiment, which directly probes the ability of GMUs to recover sparse latent causes that determine both inputs and labels. The remaining experiments, including Fourier series recovery, k-subspace thresholding, sparse neural structure sampling, tree-structured sampling, polynomial boundary sampling, and conditional Gaussian sampling, are presented in the Appendix with full details and results.

1. **Sparse Linear Prediction**: We evaluate whether linear GMUs retain their advantages when the underlying data is generated from a sparse linear process. This serves as a baseline test of their ability to exploit sparsity in settings where only a small subset of latent causes is active for each sample.

2. **Fourier series function v/s Noise Differentiation**: We show how single-layered GMU architectures can learn underlying function structures in the data, when the functions are represented generally as a Fourier series.

3. **k-subspace distance based thresholding**: We show that the problem that is the most natural test for a single-layered GMU architecture, k-subspace distance based thresholding, is significantly harder for other networks.

4. **Sparse Neural Structure Sampling:** Linear GMUs are motivated from a linear structural causal model's sampling process, as observed in Section 3. So, it is natural to ask whether the linear structure assumption in linear GMUs can still impart any benefits when the underlying data is generated non-linearly. Specifically, following the CL-SCM sampling as defined in Section 3, the generating function f in the SCM of Figure 1 (c) is set to a 2-layered neural network.

5. **Tree-structured sampling:** We evaluate GMUs on data generated via a synthetic SCM in which the generating causes are the observed input dimensions themselves; each class is associated with a distinct tree over the observed variables, and samples are generated by propagating from a root variable with additive noise.

6. **Polynomial Boundary Sampling**: Here, we consider the underlying ground truth distribution to be separable by a non-linear, polynomial decision boundary. Linear decision boundaries are not realistic for most natural datasets, and thus this provides a natural scalability test for GMUs.

7. **Conditional Gaussian Sampling:** This is a well-known and studied setting where the conditional distributions $P(X|Y = i)$ are all Gaussian. We vary the overlap between the Gaussians and test whether GMUs can robustly learn in this setting.

3.1 Sparse Linear Structure Prediction

Problem Outline. We argue that in many natural data sources, only a sparse subset of latent causes is active in generating any given observation. This intuition is consistent with prior work in sparse representation learning [Lee et al., 2006], where sparsity is viewed as a key inductive bias for capturing real-world structure. To test whether GMUs can leverage this property, we construct a synthetic scenario in which the set of active latent causes not only generates the observed input but also directly determines the class label. Concretely, this sampling process is a special case of the CL-SCM framework (Figure 2), where the class-wise models $SCM_1, SCM_2, \ldots, SCM_M$ may

Algorithm 1 Sparse Linear Structure Sampling

Setup: $X \in \mathbb{R}^d$, labels $y \in \{1, \ldots, C\}$, $W = [W_1, W_2, .., W_{N_{\text{total}}}] \in \mathbb{R}^{N_{\text{total}} \times d}$
For each class y**:** Select up to $N_{\max}$ active latent variables from $\{\theta_i\}_{i=1}^{N_{\text{total}}}$
Generative Process:
  Sample $y \sim \text{Unif}\{1, \ldots, C\}$
  Compute $x = \sum_{i=1}^{N_y} W_{g_y(i)}^T \theta_{g_y(i)} + \epsilon$
where $\epsilon \sim \mathcal{N}(0, \sigma^2 I_d)$

share common latent causes and weight structure. The generative process is formally described in Algorithm 1.

Setup					Test Config	GMU(1)	GMU(2)	GMU(3)	GMU(4)	GMU(5)	GMU(6)	MLP	MLP (norm.)	R-[1,1]	R-[1,1] (norm.)
C	N_{total}	d	σ	N_{max}											
20	25	10	0.01	3	same	0.8997	0.9311	0.9654	**0.9794**	0.9694	0.966	0.9505	0.9728	0.9622	0.9714
20	25	10	0.01	3	ood	0.685	0.84525	**0.9735**	0.8922	0.928	0.9442	0.5725	0.7087	0.631	0.6525
20	25	100	0.1	3	same	0.944	0.9271	0.9348	**0.9488**	0.9482	0.9482	0.9197	0.9451	0.946	0.9462
20	25	100	0.1	3	ood	0.8785	0.9005	**0.926**	0.848	0.8592	0.9092	0.6457	0.733	0.681	0.724
20	25	100	0.01	3	same	0.9471	0.9494	0.9814	**0.9874**	0.9862	0.9825	0.9545	0.9754	0.9654	0.9788
20	25	100	0.01	3	ood	0.892	0.9215	**0.981**	0.9335	0.9365	0.9612	0.6282	0.7492	0.693	0.7522
20	25	100	0	3	same	0.9474	0.9565	0.9962	0.9951	0.9974	**0.9991**	0.9548	0.9797	0.9622	0.9868
20	25	100	0	3	ood	0.892	0.925	**0.9977**	0.9915	0.9835	0.9905	0.6275	0.7495	0.6842	0.7432
20	25	500	0.01	3	same	0.9502	0.9434	0.9831	0.9891	**0.9894**	0.9874	0.9537	0.9771	0.9685	0.9845
20	25	500	0.01	3	ood	0.9087	0.9215	**0.9845**	0.9395	0.948	0.9655	0.59725	0.7677	0.6467	0.7665
20	25	1000	0.01	3	same	0.9111	0.9502	0.9834	0.9957	0.9914	**0.996**	0.9662	0.9814	0.9762	0.9845
20	25	1000	0.01	3	ood	0.8657	0.9017	0.975	0.9537	**0.987**	0.983	0.517	0.7525	0.6572	0.7837
20	25	1000	0.01	6	same	0.978	0.9385	0.9791	0.9908	0.9885	**0.9951**	0.9637	0.9882	0.9757	0.9882
20	25	1000	0.01	6	ood	0.9272	0.6867	0.9462	0.9462	0.9905	**0.9997**	0.6407	0.9107	0.641	0.904
50	10	1000	0.01	3	ood	0.5752	0.743	0.8997	**0.9077**	0.891	0.8712	0.5492	0.5575	0.4132	0.5352
Wins/Losses/Ties vs GMU(3)						14/1/0	15/0/0	–	7/7/1	6/9/0	6/9/0	15/0/0	12/3/0	14/1/0	10/5/0
Wilcoxon p-value						1.8e-04	6.1e-05	–	0.432	0.639	0.804	6.1e-05	8.4e-03	4.3e-04	1.8e-02

Table 8: Test accuracy results on the sparse linear structure prediction experiments.

Experimental Setup. We generate datasets by first sampling the generative weights W from a standard normal distribution $\mathcal{N}(0, 1)$. For each instance, the number of active causes N_y is drawn uniformly from $[1, N_{\max}]$, and the corresponding latent variables are activated to produce the observed input vector x. Unless otherwise specified, Gaussian noise with variance σ^2 is added to each dimension. To probe generalization, we vary the range of the latent variables $\theta_{l_{y(i)}}$ at test time, thereby creating OOD conditions where the test distribution differs from training. This allows us to evaluate whether models capture the underlying generative mechanism or merely memorize the training data.

Baselines and Models. We compare GMUs against strong feedforward baselines. Specifically, we evaluate a two-layer MLP with 512 ReLU-activated hidden units (denoted MLP-512) and a ResNet with two groups, each containing one block of 512 units (denoted ResNet-512-[1,1]). To ensure fairness, all models are trained with the same optimizer and learning rate schedule. We also evaluate normalized variants of the baselines, where inputs are scale-normalized and mean-adjusted, denoted with the suffix "(norm.)". For GMUs, we use a single GMU layer with d inputs and C outputs, denoted GMU(k), where k indicates the order of the GMU. This design allows us to directly test whether the inductive bias of GMUs provides an advantage over standard architectures.

Results. The results are summarized in Table 8. GMUs consistently outperform both MLP and ResNet baselines across parameter settings, with the gap most pronounced in OOD scenarios where test distributions differ from training. This suggests that GMUs capture aspects of the underlying generative process better than the other counterparts. We also find that higher-order GMUs ($k > N_{\max}$) perform comparably to $k = N_{\max}$, since larger-order GMUs can emulate lower-order ones by simply setting some weights to zero. The Wilcoxon signed-rank test shows no significant gains once k exceeds the true number of active causes, indicating that while overparameterization is not harmful, its benefits saturate beyond the underlying generative complexity.

Discussions. These findings highlight two key points. First, GMUs are well-suited for tasks where the discriminative signal is tied to a sparse generative structure, a property that is common in many real-world domains (e.g., vision). Second, the robustness of GMUs under OOD shifts suggests that their inductive bias aligns more closely with the true data-generating process in this setting. In Appendix 3, we extend this analysis to five other synthetic scenarios, which probe complementary aspects of GMU behavior such as functional approximation (Fourier series), geometric discrimination (k-subspace thresholding), non-linear generative sampling (sparse neural structure), hierarchical

observed-variable dependencies (tree-structured sampling), and non-linear decision boundaries (polynomial and Gaussian mixtures). As an example, although the experiment shown here assumes linear generative structures, even when the generative structures are non-linear (Appendix 3.4), GMUs can exploit local linearity and yield better generalization performance. Together, these experiments provide a broad and rigorous evaluation of GMUs under diverse synthetic conditions.

3.2 Fourier series function v/s Noise Differentiation

Problem Setup: We consider a scenario where $x \in \mathbb{R}^d$ can belong to two categories: Fourier series function or noise. In the noise category, x is generated as $x \sim \mathcal{U}(-0.5, 0.5)^d$, where $\mathcal{U}(a, b)$ denotes the uniform distribution sampled from the range (a, b). In the Fourier series function category, the ith dimension of x, x_i is generated as $x_i = a_0 + \sum_{j=1}^{K} a_j sin(10 * i/d) + b_j cos(10 * i/d)$. Here $a_0, a_1, ..a_K$ and $b_1, b_2, .., b_K$ are coefficients of the Fourier series representation of X. Thus, when x is a Fourier series function, the dimensions of x are essentially sampled from the outputs of a Fourier series representation. We consider $K = 2$ for our experiments. The objective of this experiment is to identify if x is a noisy signal, or has a function pattern via a Fourier series representation.

GMU construction: We design a GMU as follows.

$$\text{gmu}(x, w) = \exp\left(-\min_{\theta} \mathbb{E}_i \left[\left(f(\theta, w_i) - x_i\right)^2\right]\right),$$

where $f(\theta, w_i) = \theta_0 + \sum_{j=1}^{k} (w_i)^j \theta_j$. This GMU, in a nutshell, can find to what degree the input dimensions of $x = [x_1, x_2, .., x_d]$ can be represented as a polynomial function $x_i = f(\theta, w_i)$ of the weights.

Results: We compare the performance of a single GMU-k unit, where k is its order, with a single perceptron unit, a 2-layer μ-RBF networks having 512 μ-RBF units in the first layer [1], a relu-activated single hidden layer neural network with 256 units (MLP-256) and with 512 units (MLP-512). The results are shown in Figure 5a. We also report the parameter counts for comparison. Overall, we find that GMU-4 and GMU-5 quickly capture the underlying structure, achieving 100% test accuracy with only 50 samples. In contrast, we see that the MLP and RBF variants need more data and parameters to achieve similar performance.

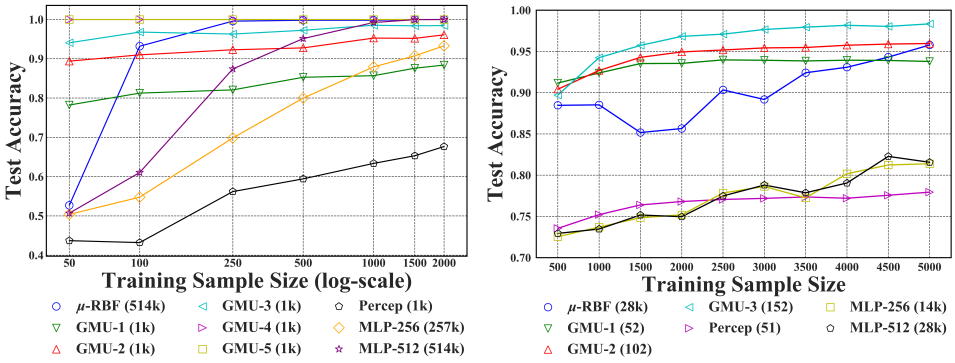
3.3 k-subspace distance based thresholding

Problem Setup: Just as a single-layer perceptron aligns with problems where the underlying distribution is linearly separable, and similarly, a single RBF neuron can perfectly classify data when the distribution is separable by a sphere, GMUs address an equivalent problem in a different geometric sense. We consider a supervised classification problem where the datapoint $x \in \mathbb{R}^d$ is categorized according to the distance it has to a k-dimensional subspace (i.e., k-subspace distance in Section 5.1), represented by the basis vectors as $U = Span\{u_1, u_2, .., u_k\}$, where $u_i \in \mathbb{R}^d$. Let the distance between U and x be denoted by $d(x, U)$. Then, given a threshold distance d_{thres}, we categorize x as the category 0 if $d(x, U) \leq d_{thres}$ and the category 1 if otherwise. To construct the training data, we sample x from the uniform distribution $x \sim \mathcal{U}(0, 1)^d$ and each element of $u_1, u_2, .., u_k$ is also chosen randomly from $\mathcal{U}(0, 1)$. Then, d_{thres} is chosen such the number of samples per class are same for both categories. We use single-layer linear GMU networks for classification. In this case, the shape of the decision boundary exactly aligns with thresholding the single GMU output.

Results: We compare the performance of a single linear GMU-k unit, with a single perceptron unit, 2-layer μ-RBF networks having 512 μ-RBF units in the first layer, a relu-activated single hidden layer neural network with 256 units (MLP-256) and another one with 512 units (MLP-512). The results are shown in Figure 5b. Unsurprisingly, we find that GMU-3 significantly outperforms all other approaches for all sample sizes. Interestingly, although we see the μ-RBF catching up to the GMUs performance with more data, but the MLP-variants still show significant drop in terms of accuracy, demonstrating the hardness of this function fitting task from the perspective of MLPs.

Takeaways: These results highlight the hardness of the task for traditional architectures. While GMU-3 consistently achieves high classification accuracy, MLP variants struggle to generalize, exhibiting a significant accuracy drop even with more data. Overfitting remains a concern, particularly for deeper networks, as they fail to capture the structured geometric separability essential for robust generalization.

[1] Note that we tried training with the additional σ term, but we got best results without it.



(a) **(b)**

Figure 5: Comparing test accuracy of a single GMU with other approaches on (a) Fourier series function v/s noise differentiation and (b) k-subspace distance based thresholding experiments. The parameter counts for each model are given within the brackets.

Algorithm 2 Sparse Neural Structure Sampling

1: **Setup:** $X \in \mathbb{R}^d$, labels $y \in \{1, \ldots, C\}$, class-specific weights $W_y^{(1)} \in \mathbb{R}^{N_{\text{total}} \times h}$, $W_y^{(2)} \in \mathbb{R}^{h \times d}$, latent variables $\theta \in \mathbb{R}^{N_{\text{total}}}$

2: **Latent Variable Sampling:** $\theta_i \sim \mathcal{U}(0, 1)$ for all $i \in \{1, \ldots, N_{\text{total}}\}$

3: **Generative Process:**

4: Sample $y \sim \text{Unif}\{1, \ldots, C\}$

5: Compute x:

$$x = \tanh\left(W_y^{(2)} \tanh\left(W_y^{(1)}\theta\right)\right) + \epsilon$$

6: where $\epsilon \sim \mathcal{N}(0, \sigma^2 I_d)$

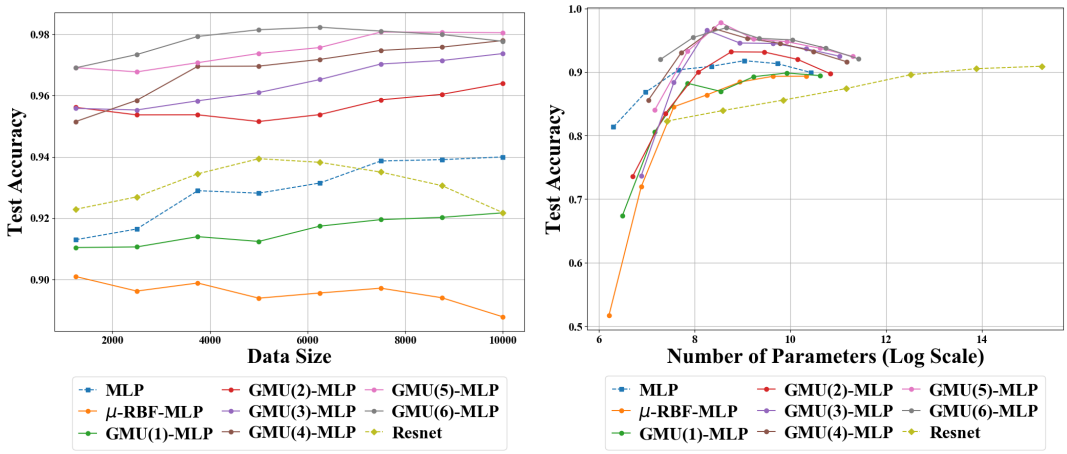
3.4 Sparse Neural Structure Sampling

Problem Outline: We consider the case when the underlying data follows a CL-SCM sampling process (Figure 1 (b)). However, unlike the sparse linear structure experiments in the main paper where the generating function f of the SCMs is linear, we consider non-linear f. In particular, we consider the case when f is a two layered neural network with tanh activations in each layer. Following the CL-SCM structure, each class has its own neural network weights it uses to generate the input. We formally outline the sampling process in Algorithm 2.

Experiments: Here, we focus on testing and comparing the scalability of GMUs with MLPs, RBFs and Resnet baselines, in two different settings. For each setting we consider two layered neural network SCMs with 32 hidden units ($h = 32$ in Algorithm 2), $\sigma = 0.01$, $d = 10$ and $C = 20$. In the first setting, we consider a total of 2 latent generating variables (i.e., $N_{total} = 2$ in Algorithm 2) and consider $N_{total} = 5$ for the second setting.

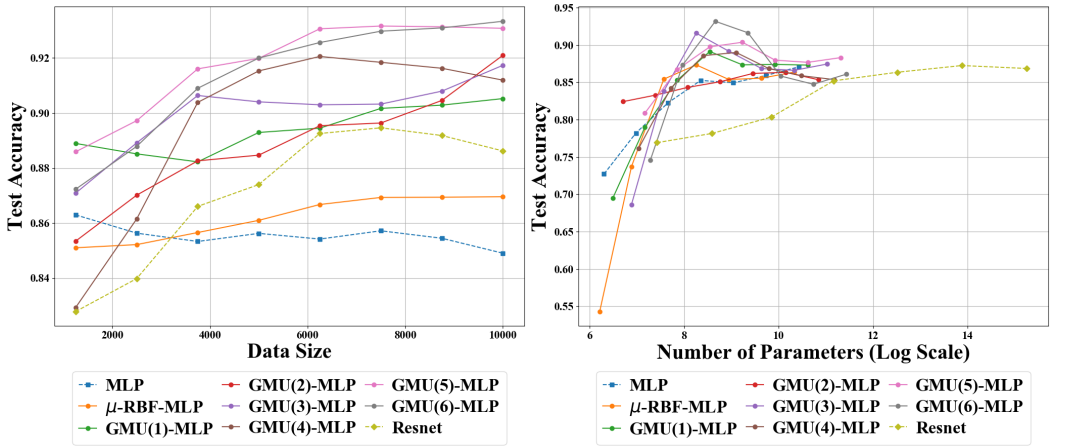
To rigorously assess whether the networks are truly learning the underlying function, we introduce a structured distribution shift: during training, latent variables θ_i are sampled from $\mathcal{U}(0, 0.5)$ or $\mathcal{U}(0.5, 1.0)$, while in testing, they are drawn from the opposite interval. This avoids any data overlap and thus prevents memorization as a reliable strategy to improve performance. In each setting, we plot the test accuracy trends in response to dataset size, and also plot the test accuracy trends of all networks w.r.t their complexity (parameter count). For the dataset size trends, each network's number of hidden neurons per layer was fixed to 256. The results are shown in Figure 6.

Observations: We find that GMU-MLPs show better scaling to data size in this scenario, when compared to the other baselines. Furthermore, we find that higher order GMUs showcase better performance overall. Interestingly, we see that the response to data size increase can even be negative for MLPs (e.g. when $N_{total} = 5$), which highlights the hardness of the generalization problem in this case. When considering the accuracy versus model complexity plots, we see that GMU architectures are closest to the pareto front, although there is a visible decrease in accuracy that accompanies GMU architectures when the parameter count increases beyond a point. Interestingly, the results hint at an optimal range of parameter count for the GMU architectures to significantly outperform the baselines.



(a) Accuracy vs. dataset size ($N_{total} = 2$).

(b) Accuracy vs. model complexity ($N_{total} = 2$).



(c) Accuracy vs. dataset size ($N_{total} = 5$).

(d) Accuracy vs. model complexity ($N_{total} = 5$).

Figure 6: Scalability studies in two experimental settings in the class-wise sparse neural structure sampling experiments. The underlying SCM here is a 2-layered tanh-activated neural network. We report accuracy trends in response to training dataset size changes in (a) and (c), and in response to parameter count changes in (b) and (d). Please note that only the MLP and Resnet baselines are represented via dashed lines.

3.5 Tree-structured sampling

Problem Setup. This synthetic experiment is a classification benchmark designed to probe GMUs under structured dependencies in which *observed variables themselves act as causal nodes*. Each class y is associated with a distinct directed spanning tree G_y defined over the d observed input coordinates in $X = [x_1, \ldots, x_d]$. Samples are generated by propagating a value sampled at the class-specific root down the directed edges with additive edge noise (Algorithm 3). This setting contrasts with latent-only CL-SCM setups because the causal graph is explicit over input dimensions; consequently the sampling process induces hierarchical linear relations among observed features that we expect GMUs to exploit. Unless stated otherwise we fix the root variance $\sigma_0 = 0.1$ and set the nominal edge coefficients $\alpha_i = 1$.

Tree construction and branching factor. Each class tree G_c is constructed at the start of a run by selecting a root uniformly from $\{1, \ldots, d\}$ and growing a directed spanning tree in a breadth-first fashion. The branching factor Δ controls how many children a node may receive during construction: larger Δ yields shallower, wider trees while smaller Δ yields deeper, narrower trees. We record the parent map $pa(\cdot, G_c)$ for each class and fix the tree topology per seed. Specifying Δ alongside d (as in Table 10) removes ambiguity about tree shape and clarifies how path length and noise accumulation vary across experiments.

Abbrv.	Description
G	Gaussian: $\epsilon_i \sim \mathcal{N}(0, 0.1)$
GS	Skewed Gaussian: $\epsilon_i \sim 0.1 \cdot \text{SN}(0, 1, 4)$
GM1	Gaussian with multiplicative scaling: $\epsilon_i \sim \mathcal{N}(0, 0.1)$ and $\alpha_i \sim \mathcal{N}(0, 1)$ (randomly sampled in each generative pass)
GM2	Gaussian with multiplicative scaling: $\epsilon_i \sim \mathcal{N}(0, 0.1)$ and $\alpha_i \sim \mathcal{N}(0, 4)$ (randomly sampled in each generative pass)
MG	Gaussian with fixed multiplicative scaling: $\epsilon_i \sim \mathcal{N}(0, 0.1)$, $\alpha_i \sim \mathcal{N}(0, 1)$ (Fixed)
MGN	Gaussian with noise-added multiplicative scaling: $\epsilon_i \sim \mathcal{N}(0, 0.1)$, $\alpha_i \sim \alpha_i^{\text{train}} + \mathcal{N}(0, 1)$ where α_i^{train} is the α_i fixed at training time

Table 9: Table of Abbreviations for the Dynamic tree prediction experiments.

Generative process and configurations. A datum is generated by sampling a class label $y \sim \text{Unif}\{1, \ldots, N_y\}$, sampling the root value $x_{r(G_y)} \sim \mathcal{N}(0, \sigma_0^2)$, and then for each non-root node i in topological order setting

$$x_i \leftarrow \alpha_i \, x_{pa(i,G_y)} + \epsilon_i,$$

with $\epsilon_i \sim P_\epsilon$. The experimental difficulty and the types of distributional shifts we evaluate are controlled via the choice of P_ϵ and whether the edge coefficients α_i are fixed or resampled. Table X (below) lists the abbreviations used throughout (G, GS, GM1, GM2, MG, MGN) and their precise sampling rules; for example, **G** denotes Gaussian edge noise with $\epsilon_i \sim \mathcal{N}(0, 0.1)$, while **GM1** and **GM2** introduce multiplicative scaling by resampling α_i at each generative pass from broader Normal distributions (see table for exact parameters). All shift configurations preserve the tree topology while changing effective coupling strengths or variances along edges, producing class-conditional but distributionally shifted test data.

Algorithm 3 Tree-structured sampling (SCM over observed variables)

Setup: input dimension d, number of classes C, branching factor $\Delta \in \mathbb{N}$, edge coefficients $\alpha \in \mathbb{R}^d$, root variance σ_0^2, edge-noise distribution P_ϵ
Tree construction (per class c):
  Pick a root node r_c uniformly from $\{1, \ldots, d\}$
  Initialize frontier $F \leftarrow \{r_c\}$, visited $V_c \leftarrow \{r_c\}$
while $|V_c| < d$ **do**
  Pop node u from F
  Sample $b \sim \min(\Delta, d - |V_c|)$ (number of children for u)
  Select b nodes uniformly at random from $\{1, \ldots, d\} \setminus V_c$ and add directed edges $(u \to v)$ to E_c
  Add selected nodes to V_c and append them to F
end while
Ensure $G_c = (V_c, E_c)$ is a directed spanning tree over $\{1, \ldots, d\}$
Generative process (sample one datapoint):
  Sample class label $y \sim \text{Unif}\{1, \ldots, C\}$
  Sample root value $x_{r_y} \sim \mathcal{N}(0, \sigma_0^2)$
for each non-root node i in topological order of G_y **do**
  Sample noise $\epsilon_i \sim P_\epsilon$
  Set $x_i \leftarrow \alpha_i \, x_{pa(i,G_y)} + \epsilon_i$
end for
Return: $(X = [x_1, \ldots, x_d], y)$

Train/Test nomenclature and reproducibility. The Train/Test columns in Table 10 indicate the generative configuration used for the corresponding split (for instance, Train = G, Test = GM1). When an experiment fixes α_i during training but resamples or perturbs it at test time we indicate this using the MG/MGN abbreviations described in the table. Trees $\{G_1, \ldots, G_{N_y}\}$ are resampled for each experimental seed and then held fixed within that seed to ensure repeatability of class-conditional structure across samples.

Experimental protocol. For the results reported in Table 10 we fix $N_y = 10$, $\sigma_0 = 0.1$, and generate 500 training and 3500 test samples per configuration. We compare GMU(k)-MLP-512 (a single GMU layer whose outputs feed a 512-unit linear head), MLP-512, ResNet-512-[2,1], and ResNet-512-[2,2]. All models are trained discriminatively using supervised cross-entropy; GMUs are not trained with any reconstruction objective in these experiments. Reported accuracies are averages over random seeds (tree topology, noise draws, and any sampled α_i variations) and the table explicitly lists the branching factor Δ used for each run.

Why Δ, tree shape and shift types matter. The branching factor and resulting tree geometry determine path lengths over which signal and noise accumulate: deeper trees propagate root information through more multiplicative steps, making downstream coordinates more sensitive to edge-wise scaling and noise; wider trees distribute signal across many shallow branches, changing how discriminative information is localized. The different abbreviations (G, GS, GM1, GM2, MG, MGN) correspond to structured perturbations that either alter noise skew/scale or resample multiplicative edge coefficients at train or test time (see the abbreviations table). These controlled variations let us separate robustness to noise shape (e.g., symmetric vs skewed) from robustness to multiplicative coupling changes.

Summary of empirical behavior. As reported in Table 10, higher-order GMUs consistently outperform lower-order variants and standard feedforward baselines, with gains growing at larger d and under several shift types. Notably, under structure-preserving but distribution-shifting perturbations (e.g., Train: G, Test: GM1/GM2), higher-order GMUs retain substantially more accuracy than MLPs and ResNets. Intuitively, the GMU's internal projection onto low-dimensional generative subspaces is better aligned with the hierarchical linear relations imposed by the tree, producing discriminative signals that are more stable when edge-wise couplings or variances change while the overall SCM topology remains unchanged.

3.6 Polynomial Boundary Sampling

Problem Setup: We consider the scenario where ground truth labels y are assigned based on a set of polynomial functions, with one function corresponding to each class. Given an input X, the label is chosen as the one whose polynomial function attains the highest value.

Definition 3.1. (Polynomial Boundary Sampling) Let $X \in \mathbb{R}^d \sim Unif(0,1)^d$, and $y \in \{1, 2, 3, \ldots, C\}$. We define a set of weight matrices $\{W_1, W_2, \ldots, W_d\}$, where $W_i \in \mathbb{R}^{N_y \times p}$. Each class y_i is associated with a polynomial function:

$$G(y_i|X) = \sum_{j=1}^{d} W_{ji}[X_j, X_j^2, \ldots, X_j^p]^T. \tag{8}$$

The label assignment follows a deterministic selection rule as follows:

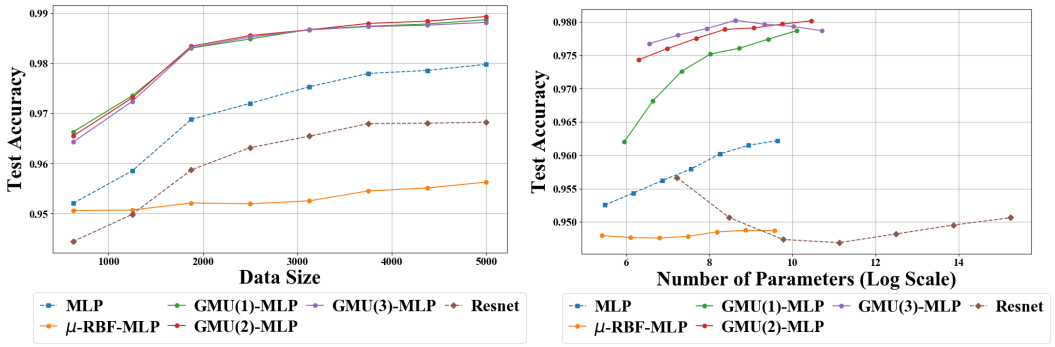
$$y^* = \underset{i \in \{1,2,3,\ldots,C\}}{\operatorname{argmax}} G(y_i|X). \tag{9}$$

Here, y^* denotes the assigned label based on the evaluated polynomials.

Remark 3.2. We consider the binary case, where two polynomial functions define the class boundaries. The label assignment follows: $y = 1$ if $G(y_1|X) - G(y_2|X) > 0$, and $y = 2$ otherwise. Since both $G(y_1|X)$ and $G(y_2|X)$ are polynomials of order p, their difference is also a polynomial of order p.

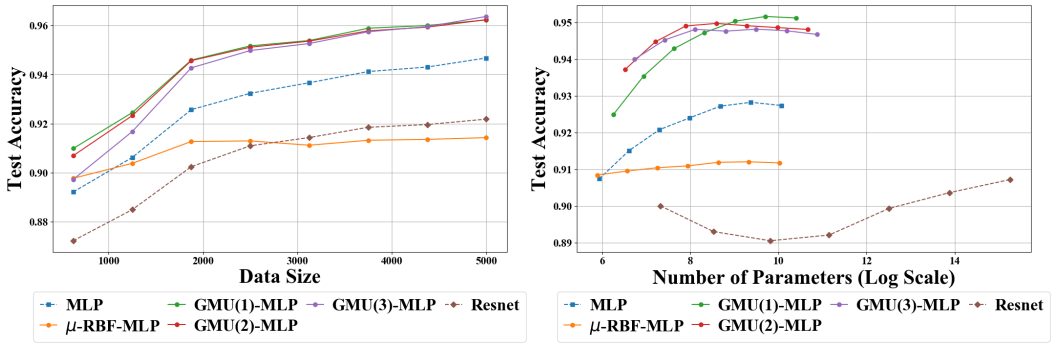
Setup ($N_y = 10, \sigma_0 = 0.1$)				GMU(0)	GMU(1)	GMU(2)	GMU(3)	GMU(4)	GMU(5)	GMU(6)	GMU(7)	GMU(8)	MLP	R-	R-
d	Δ	Train	Test	-MLP	-MLP	-MLP	-MLP	-MLP	-MLP	-MLP	-MLP	-MLP		[2,1]	[2,2]
10	2	G	G	0.5548	0.564	0.5608	0.5574	0.5608	0.5597	0.5605	0.5568	0.5571	**0.576**	0.5188	0.502
10	4	G	G	0.4362	0.4371	0.4277	0.4248	0.4265	0.4331	0.4342	**0.44**	0.4394	0.4345	0.3657	0.3525
50	4	G	G	0.6943	0.9286	0.9371	0.9420	0.9394	0.9403	0.9406	0.9394	**0.9440**	0.7965	0.8128	0.758
50	8	G	G	0.6403	0.8471	0.8686	0.8617	0.8649	0.8680	0.8651	**0.8714**	0.8686	0.7048	0.7194	0.6802
100	8	G	G	0.4503	0.9623	0.9703	0.9769	0.9769	**0.9806**	0.9797	0.9797	0.9783	0.7617	0.7931	0.7568
500	8	G	G	0.2371	0.9991	0.9991	0.9983	0.9980	0.9989	0.9997	0.9994	**1.0000**	0.7437	0.8102	0.772
500	8	GS	G	0.3980	0.7154	0.7829	0.5966	0.7009	0.7806	0.7777	0.8477	**0.8917**	0.4157	0.4122	0.3448
500	8	G	GM1	0.1583	0.7760	0.8054	0.7783	0.7883	0.7463	0.7963	**0.8374**	**0.8374**	0.29	0.2691	0.2317
500	8	G	GM2	0.2237	0.9929	0.9960	0.9954	0.9960	0.9943	0.9954	0.9971	**0.9983**	0.6531	0.6974	0.6551
500	8	MG	MGN	0.2071	0.9657	0.9586	0.9697	0.9551	0.9771	0.9789	0.9740	**0.9803**	0.6394	0.6471	0.6

Table 10: Test accuracy results on the dynamic tree structure prediction experiments.



(a) Accuracy vs. dataset size ($C = 2$).

(b) Accuracy vs. model complexity ($C = 2$).



(c) Accuracy vs. dataset size ($C = 10$).

(d) Accuracy vs. model complexity ($C = 10$).

Figure 7: Scalability studies in two experimental settings in the polynomial boundary sampling experiments. The underlying data distribution is separable via a non-linear decision boundary that is a polynomial function of the data dimensions. We report accuracy trends in response to training dataset size changes in (a) and (c), and in response to parameter count changes in (b) and (d). Please note that only the MLP and Resnet baselines are represented via dashed lines.

Setup p	GMU(0)	GMU(1)	GMU(2)	GMU(3)	GMU(0) -MLP	GMU(1) -MLP	GMU(2) -MLP	GMU(3) -MLP	Linear	MLP
1	0.8223	0.8406	0.8583	0.8643	**0.9843**	0.9763	0.9743	0.97	0.983	0.9823
2	0.9363	0.9363	0.9363	0.9363	0.9816	**0.9873**	0.986	0.982	0.9763	0.9801
3	0.7916	0.803	0.816	0.8336	0.9596	**0.975**	0.974	0.9746	0.9456	0.9606

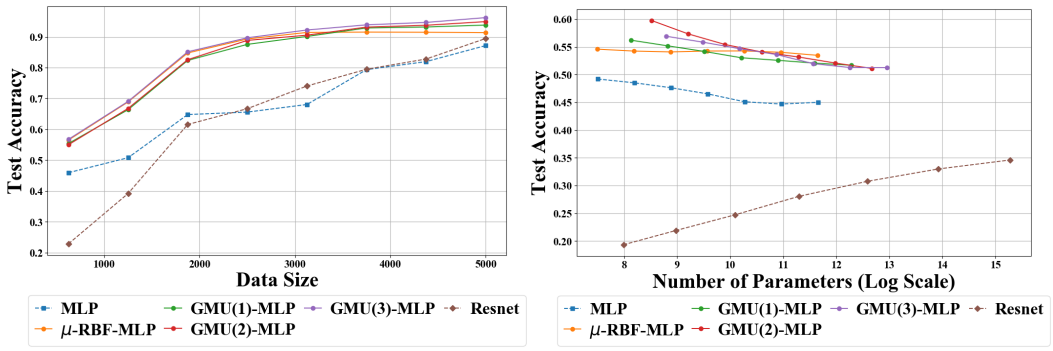
Table 11: Test accuracy results on the polynomial boundary sampling experiments.

This establishes a decision boundary that is a polynomial function of order p, thus yielding non-linear decision boundaries.

This is why we refer to the method as polynomial boundary sampling, as the underlying data distribution is separated by non-linear, polynomial decision boundaries.

Experiments: Each element in the weight matrices $\{W_i\}$ is randomly initialized from $\mathcal{N}(0, 1)$. We vary the polynomial order p while keeping $d = 10$ fixed. The following models are compared: GMU(k), GMU(k)-MLP-512 (GMU layer with 512 hidden units followed by a linear layer), a single linear layer, and MLP-512. Results are presented in Table 11. Interestingly, we find that as the polynomial order increases, GMU-MLPs of non-zero order show significant improvements on other baselines. To perform a deeper analysis, we conduct scalability studies for the higher order case $p = 3$ as follows.

Scalability Analysis: Similar to the sparse neural structure setting, we conduct experiments here to test the performance of networks in response to changes in scale of the dataset and the network complexity itself. We consider two settings. In both settings, we set the order of the polynomial $p = 3$ and input dimensionality $d = 10$. For the first setting, we set $C = 2$ and we set $C = 10$ for the second setting. Due to polynomial decision boundaries $C = 10$ should yield a harder classification problem. The results are shown in Figure 7.



(a) Accuracy vs. dataset size.

(b) Accuracy vs. model complexity.

Figure 8: Scalability studies for the conditional Gaussian sampling experiments. The underlying data distribution of each class is a Gaussian distribution with randomly set covariance matrix and mean. We report accuracy trends in response to training dataset size changes in (a) and in response to parameter count changes in (b). Please note that only the MLP and Resnet baselines are represented via dashed lines.

Setup			GMU(0) -MLP	GMU(1) -MLP	GMU(2) -MLP	GMU(3) -MLP	Linear	MLP	R-[1,1]	R-[2,2]
N_y	d	σ_μ								
2	10	0.01	**0.9942**	0.9888	0.9914	0.9825	0.6474	0.9848	0.9817	0.9782
10	10	0.01	**0.2714**	0.2608	0.244	0.2405	0.1262	0.1908	0.1582	0.1462
10	10	0.1	**0.8888**	0.8511	0.8254	0.8091	0.3862	0.7685	0.7585	0.7511
2	100	0.01	**0.7151**	0.704	0.6902	0.6851	0.5577	0.6077	0.5694	0.5594
2	100	0.1	0.9414	0.9428	**0.9462**	0.946	0.782	0.9248	0.9031	0.89
10	100	0.1	**0.5145**	0.4985	0.4897	0.4897	0.2234	0.4068	0.3214	0.2674
2	500	0.1	0.7377	0.7411	0.7388	**0.7454**	0.6751	0.7285	0.6905	0.6848
2	500	1	0.986	0.9911	**0.99514**	0.9951	0.9891	0.9928	0.972	0.9794

Table 12: Test accuracy results on the conditional Gaussian sampling experiments.

We find that GMU-MLPs show significantly higher generalization performance in this case. For the datasize scaling experiments, we see performance of all networks improve in response to increasing data size, however, the GMU-MLP variants always show a clear improvement. Similarly, for the parameter count versus test accuracy plots, we see that for the same parameter count, GMU-MLPs show significantly higher test accuracy and seem to establish the pareto front. These results highlight the flexibility of GMUs in learning non-linear decision boundaries.

3.7 Conditional Gaussian Sampling

Problem Setup: We conduct a simple experiment where the conditional distributions $P(X|y)$ are Gaussian, where $X \in \mathbb{R}^d$ and $y \in \{1, 2, .., C\}$. Specifically, we generate $P(X|y) \sim \mathcal{N}(\mu_y, \Sigma)$. For each dataset, we choose the class-wise mean values by randomly generating them as $\mu_y \sim \mathcal{N}(0, \sigma_\mu^2 I_d)$. Similarly, we pick a randomly generated covariance matrix via $\Sigma = A^T A$ where $A \sim Unif(0, 1)^{d \times d}$.

Experiments: We pick a range of parameter choices for the sampling process, and compare the following networks: GMU(k)-MLP-512 (GMU layer with 512 hidden units followed by a linear layer), MLP-512, Resnet-512-[2,1] and Resnet-512-[2,2]. The results are shown in Table 12, for a wide range of parametric choices. We find that overall, μ-RBF networks (GMU(0)-MLP) showcase superior performance. This is unsurprising, as the distance based μ-RBF units are closely aligned with the underlying problem. We also see that for higher dimensionality scenarios, GMU-MLPs can often outperform μ-RBFs. Furthermore, we see that in most cases, GMU-MLP performance for non-zero order is competitive with μ-RBFs, which can also be explained via the universal approximation ability of GMU-MLPs (Proposition 5.1).

Scalability Analysis: Similar to the previous settings, we conduct experiments here to test the performance of networks in response to changes in scale of the dataset and the network complexity itself. We consider the setting where the input dimensionality $d = 100$, $\sigma_\mu = 0.5$ and the number of output classes $C = 10$. The results are shown in Figure 8.

We find that GMU-MLPs and μ-RBFs show better performance than the other baselines. For the datasize scaling experiments, we see performance of all networks improve in response to increasing sample size, and the performance seems to converge for larger sample size. Interestingly, μ-RBF's

test accuracy seems to plateau for larger data sizes, whereas GMU-MLPs continue to increase. For the parameter count versus test accuracy plots, we again see that for the same parameter count, μ-RBFs and GMU-MLPs showcase significantly better performance overall. Interestingly, we see that the best performance is achieved for lower parameter counts in this setting, and GMU-MLPs showcase slightly better test accuracy than μ-RBFs.

4 Ablations and Further Insights

In this section, we provide detailed ablations to isolate the factors contributing to the GMU's performance. Specifically, we investigate whether the robustness of GMUs is merely a byproduct of normalization and explore the benefits of depth in generative matching architectures.

4.1 Isolating Effects of Patch Normalization

A natural question arises: is the superior robustness of GMU-CNNs simply due to the implicit normalization inherent in the distance calculation? To isolate the effect of point-to-manifold matching, we compare the GMU-CNN against a *Mean-SD Normalized CNN*. In this baseline, every input patch is mean-centered and variance-normalized before being processed by a standard convolution. To ensure numerical stability, a noise term of $\epsilon = 10^{-4}$ is added during normalization. This architecture provides a like-for-like comparison by mimicking the moment-matching properties of GMUs without the manifold-reconstruction mechanism.

Dataset	TTA	Standard Accuracy	Mean Corruption Acc. (Base)	GMU-CNN (Ours)
MNIST	No	0.9936	0.8635	**0.9434**
	Yes	0.9937	0.9461	**0.9678**†
Fashion-MNIST	No	0.9133	0.6707	**0.7558**
	Yes	0.9133	0.7556	**0.8122**†

Table 13: Comparison between Mean-SD Normalized CNNs and GMU-CNNs. † indicates results from the extended Test-Time Adaptation (TTA) experiments in Appendix 2.

As shown in Table 13, the Mean-SD Normalized CNN performs significantly worse than the GMU-CNN across all settings. Even with the aid of Test-Time Adaptation (TTA), the normalized baseline fails to reach the intrinsic robustness levels of our approach. This confirms that the GMU's advantage is derived from its **point-to-manifold geometry** and its ability to achieve Bayes-optimality under structured sampling, rather than simple input scaling.

4.2 Stacked GMU Architectures

While stacking RBF-style units typically leads to gradient collapse due to nested exponentials, we leverage our theoretical results in Section 3 to design a two-layer GMU stack (GMU_2). Following Proposition 2.4, we remove the explicit activation function ϕ in the final layer to align the unit output with the softmax operator.

We evaluate a configuration where Layer 1 uses $k = 5$ with exponential activation, and Layer 2 uses $k = 3$ with no activation. We compare this against a hybrid GMU-MLP baseline (one layer of GMUs followed by a linear layer) across various data regimes. All hyperparameters were optimized on a separate validation set when trained on MNIST-1k and fixed subsequently.

Analysis of Results:

- **Statistical Significance:** As detailed in Table 14, the improvement of GMU_2 over GMU-MLP is statistically significant in 10 out of 12 settings. This confirms that the generative matching principle can be effectively extended to multi-layer hierarchies.
- **Optimization Stability:** Crucially, the stacked GMU architecture consistently exhibits tighter confidence intervals (CI) than the GMU-MLP baseline. This suggests that the manifold-to-manifold mapping in deep GMU layers facilitates more stable and predictable optimization than traditional linear transformations.

Dataset	Model	Standard Test Accuracy (%)	Mean Corrupted Acc. (%)
MNIST-1k	GMU-MLP	89.98 ± 0.27	77.89 ± 1.23
	GMU$_2$	$\mathbf{91.63 \pm 0.08}$	$\mathbf{81.67 \pm 0.28}$
MNIST-5k	GMU-MLP	94.54 ± 0.13	81.10 ± 0.91
	GMU$_2$	$\mathbf{95.00 \pm 0.03}$	$\mathbf{83.24 \pm 0.53}$
MNIST-10k	GMU-MLP	95.89 ± 0.08	80.87 ± 0.71
	GMU$_2$	$\mathbf{96.06 \pm 0.07}$	$\mathbf{81.47 \pm 0.50}$
F-MNIST-1k	GMU-MLP	79.00 ± 0.30	61.64 ± 1.04
	GMU$_2$	$\mathbf{80.71 \pm 0.11}$	61.57 ± 0.49
F-MNIST-5k	GMU-MLP	80.45 ± 0.29	60.38 ± 1.75
	GMU$_2$	$\mathbf{83.57 \pm 0.12}$	$\mathbf{66.31 \pm 0.37}$
F-MNIST-10k	GMU-MLP	84.22 ± 0.11	62.11 ± 0.47
	GMU$_2$	$\mathbf{85.64 \pm 0.11}$	$\mathbf{68.66 \pm 0.29}$

Table 14: Performance of stacked GMU architectures over 10 independent trials (95% Confidence Intervals).

4.3 Analysis of GMU Order k

While a closed-form approach for estimating the optimal GMU order k remains an open research question, our extensive empirical observations across synthetic, tabular, and vision benchmarks consistently indicate that higher k values are generally "safer" and more robust, with rare exceptions such as SmallNORB. We demonstrate this trend through multiple ablations in our synthetic experiments (see Tables 1, 10 and Figures 4, 5). Due to computational constraints, we constrained our current testing to a maximum of $k = 8$.

Beyond empirical performance, we highlight two structural reasons why a larger k is theoretically preferred:

- **Subsumption Property:** A GMU of order k_1 can always emulate a GMU of order k_2 (where $k_1 > k_2$) by either setting specific weight columns to zero or reducing the rank of the weight matrix to k_2. This ensures that a higher-order GMU maintains a hypothesis space that strictly includes all lower-order configurations.
- **Representational Flexibility and Diversity:** Larger k allows individual units to span subspaces of any dimensionality up to k, facilitating greater diversity in the manifolds captured across the network.

4.3.1 Validating Flexibility via Stable Rank Analysis

To test the hypothesis that larger k provides increased expressive flexibility, we analyzed the **stable rank distribution** of the first-layer weight matrices in a GMU-CNN trained on MNIST-10k. In this setting, each unit processes 3×3 input patches vectorized to $\mathbb{R}^9$. Consequently, the weight matrix for each unit is $W \in \mathbb{R}^{9 \times k}$. The stable rank of these matrices serves as a proxy for the effective dimensionality of the manifold represented by each unit.

Our analysis confirms this intuition: as illustrated in Figure 9, higher k values lead to a significantly greater variance in stable rank across the filter bank. This indicates that higher-order GMU networks possess the expressive flexibility to **adaptively capture manifolds of varying dimensionalities** within the same layer. Rather than utilizing the extra capacity uniformly, the network "right-sizes" its representation for each filter; some units may model low-rank edge structures while others utilize the full capacity to capture more high-dimensional local variability.

4.3.2 Ablation of Order k on Vision Benchmarks

To isolate the impact of the manifold order k from other architectural factors, we perform a direct ablation on **MNIST-10k** and **Fashion-MNIST-10k**. In these experiments, we vary k from 0 to 7. To ensure a rigorous comparison with the $k = 0$ case (which corresponds to an μ-RBF unit), we mean-center and variance-normalize the inputs for the baseline. Consequently, any performance gains observed for $k > 0$ are solely attributable to the inclusion of the GMU's internal k-dimensional latent subspace.

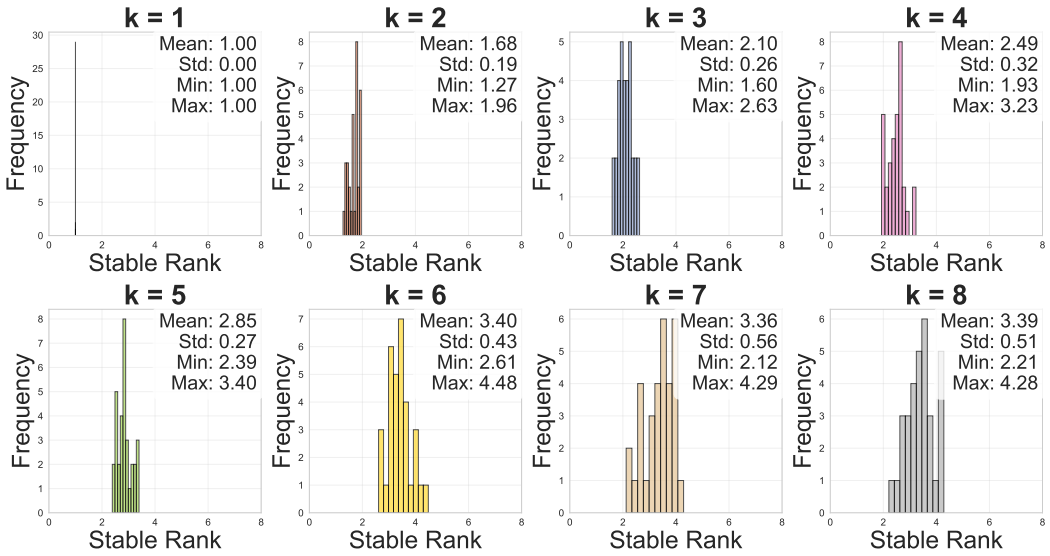


Figure 9: Distribution of stable ranks for 3×3 GMU filters on MNIST-10k. As k increases toward the patch dimensionality ($d = 9$), the network exhibits a significantly broader variance in rank, adaptively spanning the available subspace to represent local geometric features of varying complexity.

	MNIST-10k		**Fashion-MNIST-10k**	
GMU Order (k)	**Standard**	**Corrupted**	**Standard**	**Corrupted**
CNN (Baseline)	0.9833	0.5716	0.8822	0.4931
0 (mean-sd normalized μ-RBF)	0.9837	0.8187	0.8877	0.6762
1	0.9870	0.8520	0.8921	0.6853
2	0.9876	0.8460	0.8924	0.6923
3	**0.9879**	0.8799	0.8957	0.6876
4	0.9867	0.8594	0.8911	0.6752
5	0.9866	0.8692	0.8892	**0.6974**
6	0.9878	0.8568	**0.8958**	0.6702
7	0.9868	**0.8810**	0.8915	0.6480

Table 15: Impact of manifold order k on test accuracy. Corrupted accuracy represents the mean across 9 corruption types. In all cases, $k > 0$ configurations outperform both the standard CNN and the μ-RBF ($k = 0$) baseline.

Observations: The results in Table 15 confirm that increasing k consistently improves performance over standard first-order and isotropic biases. We observe that:

- **General Superiority:** In every experimental setting, the accuracy achieved with $k > 0$ exceeds that of both the standard CNN and the $k = 0$ baseline.
- **Robustness Gains:** The most significant improvements are observed in corrupted data regimes, where higher values of k (e.g., $k = 7$ for MNIST and $k = 5$ for Fashion-MNIST) provide the network with the geometric flexibility necessary to maintain discriminative power under distribution shift.
- **Non-Monotonicity:** While higher k is generally "safer," the optimal value varies slightly between datasets, suggesting that k acts as an expressive ceiling that the network utilizes adaptively depending on the local complexity of the data.

4.4 Verifying the CL-SCM Assumption

A core element to our proposed unit is the CL-SCM assumption, that states that there exists a functional generative mechanism that is class-dependent for most natural datasets. The original assumption holds for any general function f of the latent variables θ, but as in our work we focus

on linear f, in what follows we conduct experiments to verify this assumption in the context of linear generation. We focus on two ways to check this alignment: A Linear Probe Analysis and Single-Layer Performance.

Linear Probe Analysis: A core motivation for utilizing GMUs in the initial layers of a network is their ability to adhere to the underlying manifold structure of the data. Theoretically, a unit that accurately captures the generative geometry of the input should produce a representational space that is more linearly separable than one produced by standard first-order transformations. To test this, we conduct a **linear probing analysis** on the first 10 tabular datasets from our benchmark suite.

We freeze the weights of the first layer and train a linear classifier (probe) on its output. We compare the probing accuracy of a standard MLP first layer (Baseline) against GMU layers of varying manifold order $k \in \{0, 1, 2, 3\}$. The results, averaged across the 10 datasets, are summarized in Table 16.

First Layer Configuration	**Linear Probe Accuracy (Layer 1 Output)**
MLP (Baseline)	0.8771
GMU ($k = 0$)	0.8319
GMU ($k = 1$)	0.9004
GMU ($k = 2$)	0.9101
GMU ($k = 3$)	**0.9121**

Table 16: Mean linear probing accuracies across 10 tabular datasets. All GMU configurations with $k \geq 1$ yield representations that are significantly more linearly separable than the standard MLP baseline.

Dataset	Model Architecture	Standard First Layer	GMU First Layer ($k = 3$)
MNIST	4-layer CNN	94.94%	**95.28%**
Fashion-MNIST	4-layer CNN	84.47%	**89.72%**
SVHN	ResNet-18	72.86%	**82.64%**
CIFAR-10	VGG-16	45.50%	**51.22%**

Table 17: Linear probe test accuracies across vision datasets. Architectures with a GMU first layer ($k = 3$) consistently outperform their standard first-layer counterparts across all datasets.

We also test the linear probe experiments on vision data. Experiments are conducted on MNIST (4-layer CNN), Fashion-MNIST (4-layer CNN), SVHN (ResNet-18), and CIFAR-10 (VGG-16) with the same dataset-specific architectures used in the vision experiments in the main . Each architecture has its GMU variant where the first layer is replaced by an equal number of GMU units of order $k = 3$ (all of same configuration as used in the vision experiments in the main mody). We outline the linear probe test accuracy results in Table 17.

We see that in every case the GMU feature outputs are more linearly separable and generalize better than the standard convolutional counterparts. For there to be a significant accuracy difference with only the first layer outputs alone suggests that the class-wise structured linear generative assumption in CL-SCMs has visible alignment with the data.

Single-Layer Architecture Performance: Another way we can probe this phenomenon is via directly training single layer GMUs and comparing with linear perceptrons, in a like-for-like parametric setting. We compare a standard perceptron with a basic GMU with zero bias and no normalization in eq (2). Furthermore, for parametric fairness, we use the polynomial expanded form of the weights $W = [w, w^2, ..w^k] \in \mathbb{R}^{d \times k}$ for the k-order GMU. Following the construction motivated in Proposition 3.4 we choose an identity activation $\phi(x) = x$. For both networks a batch normalization layer is added before the softmax to accelerate training. This is the same configuration we used in our stacking experiment (4.2) where the last layer was a GMU instead of a linear layer. Both networks are trained for 200 epochs to ensure convergence.

Results are reported in Table 18. We find that not only do GMUs generalize significantly better than a standard perceptron across all benchmark datasets, but scaling to higher-order GMUs ($k > 1$) systematically improves test performance without introducing any additional parameter overhead. This indirectly highlights the alignment of the structured generative assumptions in the CL-SCM

model with real data. We report performance on the 10k subsets of four datasets: MNIST, SVHN, CIFAR-10 and Fashion-MNIST.

Model Architecture	GMU Order (k)	MNIST-10k	F-MNIST-10k	SVHN-10k	CIFAR10-10k
Standard Perceptron	—	0.8461	0.7601	0.0893	0.2979
GMU	0	0.8504	0.8027	0.2264	0.3475
GMU	1	0.9064	0.8250	0.2580	0.3893
GMU	2	0.9085	0.8226	0.2527	0.3904
GMU	3	0.9143	0.8319	0.4262	0.4099
GMU	4	0.9134	0.8284	**0.4973**	0.4206
GMU	5	**0.9231**	**0.8358**	0.4895	**0.4245**

Table 18: Single-layer architecture performance comparison across 10k subsets of MNIST, Fashion-MNIST, SVHN, and CIFAR-10. Scaling to higher-order GMUs ($k > 1$) systematically improves test performance.

Analysis and Insights: The results demonstrate that introducing manifold-aware bias via the GMU significantly impacts the representational quality from the very first transformation. While the $k = 0$ case (which reduces to an isotropic distance) is insufficient for complex tabular data, even a first-order manifold ($k = 1$) outperforms the MLP baseline.

This improvement in linear separability confirms the CL-SCM alignment for real data: by treating the forward pass as a distance-to-manifold calculation rather than a simple hyperplanar projection, the GMU pre-conditions the data into a more discriminative space. This reduces the representational burden on subsequent layers, explaining why GMU-based architectures achieve higher total network accuracy and more stable optimization paths.

4.5 Mechanistic Explanation of GMU Robustness

In this section, we pursue a mechanistic explanation and interpretation of why GMUs are showing better robustness across our experiments, such as to out-of-distribution transformations at test time. We hypothesize that this added robustness stems from the ability of individual GMUs to ignore noise that preserves the distance to the manifold. Following our theoretical results, a GMU's output-preserving contours (i.e. level sets) resemble oriented hypercylinders with k flat directions and $d - k$ curved ones. While one could argue that standard units (e.g., RBFs or MLPs) also possess certain geometric invariances, the critical distinction is that the GMU's invariance is explicitly rooted in a structured assumption of data generation (CL-SCM). Furthermore, as the previous section demonstrates that this assumption aligns well with both tabular and vision data, we consider this the key reason for the higher robustness.

Intuition: Learning Topology via Weights. However, our objective here is to chase a deeper mechanistic explanation of the nature of information captured by GMUs. In the case of vision datasets, we hypothesize that GMUs capture spatial structure uniquely, in a way that improves their robustness to out-of-distribution data. To intuitively understand how GMUs capture spatial structure, we refer to our synthetic Fourier Series vs. Noise experiment (see Appendix 3.2). The GMUs used there were of order k, with non-zero bias, no normalization and polynomial expanded weights. The GMU function there is:

$$\mathrm{gmu}(x, w) = \exp\left(-\min_{\theta} \mathbb{E}_i \left[(f(\theta, w_i) - x_i)^2\right]\right)$$

where $f(\theta, w_i) = \theta_0 + \sum_{j=1}^{k} (w_i)^j \theta_j$ and x_i is the ith input dimension.

Here is the main mechanism: if the network learns weights that respect the underlying spatial structure (e.g., $w_i \propto i$), the inner minimization essentially becomes a k-degree polynomial regression of the input x over its spatial coordinates. If x is a smooth spatial function, the polynomial fits well, the mean squared error is near zero, and the GMU activation is high (≈ 1). However, if x is random noise w.r.t its spatial structure, it cannot be fit by a smooth low-degree polynomial that follows the same spatial structure via the weights. The fitting error becomes large, and the negative exponential drastically suppresses the GMU output toward zero.

Thus, by simply aligning its weights with the underlying spatial topology, the GMU naturally acts as a structural filter. This enables our GMU variants to differentiate smooth functions from noise with 100% accuracy using only 50 samples (Fig. 5a), significantly outperforming MLP and RBF baselines.

Activation $\phi(x)$	**MNIST-10k** (μ**-RBF**)	**MNIST-10k** ($k = 3$)	**CIFAR10-10k** (μ**-RBF**)	**CIFAR10-10k** ($k = 3$)
Exponential (e^{-x})	98.39%	**98.64%**	57.36%	58.18%
Square Root ($\sqrt{1-x}$)	98.37%	98.51%	55.79%	**59.10%**
Logarithmic ($-\log x$)	98.23%	98.44%	55.12%	58.00%
Negation ($-x$)	98.17%	98.48%	55.78%	58.43%
*Note: CNN Baseline achieves **98.24%** on MNIST-10k and **57.28%** on CIFAR-10-10k.*				

Table 19: Sensitivity analysis of GMU-CNN architectures across four activation functions on MNIST-10k and CIFAR10-10k datasets.

Empirical Verification on Vision Models: To conclusively verify that this adaptation to spatial structure drives robustness in deep networks, we analyzed the learned representations of ResNet-18 (Standard vs. GMU-first-layer) trained on SVHN across three targeted experiments. These summarize our findings:

1. **Loss of Spatial Structure (Pixel Shuffling)**: When test images are randomly pixel-shuffled, thus removing any spatial structure while preserving pixel intensity distributions, the standard CNN maintains 96.5% of its baseline activation magnitude. In contrast, the GMU's activation magnitude collapses to 36.4%, with inactive neurons jumping from 36.1% to 63.4%. The GMU recognizes the loss of structure and significantly dampens the output signal.

2. **Scale-Invariant Weight Smoothness:** To measure whether GMU weights better adapt to the underlying spatial structure, we computed a *Normalized Spatial Smoothness*:

$$\frac{\mathbb{E}\left[(w_i - w_{N(i)})^2\right]}{\mathrm{Var}(w)},$$

   where $N(i)$ denotes the number of spatial neighbors of i within each 7×7 kernel. This scale-invariant metric essentially measures to what degree the network weights adhere to the underlying spatial topology of images. GMUs yield an average smoothness of 1.6429 compared to CNN's 2.6574, confirming that GMU weights have a significantly higher spatial adherence to the underlying spatial structure of the inputs.

3. **Invariance to Monotonic Shifts:** Subjecting test images to an order-preserving monotonic transformation ($f(x) = x^{2.5}$) disrupts raw pixel intensities but perfectly preserves underlying spatial concepts. Under this shift, we find the CNN's feature cosine similarity to the untransformed inputs drops to 0.795, whereas the GMU remains significantly invariant with a cosine similarity of 0.967.

These findings confirm that the robustness primarily stems from the ability of GMUs to ignore manifold-distance-preserving noise. And following that, as our analysis shows, this specific invariance is effective in practice precisely because it is rooted in a structured data generation assumption (CL-SCM) that naturally aligns with the properties of structured datasets like vision with an underlying spatial topology.

4.6 Activation Function Ablations

In the main text body, we used the standardized exponential activation function ($\phi(x) = e^{-x}$) for both RBFs and GMUs for all real-world dataset experiments. This was done to ensure consistency and to ensure our performance gains were not the result of activation tuning. From a kernel theory perspective, exponential activation is a natural first choice, as it ensures a positive semi-definite Gram matrix and prevents exploding outputs (unlike the logarithmic activation).

However, this leads to the natural question of whether other activation function choices are just as robust or if they significantly impact performance negatively. Here, we test the sensitivity of the GMU-CNN architecture to four distinct activation functions on the MNIST and CIFAR-10 datasets: exponential (e^{-x}), logarithmic ($-\log x$), square root ($\sqrt{1-x}$), and negation ($-x$). Other than the

choice of activation, we used the same GMU configuration from our vision experiments. We restricted the focus to shallow 3-layer CNN architectures trained on 10k subsets of MNIST and CIFAR-10, comparing $k = 3$ GMUs against $k = 0$ (μ-RBF) units.

Results are summarized in Table 19. The results yield two key insights:

1. The order $k = 3$ GMU consistently outperforms both the baseline CNN and the $k = 0$ (μ-RBF) unit across every activation choice. Because distance $d \geq 0$ is inherently non-negative, any monotonic decay function $\phi(d)$ preserves the topological ordering of the k-subspace distance (mapping smaller distances to higher activations). This confirms that the representational gains stem mainly from the k-subspace geometry itself, rather than the choice of activation.
2. While the exponential function remains highly stable and justifies its selection as our default, the architecture is robust across choices (e.g., the square root activation yields the highest performance on CIFAR-10 at $k = 3$).

5 Discussions: Contextualizing GMUs in Literature

In this section, we discuss GMUs in the context of existing architectures for supervised learning beyond perceptron-based neural networks. Capsule networks have been proposed in recent years as an alternative learner incorporating feedback. Each hidden node is a *capsule* that contains a vector of activations, and routing iterations adjust the coupling coefficients, directing inputs to hidden units with greater *agreement*. While this introduces a feedback mechanism, it does not function as the internal generative model like GMUs. Furthermore, after routing iterations, the forward computation remains linear, followed by squashing normalization, meaning the complexity bias is still present in the forward pass [Sabour et al., 2017].

Another interesting alternative to multi-layer perceptrons (MLPs) is Kolmogorov-Arnold Networks (KANs, Liu et al. [2025]), which leverage the Kolmogorov-Arnold approximation theorem [Kolmogorov, 1957] to construct more expressive units. However, KANs primarily focus on function approximation and retain a fundamentally feedforward computational paradigm, contrasting with GMUs, which primarily look for dynamic generative structures in the input.

GMUs are also distinct from architectures that rely on weight adaptation mechanisms, such as hypernetworks [Ha et al., 2016] or dynamic convolution [Chen et al., 2020]. Unlike these approaches, which dynamically adjust weights, GMUs are designed to reconstruct the input based on projections onto internal k-dimensional subspaces, effectively representing reconstruction errors. This makes GMUs complementary to these models rather than functionally equivalent, as hypernetworks, steerable filters, and similar methods can be used in conjunction with GMUs, just as they would with perceptron or radial basis function (RBF) networks.

Furthermore, GMUs differ structurally from linear self-attention mechanisms [Wang et al., 2020]. Attention mechanisms are fundamentally designed for sequences: they operate by relationally grouping ordered inputs, and even in vision transformers (ViTs,Dosovitskiy et al. [2021]), images are reshaped into patch-based sequences with positional encodings before attention is applied. The goal of attention is thus to dynamically weight interactions across sequence elements. By contrast, GMUs emphasize the structured generative properties of the input and act as direct analogues of classical computational units such as perceptrons or RBFs. Each GMU has its own set of weights and biases and produces an activation response, which in the linear case is simply obtained by computing a k-subspace distance followed by an activation function. This generalizes Euclidean distance (used in RBFs) and positions GMUs as a new type of unit at the same architectural level as perceptrons, designed not for sequence modeling but for robust supervised function fitting with explicit theoretical guarantees.

Importantly, just as attention computes similarity via query–key dot products:

$$\text{score}(q, p) = \frac{q^\top p}{\sqrt{d}}, \tag{10}$$

recent variants such as Euclidean Attention and Elliptical Attention (Nielsen et al. [2024]) demonstrate that the score can be defined by alternative distance metrics. In this broader context, one can analogously define a GMU-based attention score using k-subspace distance:

$$\text{score}_{k,W}(q, p) = \phi\left(\frac{1}{d} \min_{\theta \in \mathbb{R}^k} \left\|q - p - W^\top \theta\right\|^2\right). \tag{11}$$

As we have shown in this paper, for linear GMUs the k-subspace distance admits a closed-form solution (Eq. (3)) and can be efficiently parallelized using tensorized computation with Einstein summations, enabling real-time inference. GMUs can thus be integrated into attention frameworks in two complementary ways: (i) by replacing the similarity score with a generative subspace projection, or (ii) by substituting selected MLP layers within attention blocks with GMU layers. We consider these integration pathways as important directions for future work.

Test-time adaptation (TTA) techniques [You et al., 2021, Lim et al., 2023] provide another line of work relevant to improving robustness, as they adapt model statistics to distribution shifts encountered during inference. However, these methods do not introduce fundamentally new computational units; rather, they modify existing architectures to better handle domain variability. As we've seen in Appendix 2, TTA approaches are complementary to GMUs: while GMUs define a novel unit based on generative subspace projections, TTA can be applied on top of GMU-based networks to further enhance adaptability across domains.

Generative Capsule Models (GCMs): Generative Capsule Models (GCMs, Nazabal et al. [2023]) provide an interesting point of comparison to GMUs, since both involve latent linear models for capturing appearance variability. However in GCMs, part-based appearance templates and geometric transformation components are optimized jointly to minimize reconstruction error across samples, yielding structured generative representations. GMUs, by contrast, are not trained to minimize a reconstruction loss at all; their weights are updated exclusively through backpropagation from a supervised objective such as cross-entropy, making them inherently discriminative units. The internal generative model within a GMU serves only as a mechanism for producing its activation signal, and the inferred latent variables (θ) are discarded after the forward pass. This distinction highlights a fundamental difference in purpose: GCMs are designed for unsupervised generative modeling, where the goal is faithful reconstruction of inputs, while GMUs are designed for supervised function fitting, where the goal is predictive performance. Moreover, GCMs rely on more computationally expensive variational inference schemes (such as ELBO optimization) to handle both appearance and geometric variability, whereas GMUs work with a one-shot solution that emerges from a MAP approximation under a CL-SCM sampling framework, allowing them to integrate efficiently into standard supervised pipelines. Thus, while both models have latent linear structures that do sample-wise inference, their optimization paradigms and roles in learning are fundamentally different.

5.1 Can Kolmogorov Arnold Networks (KANs) efficiently re-parameterize GMUs?

In this section we examine whether Kolmogorov–Arnold Networks (KANsLiu et al. [2025]) can serve as efficient re-parameterizations of GMUs. Although KANs and GMUs are motivated by entirely different principles; KANs by universal function approximation, GMUs by Bayes optimality to switching SCM based data generation; we ask whether KANs can nevertheless parameterize GMUs efficiently. In theory, KANs can represent *any* continuous function; here we outline the construction that would allow such equivalence and then test whether the resulting KAN design is feasible in practice. This sets the stage for evaluating if the universality promised by the Kolmogorov–Arnold representation can also reproduce the function of GMU architectures.

5.1.1 Kolmogorov–Arnold representation and canonical KANs

The Kolmogorov–Arnold theorem (KAT) states that any continuous function $f : [0, 1]^d \to \mathbb{R}$ can be represented as

$$f(x_1, \ldots, x_d) \;=\; \sum_{q=1}^{2d+1} \Phi_q \left(\sum_{j=1}^{d} \Psi_{qj}(x_j) \right), \tag{12}$$

where each inner function Ψ_{qj} is univariate in its coordinate x_j and each outer function Φ_q is univariate. Canonical KAN layers operationalize this template by replacing linear edge weights with learnable univariate functions (often spline parameterizations) and aggregating additively before a single outer nonlinearity:

$$h_i(x) \;=\; g_i \left(\sum_{j=1}^{d} f_{ij}(x_j) + b_i \right), \tag{13}$$

with $f_{ij} : \mathbb{R} \to \mathbb{R}$ univariate (e.g., spline bases) and g_i a simple scalar activation (identity or smooth). This induces an axis-aligned inductive bias and per-coordinate interpretability.

5.1.2 GMU structure and the sum-of-squares exponential form

The linear GMU unit computes an activation from a projector-induced residual norm:

$$\mathrm{gmu}(X) \;=\; \phi\left(\frac{1}{d}\left\|(I - P_W)\left(\frac{X-b}{\eta}\right)\right\|^2\right), \qquad P_W \;=\; W^\top (W^\top)^\dagger, \tag{14}$$

where $\phi(t) = \exp(-t)$ in our default choice. Writing $Z = (X-b)/\eta$ and letting $\{v_r\}_{r=1}^d$ be the row vectors of $I - P_W$.

$$\|(I - P_W)Z\|^2 \;=\; \sum_{r=1}^{d} (v_r^\top Z)^2, \tag{15}$$

so that

$$\mathrm{gmu}(X) \;=\; \exp\left(-\frac{1}{d}\sum_{r=1}^{d}(v_r^\top Z)^2\right). \tag{16}$$

This form is an exponential of a *sum of squared linear forms*, where the mixing directions $\{v_r\}$ are geometrically tied to the learned subspace via P_W.

5.1.3 Why a single canonical KAN layer cannot reproduce GMUs

A canonical KAN layer enforces a separable preactivation $\sum_j f_j(X_j)$ with one outer nonlinearity $g(\cdot)$. While setting $g(t) = t^2$ can realize *one* squared linear form $(\sum_j u_j X_j)^2$ by choosing $f_j(X_j) = u_j X_j$, a single layer cannot, under the univariate-per-edge constraint, produce a *sum of multiple squared linear forms* $\sum_{r=1}^{d}\left(\sum_j v_{rj} Z_j\right)^2$ and then apply a single exponential to that aggregate. The inner sum in canonical KANs is scalar; producing multiple squared linear forms requires multiple distinct preactivations prior to aggregation, which exceeds the expressivity of one KAN layer with strictly univariate edges.

5.1.4 Two layer KAN equivalence in theory but not in practice

For m GMU outputs with input dimension d (zero bias, unit scale), each GMU i computes

$$y_i(X) = \exp\Big(-\tfrac{1}{d}\sum_{r=1}^{d}(v_r^{(i)\top} Z^{(i)})^2\Big), \qquad Z^{(i)} = \tfrac{X-b_i}{\eta_i}.$$

A minimal two-layer KAN replication instantiates, for every output i and residual direction r,

$$\text{Layer 1:}\quad s_r^{(i)}(X) = \sum_{j=1}^{d} v_{rj}^{(i)} Z_j^{(i)}, \qquad h_r^{(i)}(X) = \big(s_r^{(i)}(X)\big)^2,$$

and then performs grouped aggregation in Layer 2. In the worst case $k_i = 0$, the total number of Layer-1 units is $N_{\mathrm{L1}} = m \cdot d$, and the grouping is explicit:

$$\text{Layer 2:}\quad S_i(X) = \sum_{r=(i-1)d+1}^{id} h_r(X), \qquad y_i(X) = \exp\Big(-\tfrac{1}{d}S_i(X)\Big), \quad i = 1, \ldots, m.$$

This construction is exact in theory, but extremely impractical in practice:

- **Strict Activation functions.** The construction requires Layer 1 to apply a squaring nonlinearity and Layer 2 to apply a single exponential. These choices are fixed and leave no room for the spline-based activation learning that defines canonical KANs.
- **Strict Connectivity.** Layer 2 must partition the $m \cdot d$ Layer-1 units into m disjoint groups of size d and sum only the units in each group to form the corresponding GMU output. This strict wiring is non-native to canonical KANs.
- **Unit and parameter blowup.** Example: $m = d = 100 \Rightarrow N_{\mathrm{L1}} = 10{,}000$ Layer-1 units. Each unit carries a length-d mixing vector, yielding $\approx m\, d^2 = 1{,}000{,}000$ parameters.
- **Non-native constraints.** Exact equivalence requires the per-output mixing vectors $\{v_r^{(i)}\}$ to originate from a Cholesky-decomposable matrix so that the squared linear forms correspond to a valid residual norm. Canonical B-spline KANs do not enforce such structured parameterizations, making equivalence fragile under standard training.

In Summary: While a two-layer KAN can replicate the GMU parameterization on paper (as it can replicate *any* continuous function by design), it demands highly specific connectivity, non-standard activation choices, and massive overparameterization. The only way to avoid overparameterization is via structured external constraints such as Cholesky factorization which are not native to KANs, and even in that case the other issues still remain. Canonical KANs therefore cannot reproduce GMUs in practice without heavy specification and computational inefficiency.

5.2 Comparing with Capsule Networks

Capsule networks and GMUs are motivated by fundamentally different principles. Capsules emphasize *self-organization via feedback and routing*: their dynamic routing mechanism iteratively adjusts coupling coefficients between capsules to capture part–whole relationships in data [Hinton et al., 2018, Sabour et al., 2017]. GMUs, by contrast, have no such feedback component. They are designed as generative computational units that measure similarity through k-subspace distances, providing robust supervised function fitting with explicit theoretical guarantees.

Importantly, capsule routing typically computes similarity between a lower-level capsule u_i and a higher-level capsule v_j via a dot product or cosine similarity:

$$c_{ij} \propto u_i^\top v_j, \tag{17}$$

where c_{ij} denotes the coupling coefficient. Recent work has explored distance-based routing strategies, such as K-Means routing for text classification [Ren and Lu, 2022] and for complex image analysis [Kwabena et al., 2022]. These approaches replace dot-product similarity with Euclidean-type distances in the clustering step, showing that capsule routing can be generalized beyond inner products:

$$c_{ij}^{\text{Euc}} = \phi\big(-\|u_i - v_j\|^2\big)\,. \tag{18}$$

In this broader context, one can analogously define a GMU-based capsule similarity using the k-subspace distance:

$$\text{sim}_{k,W}(u_i, v_j) = \phi\left(\frac{1}{d}\min_{\theta\in\mathbb{R}^k}\left\|u_i - v_j - W^\top\theta\right\|^2\right), \tag{19}$$

where $W \in \mathbb{R}^{k\times d}$ parameterizes a learnable generative subspace, b is a bias term, and $\phi(\cdot)$ is an activation function. This generalizes the Euclidean case ($k = 0$) and positions GMUs as a principled alternative similarity measure for capsule routing.

As we have shown in this paper, for linear GMUs the k-subspace distance admits a closed-form solution (Eq. 2) and can be efficiently parallelized using tensorized computation with Einstein summations, enabling real-time inference. GMUs can actually be integrated into capsule frameworks in two complementary ways: (i) by replacing the similarity score with a generative subspace projection, or (ii) by substituting the reconstruction MLP layers within capsule networks with GMU layers.

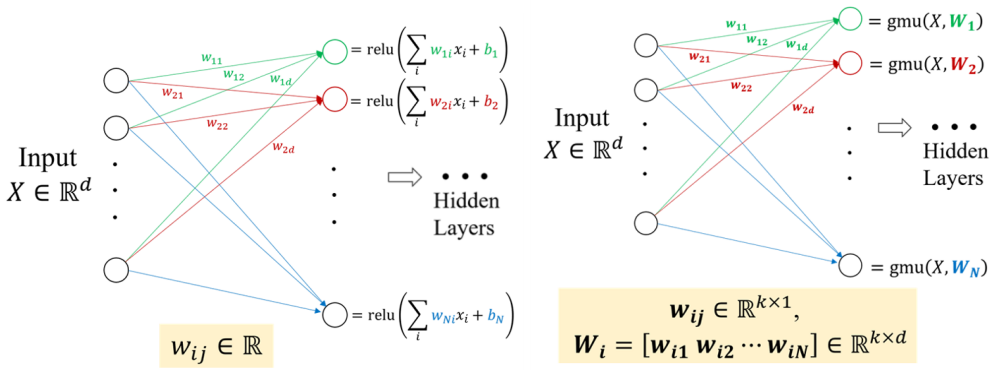
Thus fundamentally, capsule networks answer a different question than GMUs: they are about *feedback and routing*, while GMUs are about *generative similarity and robust activation*. Importantly, these principles are not mutually exclusive. As we show here, capsule routing can occur under GMU-based similarity computations, which represents an important direction of future work with GMUs.

6 Details on GMU Function and Integration

6.1 Integrating GMUs into architectures

GMU-MLPs As shown in Figure 10, GMU integration into an MLP is direct, and only involves replacing the computational units with GMU units. Like perceptron units, linear GMU units have their own weights and biases, which are visualized in Figure 10. The network is trained via standard backpropagation, which in our case is handled by Pytorch's autograd framework.

GMU-CNNs When replacing a convolution layer with GMUs, we replace each linear convolutional operation with a convolutional GMU. The operation of a convolutional GMU is shown in Figure 11. X_{ij}^a represents the input patch of size $a \times a$, which is the input to the GMU, the output of which is simply the GMU's output $\text{gmu}(X_{ij}^a, \boldsymbol{W})$, where $\boldsymbol{W}$ are its internal weights. As the operation is a convolution, this function is repeated across all patches centered at all (i, j) pairs in the input image, yielding an output image of the same size (assuming padding is used). Mathematically, we can represent the convolutional GMU operation as $\text{gmu}(., \boldsymbol{W}) * X$.



(a) MLP **(b)** GMU-MLP

Figure 10: MLPs versus GMU-MLPs: Demonstration of how GMUs are integrated into architectures. Here, we replace the first network layer with GMU computational units.

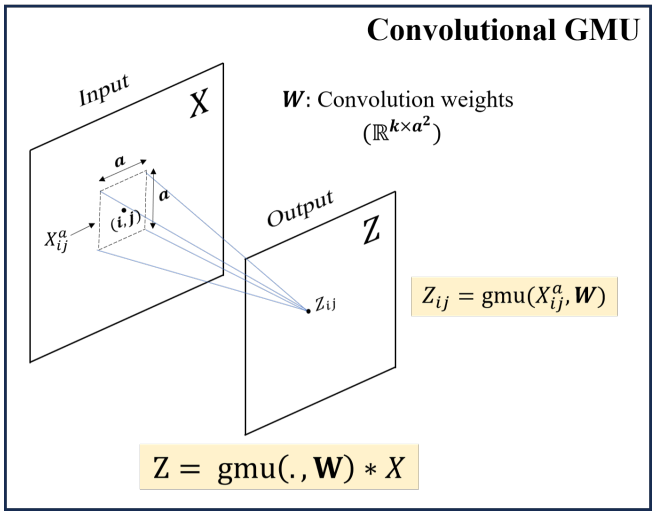


Figure 11: Diagram showing the operations within a convolutional GMU. Here, the convolutional weights W are applied to the input X yielding the output Z. For the purposes of this figure, we assume that the convolution is appropriately padded such that X and Z are of the same size.

6.2 Linear GMUs: Closed Form Expression

Here, we summarize the least square based approach that yields the closed-form expression of a linear GMU's output in (2). The minimization in (1) can be represented as

$$\min_{\theta\in\mathbb{R}^k} \left\| \frac{X-b}{\eta} - W^T\theta \right\| \tag{20}$$

Note that as $X \in \mathbb{R}^d$ (consider as a column vector), the above minimization represents the least squares solution to a set of d equations, one for each dimension of X. The solution to it is as follows

$$\underset{\theta\in\mathbb{R}^k}{\text{argmin}} \left\| \frac{X-b}{\eta} - W^T\theta \right\| = (WW^T)^{-1}W\left(\frac{X-b}{\eta_X}\right) \tag{21}$$

Note that here $\frac{X-b}{\eta_X}$ denotes a matrix of size $d \times 1$. With this, each GMU unit's output in (1) can be represented as

$$\phi\left(\frac{1}{\sigma d}\left\| \frac{X-b}{\eta} - W^T(WW^T)^{-1}W\left(\frac{X-b}{\eta}\right)\right\|^2\right) \tag{22}$$

Note that the above is the output of a single GMU, which only depends on the input to the unit X, the weights and biases W, b, and other parameters such as the normalizer η, the scaling parameter σ and the dimensionality of input d.

7 Additional Empirical Details

7.1 Synthetic Experiments

Please refer to Figure 2(b) for the detailed jargon in defining GMU types.

Sparse Linear Structure Prediction: For the GMU(k) variants, we used a unit without normalization and bias. $\phi(z) = -\log z$ (to counter-act the softmax function that follows) and $W \in \mathbb{R}^{k \times d}$. For the out-of-distribution (ood) columns, we set the training $\theta_{l_y(i)}$ ranges to either between $Unif(0, 0.5)$ or $Unif(0.5, 1)$ chosen at random. For the test data, we change the range for each $\theta_{l_y(i)}$ in such a manner that if its training configuration was $Unif(0, 0.5)$ it is set to $Unif(0.5, 1)$ and vice-versa. This ensures that at test-time the network sees values of the latent generating variables which it hasn't seen before.

Dynamic Tree Structure Prediction: To generate the random trees, first, we sample the root node from a uniform distribution over all dimensions, and then sample left and right nodes uniformly from the rest of the dimensions. The process is then repeated for the children of all leaf nodes. Once the children of all leaf nodes at a certain depth have been sampled, the process is repeated for the children of the subsequent depth of the tree.

For the GMU(k)-MLP variants, for the GMU units, we used units with normalization $\eta = \sigma(X)$ and bias. $\phi(z) = \sqrt{1 - z}$ and $W = [\mathbb{R}^{k-1 \times d}; J_{1,d}]$. To have a fair comparison, each datapoint was also normalized using zero-mean and unit variance for the MLP variants.

Sparse Neural Structure Sampling: For the GMU units in the GMU(k)-MLP variants, we used units without normalization, but non-zero bias. $\phi(z) = -\log z$ and $W = \mathbb{R}^{k \times d}$.

Polynomial Boundary Sampling: For both the GMU(k)-MLP and the GMU(k) variants, for the GMU units, we used units without normalization, but non-zero bias. $\phi(z) = e^{-z}$ and $W = \mathbb{R}^{k \times d}$.

Conditional Gaussian Sampling: For the GMU units in the GMU(k)-MLP variants, we used units without normalization, but non-zero bias. $\phi(z) = e^{-z}$ and $W = \mathbb{R}^{k \times d}$.

7.2 Tabular Experiments

In what follows, we abbreviate the GMU-Resnet-P and GMU-Resnet-S configurations as GMU-P and GMU-S respectively. In addition to the GMU results shown in the main paper (GMU-P), we test another configuration of GMUs called GMU-S, that has more parameters. Using the jargon in Figure 2(b), the details are as follows.

GMU-P: We choose non-zero bias b and the normalization factor $\eta = \sqrt{\sum_{i=1}^{d} x_i^2/d}$. The activation function is set to $\phi(z) = e^{-z}$. The weights are $W = [w, w^2, ..w^k], w : \mathbb{R}^d$.

GMU-S: We choose non-zero bias b and the normalization factor $\eta = \sqrt{\sum_{i=1}^{d} x_i^2/d}$. The activation function is set to $\phi(z) = e^{-z}$. The weights are $W = \mathbb{R}^{k \times d}$

The networks were trained in the same manner as in Kadra et al. [2021], using weighted cross-entropy loss, and for evaluation we also report the balanced accuracy, same as them. We compare GMU-Resnet-P/S with Resnet-512-[1,1]. We set the same hyperparameters for all experiments, and there is no additional hyperparameter optimization. Note that the other approaches' results are after extensive hyperparameter optimization using BOHB [Falkner et al., 2018]. Also note that Kadra et al. [2021] uses a different Shaped Resnet architecture and therefore we don't directly compare with their MLP results, and we find in some datasets our Resnet performs significantly better than theirs and vice-versa. Furthermore the MLP+C approach in Kadra et al. [2021] employs an extensive suite of regularization approaches, including data augmentation, so we don't include their results for this study. Similarly, the MLP-Dropout in Kadra et al. [2021] also uses hyperparameter optimization for the dropout levels and locations for each dataset.

We add a single dropout layer (of 0.2) at the penultimate layer for all Resnet-512-[1,1] variants, as we found it led to more stable training overall. Apart from this, there is no regularization or data

Dataset	#Ins./#Feat.	XGB.	ASK-G.	TabN.	AutoGL. S	Resnet-530 (1.2M)	GMU-P (1.159M)	GMU-S (1.195M)
credit-g	1000 / 21	0.6893	0.7119	0.6119	0.6964	0.6972	0.7071	**0.7310**
anneal	898/39	0.8542	**0.9000**	0.8425	0.8000	0.8546	0.8596	0.8396
kr-vs-kp	3196/37	**0.9985**	**0.9985**	0.9325	0.9969	0.9969	0.9954	0.9969
mfeat-factors	2000/217	0.9800	0.9750	0.9725	0.9800	**0.9850**	0.9775	0.9725
vehicle	846/19	0.7497	0.8017	0.7965	0.8379	0.8446	**0.8666**	0.8669
kc1	2109/22	0.6685	0.6335	0.5252	0.6727	0.7104	0.7090	**0.7356**
phoneme	5404/6	0.8797	0.8834	0.8682	0.8394	**0.8959**	0.8930	0.8911
cnae-9	1080/857	**0.9491**	0.9352	0.8935	0.9259	0.9398	0.9537	0.9352
blood	748/5	0.6228	0.6499	0.6433	0.6725	0.6304	**0.6813**	0.6411
Australian	690/15	**0.8972**	0.8859	0.8528	0.8825	0.8479	0.8743	0.8743
car	1728/7	0.9238	**1.0000**	0.9870	0.9968	**1.0000**	**1.0000**	**1.0000**
segment	2310/20	**0.9372**	0.9307	0.9178	0.9199	0.9254	0.9221	0.9091
jasmine	2984/145	**0.8055**	0.7888	0.7669	0.8005	0.7438	0.7604	0.7336
sylvine	5124/21	0.9307	0.9336	0.8360	**0.9375**	0.9250	0.9229	0.9288
adult	48842/15	0.7982	0.7983	0.7716	**0.8056**	0.7652	0.7812	0.7793
nomao	34465/119	0.9687	**0.9722**	0.9543	0.9642	0.9582	0.9604	0.9562
bank	45211/17	0.7266	0.7228	0.7064	**0.7948**	0.7261	0.7370	0.7598
jungle_chess	44819/7	0.8733	0.8307	0.7343	0.9302	0.9686	**0.9868**	0.9788
volkert	58310/181	0.6417	0.6343	0.5941	**0.7020**	0.6766	0.6667	0.6814
helena	65196/28	0.2199	0.2114	0.1903	**0.2712**	0.2213	0.2226	0.2137
connect-4	67557/43	0.7237	0.7265	0.7205	**0.7562**	0.7369	0.7495	0.7495
higgs	98050/29	0.7294	0.7293	0.7204	**0.7380**	0.6684	0.6853	0.6783
numerai28.6	96320/22	**0.5236**	**0.5242**	0.5160	0.5171	0.5066	0.5075	0.5101
walking-activity	149332/5	0.6162	0.6276	0.5680	0.6080	0.6203	**0.6297**	**0.6294**
ldpa	164860/8	**0.9901**	0.6895	0.5482	0.5302	0.6991	0.6816	0.6841
aloi	108000/129	0.9534	0.1353	0.9359	**0.9742**	0.9617	0.9646	0.9659
skin-seg	245057/4	0.9997	0.9997	0.9996	0.9997	0.9997	0.9997	0.9997
Wins/losses/ties	**GMU-P vs**	14/11/2	13/12/2	23/3/1	12/13/2	15/6/6	–	
Wilcoxon p	**GMU-P vs**	0.367	0.427	1.2e-04	0.819	0.012	–	
Wins/losses/ties	**GMU-S vs**	13/11/3	10/12/5	18/6/3	10/15/2	15/8/4	–	
Wilcoxon p	**GMU-S vs**	0.558	0.548	1.2e-03	0.903	0.12	–	

Table 20: Performance Comparison of Various Models on Different Datasets. Parameter counts are given in brackets in the top-row. Please note that we scaled the original Resnet-512 to Resnet-530 to match parameter counts with the GMU-Resnet models.

augmentation performed, and networks are trained in the same manner for all datasets. The average parameter count for each approach is given below the network names in Table 20.

The categorical variables within the data were one-hot encoded, and the other variables were normalized to the range (0,1), with the statistics computed only from the training split. The training-test splits are exactly the same as in Kadra et al. [2021], which is an 80-20 split, with the same random seed values set by them. This was made possible by their code, and the fact that each dataset corresponded to a specific task as numbered in Table 9 in Kadra et al. [2021].

7.3 Vision Experiments

GMU Details: Using the jargon in Figure 2(b), the details are as follows. We set bias to zero $b = 0$ and the normalization factor $\eta = \sigma_X$. The activation function is set to $\phi(z) = e^{-z}$. The weights are $W = [\mathbb{R}^{k-1 \times d}; J_{1,d}]$, same as what was used in the dynamic tree structure prediction experiment.

Architectures: We train a four-layer CNN for MNIST and Fashion-MNIST, with the architecture 64C(3,Padding=1)-2MP-128C(3,Padding=1)-2MP-128C(3,Padding=1)-2MP(Padding=1)-128C(3,Padding=1)-4MP-FC128-FC10, where C(k,Padding=j) denotes $k \times k$ convolutional layers with a padding of j in either direction, MP denotes max pooling layers, and FC denotes fully connected layers. The first convolutional layer is replaced with convolutional GMUs of order 3. For CIFAR-10, we use VGG-16 as the base network, replacing the first convolutional layer with convolutional GMU-3 units while keeping the same number of output nodes. Similarly, for smallNORB, we resize images to 64x64 and train a four-layer CNN: 64C(5,Padding=0)-2MP-128C(5,Padding=0)-2MP-128C(3,Padding=0)-2MP-128C(3,Padding=0)-4MP-FC10, with the first convolutional layer replaced with convolutional GMUs. Note that for smallNORB, we test GMUs of varying orders (1 and 3), the results of which are reported in Table 2 of the main paper.

For SVHN, we train a ResNet-18 architecture, replacing the first convolutional layer with convolutional GMU units of order 3 and kernel size 7x7, maintaining the original receptive field. We applied standard data augmentation (for all baselines) involving rotations and translations. Note that no data augmentation was applied to any of the other datasets.

7.4 Optimization

We summarize the optimization details for the experiments on the Tabular and Vision datasets. We use Adam optimizer for all experiments. All networks are trained for 200-300 epochs, with an

Dataset	Dimension	GMU-Resnet Time (ms)	Resnet Time (ms)
credit-g	63	0.0181	0.0036
anneal	72	0.0171	0.0041
kr-vs-kp	74	0.0188	0.0045
mfeat-factors	216	0.0149	0.0036
vehicle	18	0.0183	0.0044
kc1	21	0.0214	0.0051
phoneme	5	0.0167	0.0040
cnae-9	856	0.0398	0.0067
blood transfusion	4	0.0205	0.0048
australian	42	0.0221	0.0054
car	21	0.0177	0.0042
segment	16	0.0197	0.0047
jasmine	280	0.0159	0.0037
sylvine	20	0.0182	0.0043
adult	105	0.0154	0.0036
nomao	174	0.0156	0.0037
bank marketing	51	0.0156	0.0037
jungle chess	6	0.0153	0.0036
volkert	180	0.0155	0.0037
helena	27	0.0155	0.0037
connect-4	126	0.0155	0.0036
higgs	28	0.0154	0.0037
numerai28.6	21	0.0154	0.0037
walking activity	4	0.0153	0.0037
ldpa	14	0.0153	0.0036
aloi	128	0.0185	0.0036
skin segmentation	3	0.0153	0.0036
Average	-	0.0179	0.0041

Table 21: Per-sample Inference Times in tabular datasets (in milliseconds)

Dataset	GMU-1	GMU-3	GMU-5	GMU-8	CNN
MNIST	-	0.0183	-	-	0.0088
Fashion-MNIST	-	0.0193	-	-	0.0045
CIFAR-10	-	0.0232	0.0232	0.0244	0.0089
Small-NORB	0.0440	0.0456	-	-	0.0111
ImageNet (ResNet-50)	14.263	13.085	13.728	14.395	8.841

Table 22: Per-sample inference times (milliseconds) in vision datasets.

exponential learning rate scheduler $lr = init_{lr} \times 0.5^{epoch/50}$, where $init_{lr}$ is the initial learning rate. For the GMU runs, we found that picking the network with the lowest training loss yielded more stable performance, so we stick with this approach for all experiments in the Tabular and Vision datasets. For the MLP/Resnet runs, we didn't see any such improvement, so we use the standard approach of training for a fixed number of epochs. A reason why this would be the case is that GMU-MLPs'/GMU-CNNs' losses are slightly more fluctuating in nature than MLP/Resnet's losses, so if the training stopped at an epoch which had higher fluctuations of training loss, that could be a detriment to performance.

7.5 Computation Times

We document the per-sample inference times of GMU-based architectures and compare to the standard counterparts in each case. First, we report the per-sample inference times (in milliseconds) in all our tabular datasets tested in Table 21. Overall, although the standard counterparts are faster, we find that GMUs only take about 0.0017 milliseconds (ms) on average for per-sample inference, retaining their feasibility for real-time inference and fast training.

We also report the per-sample inference times for the GMU-CNNs tested in MNIST, Fashion-MNIST, CIFAR-10 in Table 22. To see the feasibility of scaling GMUs to Imagenet-level datasets, we also include per-sample inference times of a pre-trained ResNet-50 on 224×224×3 sized Imagenet inputs. Following the main paper, the first layer of the original networks was replaced by convolutional GMUs. We find that the compute times stay insignificant for all datasets. For Imagenet, the compute times for GMU-ResNets are larger (>10 ms), but still feasible for real-time inference, and comparable to the standard ResNet.

FLOP Analysis. To complement the timing results, we also analyze the theoretical floating-point operations (FLOPs) per GMU unit. The approximate cost is

$$\text{FLOPs}_{\text{gmu}} \;\approx\; 2k^2d + 4kd + \tfrac{1}{3}k^3 + 2k^2 + 5d,$$

where d is the input dimensionality and k is the number of slices. The dominant term is $2k^2d$, arising from the $k \times k$ pseudo-inverse. In practice, since we always keep $k \leq 8$, the cubic term never dominates. For comparison, a standard perceptron unit requires $2d$ FLOPs. Thus, GMUs incur a modest multiplicative overhead in FLOPs relative to perceptrons, but remain computationally tractable even in high dimensions, consistent with the empirical inference times reported above.

Dataset	Model	Baseline Params	GMU Variant	GMU Params	% Change
MNIST / Fashion-MNIST	Shallow CNN	0.39M	GMU-3	0.39M	+0.3%
CIFAR-10	VGG16	14.7M	GMU-3 / GMU-8	14.7M / 14.7M	+0.02% / +0.08%
SVHN	ResNet18	11.2M	GMU-3	11.2M	+0.17%
smallNORB	Shallow CNN	0.52M	GMU-1 / GMU-3	0.52M / 0.52M	0% / +0.6%
Tabular (Average)	ResNet-530	1.20M	GMU-S / GMU-P	1.19M / 1.16M	-0.8% / –3.4%

Table 23: Baseline vs GMU parameter counts across datasets and models. GMUs are applied only in the first layer

7.6 Parameter Count Comparisons

We report the parameter counts of baseline architectures and their GMU-augmented counterparts across all our real dataset experiments. GMUs are only substituted in the first convolutional or dense layer, so the overall increase in parameters remains negligible. For tabular experiments, GMU-P and GMU-S were configured to keep parameter counts close to the ResNet-530 baseline.

As seen in Table 23, the parameter growth from GMU substitution is insignificant relative to the total model size. Even in deeper architectures such as VGG16 and ResNet18, the increase is less than one percent, while in smaller CNNs the absolute growth remains on the order of only a few thousand parameters. Note that for tabular experiments, the baseline Resnet was scaled to 530 hidden units for fair comparison, after which the parameter counts are similar, with GMU-P having slightly lower counts than the baseline.

8 Proofs of Theoretical Results

We outline the proofs of all theoretical results in the main paper in this section.

Proposition 3.2. *Let the random variables (X, Y) follow the ECI sampling process, and let $S = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ be i.i.d. samples from this distribution. Consider a single-layer neural network with computational units $\mathrm{pu}_i(X, \mathbf{W}, \mathbf{b}) \coloneqq \sum_{j=1}^{d} W_{ij} x_j + b_i, \quad i \in \{1, \ldots, M\}$, followed by a softmax layer producing output $P_{\mathrm{soft}}(Y|X, \mathbf{W}, \mathbf{b})$. Let $\mathcal{L}_{\mathrm{CE}}$ denote the empirical cross-entropy loss and assume it is uniformly bounded over the support of $P(Y|X)$. Then, for every $(\mathbf{W}^*, \mathbf{b}^*) \in \arg\min_{\mathbf{W},\mathbf{b}} \mathcal{L}_{\mathrm{CE}}(S)$, $\lim_{n\to\infty} \mathrm{KL}\left(P(Y|X) \,\|\, P_{\mathrm{soft}}(Y|X, \mathbf{W}^*, \mathbf{b}^*)\right) = 0$ almost surely.*

Proof. Using Bayes' rule,

$$P(Y|X) = \frac{P(Y)\prod_{j=1}^{d} P(x_j|Y)}{\sum_{i=1}^{M} P(Y=i)\prod_{j=1}^{d} P(x_j|Y=i)},$$

and substituting the ECI form $P(x_j|Y=i) = \frac{1}{Z_{ij}} \exp(\alpha_{ij} x_j)$ gives

$$P(Y|X) = \frac{P(Y)\prod_{j=1}^{d} \frac{1}{Z_{ij}} \exp(\alpha_{ij} x_j)}{\sum_{i=1}^{M} P(Y=i)\prod_{j=1}^{d} \frac{1}{Z_{ij}} \exp(\alpha_{ij} x_j)}.$$

This is of the same functional form as the softmax output, since the neural network logits are defined by

$$\mathrm{pu}_i(X, \mathbf{W}, \mathbf{b}) = \sum_{j=1}^{d} W_{ij} x_j + b_i,$$

and by setting $W_{ij} = \alpha_{ij}$ and $b_i = \log P(Y) - \sum_{j=1}^{d} \log Z_{ij}$ we have $\log P(Y|X) \propto \mathrm{pu}_i(X, \mathbf{W}, \mathbf{b})$, so that

$$P_{\mathrm{soft}}(Y|X, \mathbf{W}^*, \mathbf{b}^*, S) = \frac{\exp(\mathrm{pu}_i(X))}{\sum_{i=1}^{M} \exp(\mathrm{pu}_i(X))}.$$

Since the cross-entropy loss $\mathcal{L} = -\mathbb{E}_{P(Y|X)}[\log P_{\text{soft}}(Y \mid X)]$ differs from the KL divergence only by the entropy of $P(Y \mid X)$, minimizing cross-entropy is equivalent to minimizing $D_{\text{KL}}(P(Y \mid X) \,\|\, P_{\text{soft}}(Y \mid X))$. Because both $P(Y \mid X)$ and $P_{\text{soft}}(Y \mid X)$ share the same functional form under the ECI assumption, the minimum is attained when this KL divergence is zero. By the strong law of large numbers, since the samples are i.i.d. and the loss is uniformly bounded, the empirical cross-entropy converges almost surely to its population expectation as $n \to \infty$. Thus, $\lim_{n\to\infty} D_{\text{KL}}(P(Y \mid X) \,\|\, P_{\text{soft}}(Y \mid X, \mathbf{W}^*, \mathbf{b}^*)) = 0$ almost surely.

□

Proposition 3.4. *Let the random variables (X, Y) follow the CL-SCM sampling process, and let $S = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ be i.i.d. samples from this distribution. Consider a single-layer neural network with computational units $g_i(X, \mathbf{W}, \sigma) \coloneqq b_i - \frac{1}{\sigma^2} \min_\theta \mathbb{E}\left[\|f(\theta, W_i) - X\|^2\right], \quad i \in \{1, \ldots, M\}$ for $i \in \{1, \ldots, M\}$, followed by a softmax layer producing output $P_{\text{soft}}(Y \mid X, \mathbf{W}, \sigma)$. b_i are bias terms. Let $\mathcal{L}_{\text{CE}}$ denote the empirical cross-entropy loss and assume it is uniformly bounded over the support of $P(Y \mid X)$. Then, for every $(\mathbf{W}^*, \sigma^*) \in \arg\min_{\mathbf{W},\sigma} \mathcal{L}_{\text{CE}}(S)$, we have $\lim_{n\to\infty} \text{KL}(P(Y \mid X) \,\|\, P_{\text{soft}}(Y \mid X, \mathbf{W}^*, \sigma^*)) = 0$ almost surely.*

Proof. Under the CL–SCM sampling process, the generative model is

$$X = f(\theta, W_i) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2 I), \quad \theta \sim \mathcal{U}(\Omega)$$

As we're assuming a MAP estimate of posteriors, we have

$$P(X \mid Y = i) = \max_\theta P(X \mid Y = i, \theta) P(\theta)$$

Since $P(\theta)$ is uniform, this reduces to maximizing the likelihood:

$$P(X \mid Y = i) = \frac{1}{\text{Vol}(\Omega)(\sigma\sqrt{2\pi})^d} \exp\Big(-\frac{1}{2\sigma^2} \min_\theta \|X - f(\theta, W_i)\|^2\Big)$$

By Bayes' rule:

$$P(Y = i \mid X) = \frac{P(Y = i) P(X \mid Y = i)}{\sum_{j=1}^M P(Y = j) P(X \mid Y = j)}$$

Substituting the expression for $P(X \mid Y = i)$ and cancelling the shared constant terms $(\text{Vol}(\Omega)(\sigma\sqrt{2\pi})^d)$, we get:

$$P(Y = i \mid X) = \frac{\exp\big(\ln P(Y = i) - \frac{1}{2\sigma^2} \min_\theta \|X - f(\theta, W_i)\|^2\big)}{\sum_{j=1}^M \exp\big(\ln P(Y = j) - \ln \text{Vol}(\Omega_j) - \frac{1}{2\sigma^2} \min_\theta \|X - f(\theta, W_j)\|^2\big)}$$

By setting the bias terms $b_i = \ln P(Y = i)$, the computational units $g_i(X, \mathbf{W}, \sigma)$ exactly match the terms in the exponents above. Consequently, the single-layer neural network with softmax computes:

$$P_{\text{soft}}(Y = i \mid X, \mathbf{W}, \sigma) = \frac{\exp(g_i(X, \mathbf{W}, \sigma))}{\sum_{j=1}^M \exp(g_j(X, \mathbf{W}, \sigma))}$$

which shares the same functional form as $P(Y \mid X)$. Since minimizing the cross-entropy loss is equivalent to minimizing $D_{\text{KL}}(P(Y \mid X) \,\|\, P_{\text{soft}}(Y \mid X))$, the minimum is attained when this KL divergence is zero. By the strong law of large numbers, the empirical cross-entropy converges almost surely to its population expectation as $n \to \infty$. Thus, $\lim_{n\to\infty} D_{\text{KL}}(P(Y \mid X) \,\|\, P_{\text{soft}}(Y \mid X, \mathbf{W}^*, \sigma^*)) = 0$ almost surely.

□

Proposition 5.1. *(from Park and Sandberg [1991]) We are given a GMU-MLP, with GMU units in* (1) *specified as: $k = 0, \eta = 1$, $f(\theta, W)$ is such that $\exists W \, s.t. f(\theta, W) = 0$ and $\phi(z)$ is any integrable bounded function such that $\int \phi(x)dx \neq 0$. Then, this GMU-MLP can approximate any function $f \in L^p(\mathbb{R}^d)$.*

Proof. First, we note that the set of functions approximable by the GMU-MLP under the constraint $\exists W s.t. f(\theta, W) = C$ is a subset of the set of functions that the GMU-MLP can approximate without that constraint. Next, we note that when we set the weights of the GMU such that $f(\theta, W) = C$, setting $\sigma'^2 = \sigma^2 d$, and $\eta = 1$ yields an RBF unit as the bias b can be adjusted as $b' = b - C$ to yield the original RBF form. By Theorem 1 in Park and Sandberg [1991], it is known that RBF-based networks achieve universal approximation in $L^p(\mathbb{R}^d)$, thus proving the result. □

Theorem 5.4. *Let $E(X_1, X_2) = \|X_1 - X_2\|$, we can show that $\gamma_E(d+1) < \gamma_E(d)$.*

Proof. Since $X_1, X_2 \sim \mathcal{N}(0, I_d)$, we have $X_1 - X_2 \sim \mathcal{N}(0, 2I_d)$, so that $\|X_1 - X_2\| = \sqrt{2}\,\chi_d$, where χ_d is a random variable with the chi distribution with d degrees of freedom. Its mean is $\mathbb{E}[\chi_d] = \sqrt{2}\,\frac{\Gamma\left(\frac{d+1}{2}\right)}{\Gamma\left(\frac{d}{2}\right)}$ and its variance is $\mathrm{Var}(\chi_d) = d - \left(\sqrt{2}\,\frac{\Gamma\left(\frac{d+1}{2}\right)}{\Gamma\left(\frac{d}{2}\right)}\right)^2$. Therefore, the mean of $E(X_1, X_2) = \|X_1 - X_2\|$ is $\mu_E = \sqrt{2}\,\mathbb{E}[\chi_d] = 2\,\frac{\Gamma\left(\frac{d+1}{2}\right)}{\Gamma\left(\frac{d}{2}\right)}$ and its standard deviation is $\sigma_E = \sqrt{2\big(d - 2\,\frac{\Gamma\left(\frac{d+1}{2}\right)^2}{\Gamma\left(\frac{d}{2}\right)^2}\big)}$.

Defining the CV as $\gamma_E(d) = \sigma_E/\mu_E$, we obtain

$$\gamma_E(d)^2 = \frac{\frac{d}{2}\,\Gamma\left(\frac{d}{2}\right)^2}{\Gamma\left(\frac{d+1}{2}\right)^2} - 1.$$

Thus, showing that $\gamma_E(d) < \gamma_E(d-1)$ is equivalent to proving

$$\frac{\Gamma\left(\frac{d+1}{2}\right)\Gamma\left(\frac{d-1}{2}\right)}{\Gamma\left(\frac{d}{2}\right)^2} > \sqrt{\frac{d}{d-1}}, \tag{23}$$

The above follows from the result using Gurland's ratio in equation 1.1 of Tian and Yang [2021], where it states:

$$\frac{\Gamma(x)\Gamma(x+2u)}{\Gamma(x+u)^2} > 1 + \frac{u^2}{x} \tag{24}$$

Setting $x = (d-1)/2$ and $u = 1/2$, we get:

$$\frac{\Gamma\left(\frac{d+1}{2}\right)\Gamma\left(\frac{d-1}{2}\right)}{\Gamma\left(\frac{d}{2}\right)^2} > 1 + \frac{1}{2(d-1)} \tag{25}$$

Squaring the RHS, we get: $\left(1 + \frac{1}{2(d-1)}\right)^2 = 1 + \frac{1}{d-1} + \frac{1}{4(d-1)^2} > \frac{d}{d-1}$. Thus we have,

$$\frac{\Gamma\left(\frac{d+1}{2}\right)\Gamma\left(\frac{d-1}{2}\right)}{\Gamma\left(\frac{d}{2}\right)^2} > \sqrt{\frac{d}{d-1}}, \tag{26}$$

yielding the result.

□

Corollary 5.5. *We are given the k-subspace distance $S_{k,W}(X_1, X_2)$ from X_1 to X_2. With this, first, we note that $\gamma_{S_{0,W}}(d) = \gamma_E(d)$, where E denotes the Euclidean distance. Then, we have that $\gamma_{S_{k+1,W}}(d) > \gamma_{S_{k,W}}(d)$, and thus $\gamma_{S_{k,W}}(d) > \gamma_E(d)$.*

Proof. The proof directly follows by realizing that for a fixed k-subspace, the closest distance to a point is equivalent to the squared root of sum of square of $d-k$ dimensions $x_1, x_2, .., x_{d-k}$ in the Euclidean space, where each dimension $x_i \sim \mathcal{N}(0,1)$ as the original data is also distributed this way.

This holds simply because one can rotate the space to align its unit vectors with the orthogonal directions of the k-subspace, leaving only the other $d-k$ to have degrees of freedom.

With this, it directly follows that $\gamma_{S_{k,W}}(d) = \gamma_E(d-k) < \gamma_E(d-k-1) = \gamma_{S_{k+1,W}}(d)$. And it naturally follows that $\gamma_{S_{k,W}}(d) = \gamma_E(d-k) > \gamma_E(d)$. □

Proposition 5.6. *We consider the case where the RV X has low intrinsic dimensionality. Let $X \in \mathbb{R}^d$ be generated as: $X = \sum_{i=1}^{ID} a_i \boldsymbol{v_i}, \quad a_i \sim \mathcal{N}(0,1)$, where $\{\boldsymbol{v_i}\}_{i=1}^{ID}$ are orthonormal vectors in $\mathbb{R}^d$, and ID $< d$ is the intrinsic dimensionality of X. Let us set $W = [\boldsymbol{v_1}, .., \boldsymbol{v_k}]$ to be the first k orthonormal vectors. Then, we have that $\gamma_E(d) = \gamma_E(ID)$ and $\gamma_{S_{k,W}}(d) = \gamma_E(ID-k)$.*

Proof. It is trivial to show that $\gamma_E(d) = \gamma_E(\text{ID})$. The result directly follows from the fact that X lies in a ID dimensional linear subspace with all Gaussian distributed dimension components $\mathcal{N}(0,1)$. Thus, we have that $\sigma[S(X_1, X_2)] = \sigma_{X_1', X_2' \sim \mathcal{N}(0, I_d)}[S(X_1', X_2')]$, where X_1' and X_2' are the ID dimensional. Ssimilarly, it follows that $\mu[S(X_1, X_2)] = \mu_{X_1', X_2' \sim \mathcal{N}(0, I_d)}[S(X_1', X_2')]$. Thus, we obtain $\gamma_E(d) = \gamma_E(\text{ID})$.

Let $X_1 = \sum_{i=1}^{\text{ID}} a_i \boldsymbol{v_i}$ and $X_2 = \sum_{i=1}^{\text{ID}} a_i' \boldsymbol{v_i}$. As W is constructed using the first k components of the linear subspace that contains X, the k-subspace distance $S_{k,W}(X_1, X_2)$ can be expressed as:

$$S_{k,W}(X_1, X_2) = \min_{\beta} || \sum_{i=1}^{\text{ID}} a_i \boldsymbol{v_i} + \sum_{i=1}^{k} \beta_i \boldsymbol{v_i} - \sum_{i=1}^{\text{ID}} a_i' \boldsymbol{v_i} || \tag{27}$$

$$= \min_{\beta} \sqrt{\sum_{i=1}^{k} (a_i + \beta_i - a_i')^2 + \sum_{i=k+1}^{\text{ID}} (a_i - a_i')^2} \tag{28}$$

$$= \sqrt{\sum_{i=k+1}^{\text{ID}} (a_i - a_i')^2} \tag{29}$$

where the minimum is reached when $a_i + \beta_i - a_i' = 0$ for all $i = \{1, 2, .., k\}$. Subsequently we note that the resulting distance is simply the distance between two ID $-\, k$ dimensional vectors, the entries of which are generated via the Gaussian distribution $\mathcal{N}(0,1)$. Thus, the k-subspace distance here is equivalent to taking a distance in a lower dimensional space of dimension ID $-\, k$, the CV of which is $\gamma_E(\text{ID} - k)$. This proves the second part of the result. □

Theorem 5.8. *Consider an input $X \sim \mathcal{U}[-0.5, 0.5]^d$. We consider a k-order GMU be defined with zero bias, no normalization, and weight axes aligned with k of the d dimensions. The balanced classification error ϵ of an RBF network with N units of radius R attempting to emulate the decision boundary of this GMU is bounded by:*

$$\epsilon \lesssim \frac{1}{2} \left[1 - \left(1 - \frac{1}{R^2 (N V_k(1))^{2/k}} \right)^{\frac{d-k}{2}} \right] \tag{30}$$

where $V_k(1)$ is the volume of a unit k-sphere. To maintain a constant error $\epsilon < 0.5$ as dimensionality d increases, the number of RBF units N must scale as $N = O(d^{k/2})$.

Proof of Theorem 5.8. **The Geometric Objective:** Emulating a k-order GMU is equivalent to covering a d-dimensional hypercylinder $\mathcal{C}$, defined by a k-dimensional cross-section of radius R and length 1 in the remaining $(d-k)$ dimensions (due to the unit cube support $[-0.5, 0.5]^d$). Let $\mathcal{L}$ be the k-dimensional linear subspace aligned with the weight axes. We assume RBF centers $\{C_i\}_{i=1}^N$ are uniformly distributed in the k-subspace $\mathcal{L} \cap [-0.5, 0.5]^d$ according to a Poisson Point Process (PPP) with intensity $\lambda = N / \text{Vol}_k(\text{Unit Cube}) = N$. The balanced error ϵ will depend on the hypercylinder volume not covered by the union of these N spheres $\mathcal{U} = \bigcup_{i=1}^N \mathcal{B}_d(C_i, R)$.

1. Coverage Probability For any point $x \in \mathcal{C}$ at distance $h = \text{dist}(x, \mathcal{L})$ from the subspace, let $P(h)$ be the probability that x lies within $\mathcal{U}$. This is equivalent to finding at least one RBF center within a search radius $\rho_h = \sqrt{R^2 - h^2}$ on $\mathcal{L}$. Using the PPP property and the relation $V_k(r) = r^k V_k(1)$, we have:

$$P(h) = 1 - \exp\left(-N \cdot V_k(\rho_h)\right) = 1 - \exp\left(-NV_k(1)(R^2 - h^2)^{k/2}\right). \tag{31}$$

2. Expected Union Volume The total expected volume $\mathbb{E}[V_{\text{union}}]$ is obtained by integrating $P(h)$ over the $(d-k)$ orthogonal dimensions:

$$\mathbb{E}[V_{\text{union}}] = \int_0^R S_{d-k-1} h^{d-k-1} \left[1 - \exp\left(-NV_k(1)(R^2 - h^2)^{k/2}\right)\right] dh, \tag{32}$$

where S_{d-k-1} is the surface area of a unit $(d-k-1)$-sphere.

3. Concentration of Measure and Thresholding In high-dimensional settings ($d - k \gtrapprox 10$), the volume of the hypercylinder concentrates heavily near the boundary $h = R$ due to the h^{d-k-1} term. We employ a threshold approximation by defining a critical height $\hat{h}$ where the coverage probability undergoes a sharp phase transition from 1 to 0. Setting the Poisson exponent to 1:

$$NV_k(1)(R^2 - \hat{h}^2)^{k/2} = 1 \implies \hat{h} = \sqrt{R^2 - (NV_k(1))^{-2/k}}. \tag{33}$$

Approximating the integrand as a step function where the region $h < \hat{h}$ is fully covered, the integral simplifies to the volume of a $(d-k)$-sphere of radius $\hat{h}$:

$$\mathbb{E}[V_{\text{union}}] \approx V_{d-k}(1)\hat{h}^{d-k}. \tag{34}$$

4. Balanced Error Expression and Complexity The balanced error ϵ is the normalized unfilled volume within the hypercylinder $V_{d-k}(R)$:

$$\epsilon = \frac{V_{d-k}(R) - \mathbb{E}[V_{\text{union}}]}{2V_{d-k}(R)} = \frac{1}{2}\left[1 - \left(\frac{\hat{h}}{R}\right)^{d-k}\right] = \frac{1}{2}\left[1 - \left(1 - \frac{1}{R^2(NV_k(1))^{2/k}}\right)^{\frac{d-k}{2}}\right]. \tag{35}$$

Applying the limit $(1-x)^m \approx e^{-mx}$ for large d:

$$\epsilon \approx \frac{1}{2}\left(1 - \exp\left[-\frac{d-k}{2R^2(NV_k(1))^{2/k}}\right]\right). \tag{36}$$

To maintain a constant error as d increases (assuming $d \gg k$), the argument of the exponential must be constant, requiring $N^{2/k} \sim O(d)$, or $N = O(d^{k/2})$. □

Proposition 5.9 (GMU → MLP.)**.** *Two GMUs (of any order k) in a GMU–linear network can exactly emulate the decision boundary of a single perceptron.*

Proof. First, consider a linear GMU design with no normalization $\eta = 1$ and exponential activation (proof works for any monotonic activation). The unit computes: $e^{-S^2_{k,W}(x,b)/\sigma^2}$, where $S_{k,W} = \min_{\theta \in \mathbb{R}^k} \left\|x - (W^T\theta + b)\right\|$ is the k-subspace distance. We show that a GMU-MLP with only two hidden neurons can learn a perceptron decision boundary exactly. Without loss of generality, align the coordinate system so the perceptron decision boundary is $x_1 = 0$. Choose any $W \in \mathbb{R}^{d\times k}$ of rank k whose first row is zero, ensuring all k active subspace directions are orthogonal to the x_1-axis. Let both GMUs in the first layer have the same weights W, but different biases. Set the biases to $b_\pm = (\mp c, 0, \ldots, 0)^\top$ for $c > 0$. Because $W^\top\theta$ has a zero in its first coordinate, the minimal distance along x_1 is unconstrained by W. Furthermore, the squared k-subspace distance minimizes out the active k dimensions, reducing for both units to $S^2_{k,W}(x, b_\pm) = (x_1 \pm c)^2 + \sum_{j=k+2}^d x_j^2$. Subtracting the two squared distances cancels the trailing unconstrained dimensions $\sum_{j=k+2}^d x_j^2$, yielding $S^2_{k,W}(x, b_-) - S^2_{k,W}(x, b_+) = 4cx_1$ For exponential activations $\phi(z) = e^{-z/\sigma^2}$, the linear layer output $e^{-S^2_{k,W}(x,b_+)/\sigma^2} - e^{-S^2_{k,W}(x,b_-)/\sigma^2}$ has the exact sign of x_1 and vanishes precisely on $x_1 = 0$, exactly recovering the perceptron decision boundary. □

Theorem 5.10. *Consider an input $X \sim \mathcal{U}[-0.5, 0.5]^d$. We consider a k-order GMU defined with zero bias, no normalization, and weight axes aligned with k of the d dimensions. Let $m = d - k$. The balanced classification error ϵ of a single-hidden-layer MLP with N hidden units attempting to emulate the decision boundary of this GMU is approximately lower bounded by:*

$$\epsilon \gtrapprox \left[\frac{m(m-1)}{(m+1)^2}\right] \left(\frac{m-1}{m+1}\right)^{\frac{m-1}{2}} \left(\frac{mV_m}{V_{m-1}}\right)^{\frac{2}{m-1}} N^{-\frac{2}{m-1}} \tag{37}$$

where $V_m = \frac{\pi^{m/2}}{\Gamma(m/2+1)}$ is the volume of the m-dimensional unit ball. To maintain a low error $\epsilon \ll 0.5$ as dimensionality d increases, the number of MLP units must scale exponentially with d, specifically $N = \Theta\left(\left(\frac{1}{e\,\epsilon}\right)^{\frac{m-1}{2}} \sqrt{m}\right)$.

Proof. **1. Dimensionality Reduction.** A k-order GMU defines a decision boundary where the distance to a k-dimensional subspace is R. In the $m = d - k$ dimensions orthogonal to this subspace, the boundary is an m-sphere. The MLP approximates this curved m-sphere using N flat $(m-1)$-dimensional disks (facets) of radius $L/2$.

2. Local Geometry and the Height Function. Consider a hyperplane at a distance $R - \rho$ from the center of the m-sphere. Let r be the radial distance from the center of the facet. The height gap $h(r)$ between the sphere and the flat facet is:

$$h(r) = \sqrt{R^2 - r^2} - (R - \rho) \tag{38}$$

For small $r, \rho \ll R$, we use the Taylor expansion:

$$h(r) \approx \left(R - \frac{r^2}{2R}\right) - (R - \rho) = \rho - \frac{r^2}{2R} \tag{39}$$

The hyperplane intersects the sphere at $r_{int} = \sqrt{2R\rho}$.

Inside Discrepancy: For $r < r_{int}$, the sphere is "above" the plane (Volume V_{in}). Outside Discrepancy: For $r_{int} < r < L/2$, the plane is "above" the sphere (Volume V_{out}).

3. Solving for the Optimal Sink Depth (ρ^*). The balanced error is minimized when $V_{in} = V_{out}$. We integrate the height function over the $(m-1)$-dimensional facet area $dS = (m-1)V_{m-1}r^{m-2}dr$.

Volume Inside:

$$V_{in} = \int_0^{\sqrt{2R\rho}} \left(\rho - \frac{r^2}{2R}\right)(m-1)V_{m-1}r^{m-2}dr \tag{40}$$

$$V_{in} = \frac{(m-1)V_{m-1}}{2R} \int_0^{\sqrt{2R\rho}} (2R\rho - r^2)r^{m-2}dr = \frac{2V_{m-1}}{(m^2-1)R}(2R\rho)^{\frac{m+1}{2}} \tag{41}$$

Volume Outside:

$$V_{out} = \int_{\sqrt{2R\rho}}^{L/2} \left(\frac{r^2}{2R} - \rho\right)(m-1)V_{m-1}r^{m-2}dr \tag{42}$$

$$V_{out} = \frac{(m-1)V_{m-1}}{2R}\left[\frac{(L/2)^{m+1}}{m+1} - \frac{2R\rho(L/2)^{m-1}}{m-1} - \left(\frac{(2R\rho)^{\frac{m+1}{2}}}{m+1} - \frac{2R\rho(2R\rho)^{\frac{m-1}{2}}}{m-1}\right)\right] \tag{43}$$

Setting $V_{in} = V_{out}$ and defining $x = \frac{\sqrt{2R\rho}}{L/2}$, the $(2R\rho)^{\frac{m+1}{2}}$ terms cancel across the equality, leaving:

$$\frac{1}{m+1} - \frac{x^2}{m-1} = 0 \quad \Rightarrow \quad x^2 = \frac{m-1}{m+1} \tag{44}$$

Substituting $x^2 = \frac{8R\rho}{L^2}$ (where $H = \frac{L^2}{8R}$ is the max tangent height), we find:

$$\rho^* = \frac{m-1}{m+1}H \tag{45}$$

4. Calculating the Minimized Gap Volume. Substituting the balance point $r_{int}^2 = (L/2)^2 \frac{m-1}{m+1}$ into the expression for V_{in} (Note that $V_{in} = V_{out}$ at the optimum):

$$V_{in} = V_{out} = \frac{V_{m-1} r_{int}^{m+1}}{R(m+1)} = \frac{V_{m-1}}{R(m+1)} \left(\frac{L}{2} \sqrt{\frac{m-1}{m+1}} \right)^{m+1} \tag{46}$$

By factoring out one power of $\frac{m-1}{m+1}$ to simplify the denominator, the minimized gap volume per unit scales as:

$$V_{out} = \frac{(m-1) V_{m-1} (L/2)^{m+1}}{R(m+1)^2} \left(\frac{m-1}{m+1} \right)^{\frac{m-1}{2}} \tag{47}$$

5. Surface Coverage and Balanced Error. The N facets must cover the sphere's surface area $S_{m-1} = m V_m R^{m-1}$:

$$N V_{m-1} (L/2)^{m-1} = m V_m R^{m-1} \quad \Rightarrow \quad \frac{L}{2} = R \left(\frac{m V_m}{N V_{m-1}} \right)^{\frac{1}{m-1}} \tag{48}$$

The balanced error ϵ is the normalized discrepancy volume (with $V_{in} = V_{out}$):

$$\epsilon = \frac{N V_{out}}{V_m R^m} = \frac{N(m-1) V_{m-1} (L/2)^{m+1}}{V_m R^{m+1} (m+1)^2} \left(\frac{m-1}{m+1} \right)^{\frac{m-1}{2}} \tag{49}$$

Substituting the expression for $L/2$:

$$\epsilon = \frac{N(m-1) V_{m-1}}{V_m (m+1)^2} \left(\frac{m-1}{m+1} \right)^{\frac{m-1}{2}} \left(\frac{m V_m R^{m-1}}{N V_{m-1}} \right)^{\frac{m+1}{m-1}} \frac{1}{R^{m+1}} \tag{50}$$

Re-arranging, we have:

$$\epsilon = \frac{m(m-1)}{(m+1)^2} \left(\frac{m-1}{m+1} \right)^{\frac{m-1}{2}} \left(\frac{m V_m}{V_{m-1}} \right)^{\frac{2}{m-1}} N^{-\frac{2}{m-1}} \tag{51}$$

To derive the scaling law, we rearrange the error expression to solve for N. Solving for N gives:

$$N = \left[\frac{m(m-1)}{\epsilon (m+1)^2} \right]^{\frac{m-1}{2}} \left(\frac{m-1}{m+1} \right)^{\frac{(m-1)^2}{4}} \left(\frac{m V_m}{V_{m-1}} \right). \tag{52}$$

As $m \to \infty$, each factor admits a clean asymptotic form:

$$\left[\frac{m(m-1)}{(m+1)^2} \right]^{\frac{m-1}{2}} = \exp\left(\frac{m-1}{2} \log\left(1 - \frac{3}{m} + O(m^{-2}) \right) \right) \to e^{-3/2}, \tag{53}$$

$$\left(\frac{m-1}{m+1} \right)^{\frac{(m-1)^2}{4}} = \exp\left(\frac{(m-1)^2}{4} \log\left(1 - \frac{2}{m+1} \right) \right) \sim e^{-m/2}, \tag{54}$$

$$\frac{m V_m}{V_{m-1}} \sim \sqrt{2\pi m}. \tag{55}$$

Substituting these into the expression for N yields the asymptotic form:

$$N \approx e^{-3/2} \left(\frac{1}{\epsilon} \right)^{\frac{m-1}{2}} e^{-m/2} \sqrt{2\pi m} = \left(\frac{1}{e\epsilon} \right)^{\frac{m-1}{2}} e^{-3/2} \sqrt{2\pi m}. \tag{56}$$

Thus, to maintain a low error ϵ as the effective dimensionality m increases, the number of MLP units must grow exponentially in m:

$$N = \Theta\left(\left(\frac{1}{e\,\epsilon} \right)^{\frac{m-1}{2}} \sqrt{m} \right). \tag{57}$$

up to constant and sub-exponential factors.

□

Theorem 6.1. *Let $\mathcal{S} \subset \mathbb{R}^d$ be a compact, C^3-smooth, $(d-1)$-dimensional Riemannian submanifold with finite $(d-1)$-dimensional volume $\mathcal{V}_\mathcal{S}$. We assume the manifold is locally well-approximated by its second-order Taylor expansion. Let v_{d-1} denote the volume of the $(d-1)$-dimensional unit ball. To achieve a global Hausdorff error ϵ, let $N(\epsilon, \mathcal{F})$ denote the expected minimum number of units required to form an ϵ-cover of $\mathcal{S}$ under a uniform data distribution $p \sim U(\mathcal{S})$. Let $\{\kappa_i(p)\}_{i=1}^{d-1}$ be the ordered principal curvatures at $p \in \mathcal{S}$ such that $|\kappa_1(p)| \geq |\kappa_2(p)| \geq \cdots \geq |\kappa_{d-1}(p)|$.*

The expected unit complexity for different computational architectures is:

$$N(\epsilon, \mathcal{F}) \approx \frac{\mathcal{V}_\mathcal{S}}{v_{d-1}(2\epsilon)^{\frac{d-1}{2}}} \mathbb{E}_{p\sim\mathcal{S}} \left[\Delta\kappa_\mathcal{F}(p)^{\frac{d-1}{2}} \right] \tag{58}$$

where the residual curvature $\Delta\kappa_\mathcal{F}$ is defined as:

1. ***MLP:*** $\Delta\kappa_{MLP} = |\kappa_1(p)|$.
2. ***RBF:*** $\Delta\kappa_{RBF} = \frac{|\kappa_1(p) - \kappa_{d-1}(p)|}{2}$.
3. ***k-GMU:*** $\Delta\kappa_{GMU} = \min\left(\min_{k' \in \{1,\ldots,k\}} \max\left(|\kappa_{d-k'}(p)|, \frac{|\kappa_1(p)-\kappa_{d-k'-1}(p)|}{2}\right), \frac{|\kappa_1(p)-\kappa_{d-1}(p)|}{2}\right)$.

Proof. **1. Geometric Setup: The Manifold Approximation Framework**

Consider a $(d-1)$-dimensional hypersurface $\mathcal{S} \subset \mathbb{R}^d$. At any point $p \in \mathcal{S}$, we establish a local coordinate system with the origin at p and the d-th axis aligned with the normal vector $\mathbf{n}_p$ (the direction perpendicular to the surface). The manifold is locally the graph of a function $z = f(x)$, where $x \in \mathbb{R}^{d-1}$. Since p is the origin and the frame is tangent, $f(0) = 0$ and $\nabla f(0) = 0$. The Taylor expansion of the manifold's surface is:

$$z = \frac{1}{2} x^T \mathbf{H}_f x + \mathcal{O}(\|x\|^3) + .. \tag{59}$$

The eigenvalues of the Hessian matrix $\mathbf{H}_f$ $\{\kappa_1, \kappa_2, \ldots, \kappa_{d-1}\}$ represent the **principal curvatures**, which quantify how much the manifold bends along each tangent direction.

2. Local Error and Spectral Radius

A local unit defines a decision boundary $\hat{\mathcal{S}}$ via its energy function $Q(X)$. To bound the Hausdorff distance (the maximum gap between the real manifold and our approximation) by ϵ on a patch of radius h, the unit's boundary must satisfy $|f(x) - Q_{\text{boundary}}(x)| \leq \epsilon$. This local error $\delta(h)$ is dominated by the difference between the manifold's curvature and the unit's boundary curvature:

$$\delta(h) \approx \frac{1}{2} \max_{\|x\| \leq h} |x^T(\mathbf{H}_f - \mathbf{H}_Q)x| = \frac{h^2}{2} \rho(\mathbf{H}_f - \mathbf{H}_Q) \tag{60}$$

where $\rho(\cdot)$ is the spectral radius, denoting the maximum absolute eigenvalue of the difference.

3. MLP Unit Complexity

The standard Perceptron (MLP) defines a decision boundary that is a $(d-1)$-dimensional hyperplane. Consequently, its tangent Hessian $\mathbf{H}_{MLP}$ is identically zero ($\lambda = 0$).

The local approximation error at point p is determined by the manifold's full second-order geometry relative to this flat boundary:

$$\delta(h) = \frac{1}{2} \max_{\|x\| \leq h} |x^T(\mathbf{H}_f - \mathbf{0})x| = \frac{h^2}{2} \max_i |\kappa_i| \tag{61}$$

Since we ordered the principal curvatures such that $|\kappa_1| \geq |\kappa_2| \geq \cdots \geq |\kappa_{d-1}|$, the spectral radius is dominated by the maximum principal curvature, $\Delta\kappa_{MLP} = |\kappa_1|$.

4. RBF Unit Complexity

An RBF unit's decision boundary is a $(d-1)$-sphere. For the sphere to be tangent to $\mathcal{S}$ at p, the center b must lie on the normal $\mathbf{n}_p$. Let $b = (0, \ldots, 0, R)$. The boundary equation is $\sum x_i^2 + (z-R)^2 = R^2$. Solving for z near the origin:

$$z = R - \sqrt{R^2 - \|x\|^2} \approx \frac{1}{2R}\|x\|^2 = \frac{1}{2}\lambda \mathbf{I}_{d-1} \tag{62}$$

where $\lambda = 1/R$ is the constant curvature of the RBF. The local error is $\delta(h) = \frac{h^2}{2}\max_i |\kappa_i - \lambda|$. To minimize this error, we choose λ^* to be the midpoint of the curvature extremes: $\lambda^* = \frac{\kappa_1+\kappa_{d-1}}{2}$. The residual curvature is thus $\Delta\kappa_{RBF} = \frac{\kappa_1-\kappa_{d-1}}{2}$.

5. GMU Unit Complexity

The linear k-GMU boundary is defined by the distance to a k-dimensional linear subspace. As a k-GMU can exactly emulate a k'-GMU where $k' \leq k$, its tangent Hessian $\mathbf{H}_{GMU}$ has *up to* k zero eigenvalues (along its internal subspace directions) and a minimum of $d-k-1$ identical eigenvalues λ. The unit selectively turns off up to k directions of manifold curvature by aligning its internal subspace with them. Let $\mathcal{I}_{\text{off}}$ be the $k' \leq k$ directions turned off and $\mathcal{I}_{\text{on}}$ the remaining directions. The error is:

$$\delta(h) = \frac{1}{2}h^2 \cdot \min_{k' \in \{0,\dots,k\}} \max\left(\max_{i \in \mathcal{I}_{\text{off}}} |\kappa_i|, \min_{\lambda} \max_{j \in \mathcal{I}_{\text{on}}} |\kappa_j - \lambda|\right) \tag{63}$$

The minimization occurs as the k-GMU can choose an internal $k' \leq k$ dimensional subspace that minimizes this error. The optimal strategy for each k' is to assign the k' smallest absolute curvatures to the subspace directions. The remaining largest curvatures are matched by λ at their midpoint. This yields the residual curvature:

$$\Delta\kappa_{GMU} = \min\left(\min_{k' \in \{1,\dots,k\}} \max\left(|\kappa_{d-k'}(p)|, \frac{|\kappa_1(p) - \kappa_{d-k'-1}(p)|}{2}\right), \frac{|\kappa_1(p) - \kappa_{d-1}(p)|}{2}\right) \tag{64}$$

6. Integration to Global Complexity To satisfy a global Hausdorff error $d_H(\mathcal{S}, \hat{\mathcal{S}}) \leq \epsilon$, every point on the manifold must be within distance ϵ of the approximating boundary. This requires an ϵ-cover of the manifold using local patches. To minimize the total number of units required, we maximize the area covered by each unit by setting the local error $\delta(h)$ exactly to the global tolerance ϵ. Solving $\delta(h) = \epsilon$ yields the maximal local patch radius $h(p) = \sqrt{2\epsilon/\Delta\kappa(p)}$.

Each unit thus covers a $(d-1)$-dimensional local volume $V_{patch}(p) = v_{d-1}h(p)^{d-1}$. The minimum number of units N needed to cover the manifold is the integral of the local unit density $\rho(p) = [V_{patch}(p)]^{-1}$ over the manifold surface. By normalizing by the total manifold volume $\mathcal{V}_{\mathcal{S}}$, we express this as an expectation over the uniform distribution $p \sim U(\mathcal{S})$:

$$N(\epsilon) \approx \int_{\mathcal{S}} \frac{1}{v_{d-1}h(p)^{d-1}} dV_p = \frac{\mathcal{V}_{\mathcal{S}}}{v_{d-1}} \mathbb{E}_{p\sim\mathcal{S}}\left[h(p)^{-(d-1)}\right] \tag{65}$$

Substituting the geometric relation $h(p)^{-(d-1)} = (\Delta\kappa/2\epsilon)^{\frac{d-1}{2}}$ yields the final complexity:

$$N(\epsilon) = \frac{\mathcal{V}_{\mathcal{S}}}{v_{d-1}(2\epsilon)^{\frac{d-1}{2}}} \mathbb{E}_{p\sim\mathcal{S}}\left[\Delta\kappa(p)^{\frac{d-1}{2}}\right] \tag{66}$$

This confirms that the complexity scales as $O(\epsilon^{-\frac{d-1}{2}})$. Crucially, the k-GMU reduces the base of the exponent from the absolute curvature $|\kappa_1|$ to the residual curvature $\Delta\kappa_{GMU}$, leading to an exponential reduction in N as d increases.

□

Corollary 6.2 (Exponential Efficiency Gap)**.** *Assume the manifold $\mathcal{S}$ is strictly anisotropic such that $|\kappa_1(p)| > |\kappa_2(p)| > \cdots > |\kappa_{d-1}(p)| > 0$ for all $p \in \mathcal{S}$. Let $n = \frac{d-1}{2}$ be the complexity exponent.*

For any two function spaces $\mathcal{F}_1, \mathcal{F}_2$, the relative unit efficiency scales as: $\frac{N(\epsilon,\mathcal{F}_1)}{N(\epsilon,\mathcal{F}_2)} \approx \left[\frac{\|\Delta\kappa_1\|_{L_n(\mathcal{S})}}{\|\Delta\kappa_2\|_{L_n(\mathcal{S})}}\right]^n$, where $\|f\|_{L_n(\mathcal{S})} = (\mathbb{E}_{p\sim\mathcal{S}}[|f(p)|^n])^{1/n}$.

In particular, under strict anisotropy, there exist geometric constants $\gamma_1, \gamma_2, \gamma_3 > 1$ such that:

$$\frac{N(\epsilon,\mathcal{F}_{MLP})}{N(\epsilon,\mathcal{F}_{GMU})} = \Omega(\gamma_1^n), \quad \frac{N(\epsilon,\mathcal{F}_{RBF})}{N(\epsilon,\mathcal{F}_{GMU})} = \Omega(\gamma_2^n), \quad \frac{N(\epsilon,\mathcal{F}_{MLP})}{N(\epsilon,\mathcal{F}_{RBF})} = \Omega(\gamma_3^n). \tag{67}$$

Proof. **1. Pointwise Residual Hierarchy**
Based on the spectral radius of the Hessian difference $\rho(\mathbf{H}_f - \mathbf{H}_Q)$, we establish the following hierarchy for any $p \in \mathcal{S}$:

- **MLP vs. RBF:** The MLP residual is $\Delta\kappa_{MLP} = |\kappa_1|$. The RBF residual is $\Delta\kappa_{RBF} = \frac{|\kappa_1 - \kappa_{d-1}|}{2}$. By the triangle inequality, $\frac{|\kappa_1 - \kappa_{d-1}|}{2} \leq \frac{|\kappa_1| + |\kappa_{d-1}|}{2}$. Since $|\kappa_1| > |\kappa_{d-1}|$, then $\Delta\kappa_{RBF} < |\kappa_1|$. If κ_1 and κ_{d-1} share the same sign, the gap is even larger.
- **RBF vs. GMU:** The GMU residual $\Delta\kappa_{GMU}$ is the minimum over k' strategies. One such strategy is the RBF equivalent ($k' = 0$), but for $k \geq 1$, the GMU can choose to nullify the direction of $|\kappa_{d-k}|$. Since $|\kappa_{d-k}| < |\kappa_1|$ and $|\kappa_1 - \kappa_{d-k-1}| < |\kappa_1 - \kappa_{d-1}|$ under strict anisotropy, the GMU residual is strictly lower than the RBF residual: $\Delta\kappa_{GMU}(p) < \Delta\kappa_{RBF}(p)$.

2. Rigorous Complexity Ratio

The total complexity $N(\epsilon, \mathcal{F})$ integrates the local unit density $\rho(p) \propto \Delta\kappa(p)^n$. The ratio of complexities for a fixed error ϵ and manifold $\mathcal{S}$ is:

$$\frac{N(\epsilon, \mathcal{F}_1)}{N(\epsilon, \mathcal{F}_2)} = \frac{\mathbb{E}[\Delta\kappa^n_{\mathcal{F}_1}]}{\mathbb{E}[\Delta\kappa^n_{\mathcal{F}_2}]} = \left[\frac{\|\Delta\kappa_{\mathcal{F}_1}\|_{L_n(\mathcal{S})}}{\|\Delta\kappa_{\mathcal{F}_2}\|_{L_n(\mathcal{S})}}\right]^n \tag{68}$$

3. Comparison via Geometric Constants

Let $\gamma(p) = \frac{\Delta\kappa_{\mathcal{F}_1}(p)}{\Delta\kappa_{\mathcal{F}_2}(p)}$. For a compact manifold $\mathcal{S}$ with continuous principal curvatures and strict anisotropy, the infimum $\gamma_{min} = \inf_{p\in\mathcal{S}} \gamma(p)$ is strictly greater than 1.

- For **MLP vs. GMU**, we have $\Delta\kappa_{MLP}(p)/\Delta\kappa_{GMU}(p) \geq \gamma_1 > 1$.
- For **RBF vs. GMU**, we have $\Delta\kappa_{RBF}(p)/\Delta\kappa_{GMU}(p) \geq \gamma_2 > 1$.

Applying the monotonicity of the expectation:

$$\frac{N(\epsilon, \mathcal{F}_{MLP})}{N(\epsilon, \mathcal{F}_{GMU})} \geq \gamma_1^n = \Omega\left(\gamma_1^{\frac{d-1}{2}}\right) \tag{69}$$

This confirms that the GMU achieves an exponential efficiency gain over the MLP and RBF by minimizing the base of the complexity exponent through directional curvature nullification. The result similar extends to the case where one compares RBFs and MLPs with γ_3. □

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We include every contribution in the main part and the Appendix.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

Justification: We discuss the gradient collapse limitations of GMUs just like RBFs in the conclusion.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: We outline both assumptions and provide full proofs.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We have provided all details in appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Code has been shared via anonymous github.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not

including code, unless this is central to the contribution (e.g., for a new open-source benchmark).

- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Partly provided in the paper and the rest in the code.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We include this for most experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.

- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide compute times and FLOP analysis

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: Not applicable.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We don't release new datasets.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [N/A]

    Justification: We mostly use our own written code.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: We don't introduce new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Not applicable.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: No LLMs were used in any method development.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.