

# Scale-Agnostic Kolmogorov-Arnold Geometry in Neural Networks

Mathew Vanherreweghe<sup>1</sup>      Michael H. Freedman<sup>2,3\*</sup>      Keith M. Adams<sup>1†</sup>

<sup>1</sup>Pebblebed

<sup>2</sup>Logical Intelligence

<sup>3</sup>Center of Mathematical Sciences and Applications, Harvard University

mathew@pebblebed.com

## Abstract

Recent work by Freedman and Mulligan demonstrated that shallow multilayer perceptrons spontaneously develop Kolmogorov-Arnold geometric (KAG) structure during training on synthetic three-dimensional tasks. However, it remained unclear whether this phenomenon persists in realistic high-dimensional settings and what spatial properties this geometry exhibits.

We extend KAG analysis to MNIST digit classification (784 dimensions) using 2-layer MLPs with systematic spatial analysis at multiple scales. We find that KAG emerges during training and appears consistently across spatial scales, from local 7-pixel neighborhoods to the full  $28 \times 28$  image. This scale-agnostic property holds across different training procedures: both standard training and training with spatial augmentation produce the same qualitative pattern. These findings reveal that neural networks spontaneously develop organized, scale-invariant geometric structure during learning on realistic high-dimensional data.

## 1 Introduction

### 1.1 The Kolmogorov-Arnold Theorem and Neural Networks

In 1957, Andrey Kolmogorov and Vladimir Arnold proved a remarkable theorem about the representation of multivariate continuous functions [Kolmogorov, 1957, Arnold, 1957]. The Kolmogorov-Arnold (KA) theorem states that any continuous function of multiple variables can be exactly represented as a finite composition and superposition of continuous functions of a single variable. More precisely, any continuous function  $f : [0, 1]^n \rightarrow \mathbb{R}$  can be written as:

$$f(x_1, \dots, x_n) = \sum_{q=0}^{2n} \Phi_q \left( \sum_{p=1}^n \phi_{q,p}(x_p) \right) \quad (1)$$

where  $\Phi_q$  and  $\phi_{q,p}$  are continuous univariate functions. This theorem provided a negative answer to Hilbert’s thirteenth problem and revealed that high-dimensional functions possess a surprisingly low-dimensional representation. This representation has the remarkable property that the inner functions  $\phi_{q,p}$  can be chosen universally (that is, independent of  $f$ ) and possess an idiosyncratic

---

\*Mathematical formulation and theoretical insights.

†Machine learning expertise and practical guidance.

“texture”, dubbed Kolmogorov-Arnold geometry (KAG) in Freedman and Mulligan [2025], and quantified through observables discussed below.

The connection to neural networks is one-to-one in the KAN construction [Liu et al., 2024], but is also apparent in shallow multilayer perceptrons (MLPs). A standard two-layer network with activation function  $\sigma$  computes:

$$f(x) = \sum_{i=1}^h v_i \sigma(w_i^T x + b_i) \quad (2)$$

where  $x \in \mathbb{R}^d$  is the input,  $h$  is the hidden layer width,  $w_i \in \mathbb{R}^d$  are weight vectors, and  $v_i \in \mathbb{R}$  are output weights. The inner map  $\phi(x) = \sigma(Wx + b) \in \mathbb{R}^h$  transforms the input into a hidden representation, while the outer map linearly combines these features. This architecture is, indeed, merely a simplified KA representation, suggesting that neural networks might organically discover KAG-like structure during training.

## 1.2 Spontaneous Emergence of KA Geometry

Recent work by Freedman and Mulligan [2025] made a surprising discovery: vanilla MLPs trained on simple classification tasks spontaneously develop geometric signatures consistent with the Kolmogorov-Arnold representation. Analyzing the Jacobian matrix  $J = \partial\Phi/\partial x$  of the inner map, they identified three hallmark features of KAG:

1. **Zero rows:** For a typical  $x$ , many hidden units become locally constant, with  $\|\nabla\phi_i(x)\|_2 \approx 0$  where the row index  $i$  depends on  $x$ .
2. **Minor concentration (MC):** The distribution of  $k \times k$  determinants (minors) of  $J$  has a large spike at zero, but is simultaneously heavy-tailed, with a few remarkably large values. (MC is measured by three observables called: Participation ratio, KL-divergence, and RR.)
3. **Alignment:** The Jacobian structure exhibits non-generic alignment that is not preserved under random orthogonal rotations

These signatures emerge without explicit regularization or architectural constraints—they appear to be a natural consequence of gradient-based optimization. Freedman and Mulligan observed this phenomenon in shallow MLPs (single hidden layer, width  $h = 32$ ) trained on synthetic three-dimensional functions such as XOR, linear separators, and random Boolean functions.

This discovery suggests that KAG may represent a general organizational principle in neural networks, though it remained unclear whether these patterns persist beyond simple, low-dimensional settings.

## 1.3 This Work

While Freedman and Mulligan demonstrated KAG emerges in MLPs trained on synthetic 3D functions, their analysis left open questions about whether these patterns persist in high-dimensional, structured, real-world data. Moreover, their low-dimensional synthetic tasks did not permit investigation of spatial properties: questions about whether geometry varies across spatial regions or scales require data with richer spatial structure.

We extend their framework to MNIST digit classification (784-dimensional inputs, 10-class task) using 2-layer MLPs with varying capacity ( $h \in \{64, 128, 256\}$ ). We confirm that KAG emerges in this realistic high-dimensional setting and introduce systematic spatial analysis to examine geometric structure at different scales.

We find that KAG appears consistently across spatial scales from local 7-pixel neighborhoods to the full  $28 \times 28$  image. Most strikingly, this scale-agnostic property proves robust to training procedures: both standard training and training with spatial augmentation produce the same qualitative geometric patterns, though augmented training yields lower absolute participation ratios (30% reduction), consistent with reduced geometric ‘frustration’ from exposure to diverse data variations.

Section 2 describes our experimental methodology, Section 3 presents the main results, Section 4 discusses implications and connections to prior work, and Section 5 concludes.

## 2 Methods

We analyze KAG in 2-layer MLPs trained on MNIST digit classification through systematic spatial analysis at multiple scales. Complete implementation details, including all hyperparameters, hardware specifications, and reproducibility information, are provided in Appendix A.

We train 2-layer MLPs with GELU activations on MNIST digit classification ( $28 \times 28$  grayscale images, 10 classes, 784-dimensional flattened input), varying the hidden dimension  $h \in \{64, 128, 256\}$  to study how network capacity affects KAG. For both the first hidden layer map (L1) and second hidden layer map (L2), we compute Jacobians with respect to the input  $x$  to examine how KA signatures evolve with training. Models are trained using AdamW for 200 epochs, with 5 random seeds per configuration for statistical robustness (error bars reflect  $\pm 2$  standard deviations relative to these 5 runs). To test whether KAG is robust to training procedures, we train an additional set of models with the same configurations (5 seeds per hidden size) using spatial data augmentation with RandomAffine transformations (translation  $\pm 4$  pixels horizontal/vertical) applied during training. Spatial analysis refers to the study of the scale at which KAG appears when analyzing minors based on pixels in both localized and well separated regions of the 2-dimensional input image. This, of course, is quite distinct from the KAG of the first layer map from  $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^h$ , and represents a new dimension of KAG exploration.

For each hidden layer representation  $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^h$ , we compute the Jacobian matrix  $J = \frac{\partial \phi}{\partial x} \in \mathbb{R}^{h \times d}$  using PyTorch’s vectorized automatic differentiation. For the first hidden layer (L1), each of the  $d = 784$  columns corresponds to the derivative with respect to a specific input pixel in the flattened  $28 \times 28$  image. For the second hidden layer (L2), we compute the Jacobian  $\frac{\partial h_2}{\partial x}$  where  $h_2$  represents the second layer activations. Through the chain rule, each of the  $d = 784$  columns captures how the second layer outputs change with respect to each input pixel, representing the end-to-end sensitivity of the network’s internal representations to the input.

In Freedman and Mulligan [2025], the term KAG is defined in reference to the Jacobian of the first layer map. It is not a sharp, black-or-white criterion but rather a statistical tendency of the  $\text{Jac}(x)$  to assume a form similar to the Jacobians of Kolmogorov and Arnold’s first layer maps [Kolmogorov, 1957, Arnold, 1957]. The idealized KAG of  $\text{Jac}(x)$  is an  $d \times h$  matrix in which each of the  $h$  columns contains at most a single non-zero entry, where  $d$  is the input dimension and  $h$  is the number of hidden layer neurons. The observables detailed below (vanishing rows, PR, KL-divergence, and RR) quantify similarity to the KA ideal and contrariwise the departure from our untrained, random initializations.

To quantify the many-body character of KAG, we compute  $k \times k$  determinants (minors) for  $k \in \{1, 2, 3\}$  from the Jacobian. These minors measure the “ $k$ -volume” or determinant of a  $k \times k$  submatrix of  $\text{Jac}(x)$  formed by selecting some fixed set of  $k$  columns and some fixed set of  $k$  rows. We chose small  $k$ , because we wanted to avoid numerical issues with larger determinants, yet wanted to see a trend with increasing  $k$ . By design these minors are not rotationally invariant, but leverage

the preferred neuron basis. We always use the absolute value of the minor as we have detected no signal in its sign, only its magnitude. Since exhaustively computing all minors is prohibitively expensive (e.g.,  $\binom{784}{3} \approx 79$  million 3-minors for MNIST), we randomly sample column combinations (up to 10,000 samples) and use adaptive chunking to manage GPU memory.

We quantify KAG using three metrics:

- **Participation Ratio (PR):** For a set of minor values  $\{m_i\}$ , the participation ratio is:

$$\text{PR} = \frac{\sqrt{\sum_i |m_i|^2}}{\sum_i |m_i|} = \frac{\|m\|_2}{\|m\|_1} \quad (3)$$

A high PR indicates that few minors dominate (heavy-tailed distribution), consistent with KAG. We report the PR Ratio =  $\text{PR}_{\text{trained}}/\text{PR}_{\text{random}}$ , where values significantly greater than 1 indicate stronger concentration in trained networks.

- **KL Divergence:** We measure how much the minor distribution changes during training:

$$\text{KL}(P_{\text{trained}} \| P_{\text{init}}) = \sum_i P_{\text{trained}}(i) \log \frac{P_{\text{trained}}(i)}{P_{\text{init}}(i)} \quad (4)$$

where  $P$  denotes the normalized histogram of minor magnitudes. Large KL values indicate substantial distributional change. This measure was established as a reliable tool for detecting training induced shifts in minor concentration, correlating well with other metrics of KAG.(refer to Freedman and Mulligan [2025] for details)

- **Rotation Ratio (RR):** To test whether Jacobian structure is aligned with the neuron bases axis of  $\mathbb{R}^d$  and  $\mathbb{R}^h$ , we apply random orthogonal transformations  $J_{\text{rotated}} = J \cdot R$  where  $R \in \text{SO}(h)$ , computing:

$$\text{RR} = \frac{\mathbb{E}_R[\text{PR}(J \cdot R)]}{\text{PR}(J)} \quad (5)$$

averaged over 100 random rotations. Values significantly greater than 1 indicate alignment (KA-like structure).

To test whether KAG appears at different spatial scales and locations, we perform three complementary analyses. First, for Euclidean ball sampling, we select varying radii  $R \in \{7, 14, 21, 28\}$  pixels, randomly choose a seed pixel, and include all pixels within Euclidean distance  $R$ , producing approximately 150, 600, 1400, and 2500 pixels respectively. We then compute PR using only the Jacobian columns corresponding to these sampled pixels.

Second, for rectangular patch sampling, we partition the  $28 \times 28$  image into spatial regions (top-left, top-right, bottom-left, bottom-right, center) and compute PR separately for pixels in each region.

Third, to complement the localized euclidean ball analyses, we test whether geometry extends across space by constraining minor sampling to use only spatially well-separated pixels. When selecting  $k$  columns (pixels) to form a  $k \times k$  minor, we enforce that the selected pixels are pairwise separated by at least  $D$  pixels (Euclidean distance), with  $D \in \{0, 3, 5, 7, 10, 14\}$  pixels.

For all spatial analyses, we use 50 test images and report distributions across both images and random seed selections.

We compute all metrics at the end of training (epoch 200 for both standard and augmented models), comparing trained networks against random initialization baselines using the same architecture. We report mean  $\pm 2$  standard deviations across 5 random seeds for the  $h \in \{64, 128, 256\}$  configurations.

### 3 Results

Let us say concretely: We demonstrate that KAG emerges during training in 2-layer MLPs on MNIST, appearing consistently across spatial scales from local neighborhoods to the full image. This scale-agnostic property is robust to training procedures, appearing in both standard and augmented training regimes.

Figure 1 compares trained networks against random initialization across three metrics (PR, KL divergence, RR) for varying hidden dimensions ( $h \in \{64, 128, 256\}$ ) and minor orders ( $k \in \{1, 2\}$ ) for both standard and augmented training. Trained networks consistently show markedly stronger KA signatures than random initialization across all metrics, with larger networks exhibiting more pronounced geometric structure. Layer 1 shows stronger signatures than layer 2, and higher-order minors ( $k = 2$ ) reveal stronger effects than  $k = 1$ .

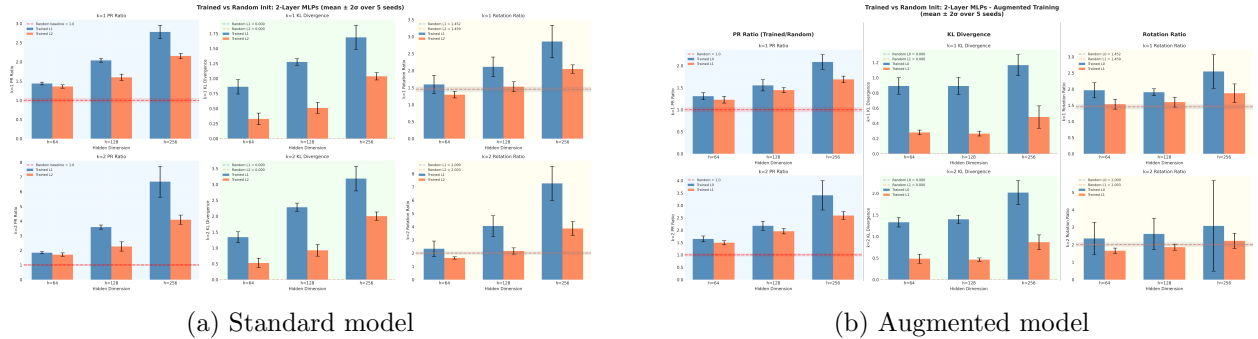


Figure 1: KAG emerges during training for both standard and augmented models. First Layer map (blue bars) and second layer map (orange bars) across three KAG metrics: PR ratio, KL divergence, and RR. Results shown for minor orders  $k \in \{1, 2\}$  with hidden dimensions  $h \in \{64, 128, 256\}$ . Red horizontal lines and shaded regions (mean  $\pm 2$  standard deviations) indicate baseline values for these metrics across 5 random seeds. Trained networks consistently exceed random baselines, with larger networks showing stronger geometric signatures. Both training procedures show the same qualitative patterns. Note that large RR error bars in the augmented model ( $k=2$ ) are due to one seed with exceptionally high values, not bidirectional variance.

The larger KAG signatures for L1 compared to L2 supports the hypothesis that KAG is primarily about *data preparation*—the first layer arranges data propitiously for later analysis. This is consistent with KA signatures diminishing once the network has been fully traversed. Since L2 is a composition of a linear output map with L1, we expect to see some residual signatures, but a linear map can only do so much to preserve or enhance geometric structure.

Notably, all model sizes achieved similar classification performance ( $\sim 98\%$  test accuracy), indicating that the differences in KAG across network capacities are not driven by differences in task performance. MNIST is a sufficiently simple task that even the smallest network ( $h = 64$ ) learns the classification problem effectively, yet larger networks develop stronger geometric signatures during this learning process. To put this in perspective, even our largest hidden dimension ( $h \leq 256$ ) are all smaller than the input dimension ( $d = 784$ ), opposite to the original KA construction where  $h \geq 2d + 1$ .

#### 3.1 Spatial Properties of KA Geometry

Having established that KAG emerges during training, we investigate at what spatial scales it appears.

Before presenting spatial analysis results, we clarify the three distinct spaces involved in our analysis. First, there is the input space (784-dimensional flattened MNIST images). Second, there is the hidden space where the network’s learned representations live ( $h$ -dimensional,  $h \in \{64, 128, 256\}$ ). We compute Jacobians for both hidden layers (L1 and L2) with respect to the input to examine how KAG signatures evolve with depth. KAG concerns the local properties of these maps from input to hidden space, specifically the structure of the Jacobian matrices  $\text{Jac}(h(x))$  at typical inputs  $x$ . Third, there is the spatial structure of the MNIST image itself ( $28 \times 28$  pixel grid). Our spatial analyses probe how KAG (which describes the input to hidden maps) varies when we sample Jacobian columns corresponding to different spatial regions of the MNIST image. This allows us to ask how KAG varies across different spatial regions of the image.

We now examine whether KAG appears uniformly across spatial scales or varies with the region sampled from the MNIST image. To ensure our findings are not artifacts of training procedure, we compare both standard and augmented models.

First, we test local to medium scales by sampling minors from pixels within varying radii  $R \in \{7, 14, 21, 28\}$  pixels from random seed points. Figure 2a illustrates this sampling procedure for different radii. Figure 2b shows that both models exhibit scale-agnostic KAG: PR ratios remain well above baseline (1.0) across all radii for both training procedures. Notably, the augmented model shows approximately 30% lower PR ratios while maintaining the same scale-invariant pattern, consistent with reduced geometric ‘frustration’ from exposure to diverse data variations during training.

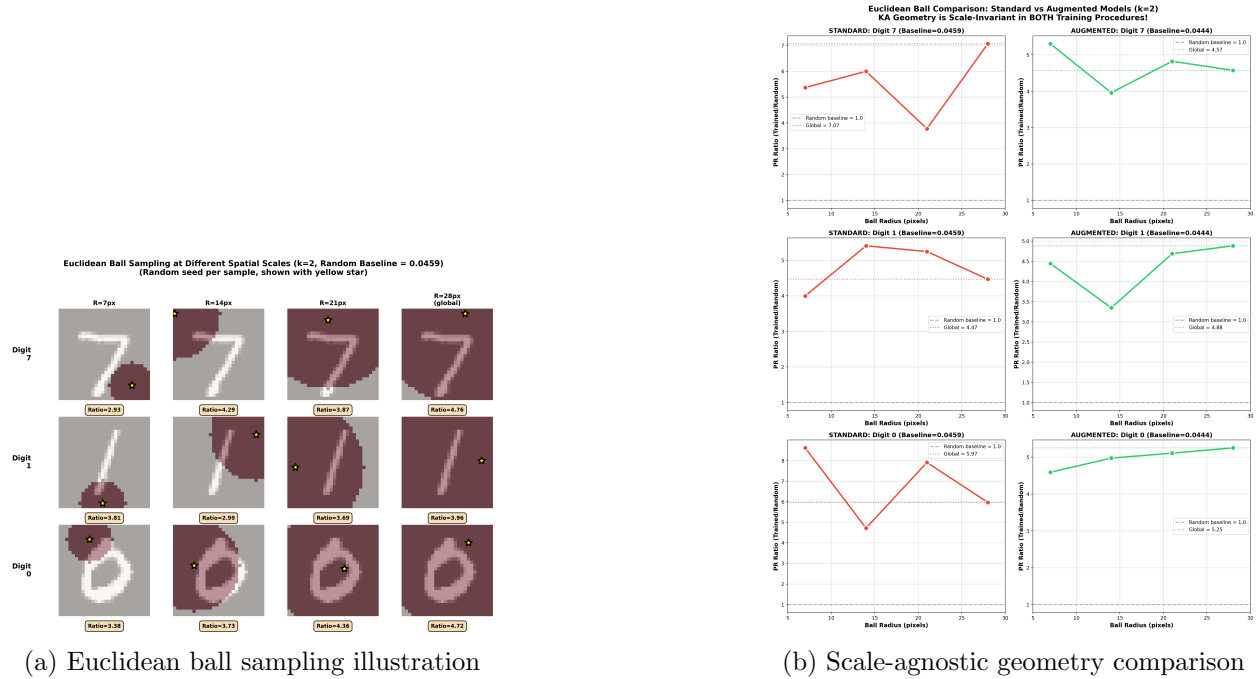


Figure 2: Scale-agnostic KAG across spatial scales. (a) Illustration of Euclidean ball sampling at radii  $R \in \{7, 14, 21, 28\}$  pixels. Yellow stars mark random seed pixels, with sampled regions highlighted. (b) Both standard and augmented models exhibit scale-agnostic KAG: PR ratios remain well above baseline (1.0) across all ball radii for  $k=2$  minors. Augmented model exhibits lower PR ratios (approximately 30% reduction), consistent with reduced ‘frustration’ from exposure to diverse data variations.

Second, we test whether geometry extends globally by constraining minors to well-separated

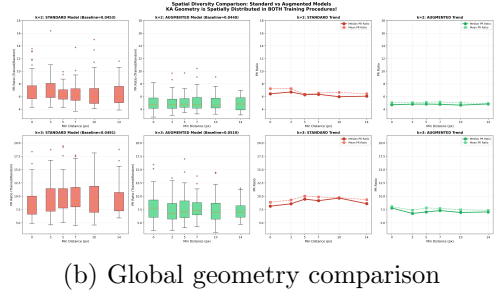
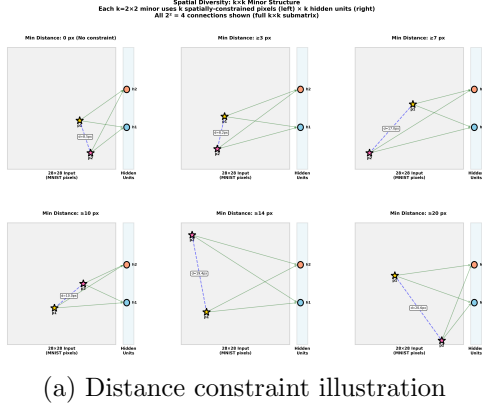


Figure 3: Spatially distributed KAG across training procedures. (a) Illustration of minimum distance constraint for  $k = 2$  pixel pairs. Colored lines connect sampled pairs with distance labels. As minimum distance increases from 0px to 14px, fewer valid pairs exist, but those that do remain spatially distributed. (b) Both standard and augmented models maintain stable PR ratios across distance constraints. Box plots show PR ratio distributions for both  $k=2$  (top) and  $k=3$  (bottom) minors, with trend lines showing median and mean stability.

pixels with minimum distance  $D \in \{0, 3, 5, 7, 10, 14\}$  pixels (Figure 3a). Figure 3b shows PR ratios remain stable even at 14 pixels apart (half the image’s linear dimension) for both models and both  $k = 2$  and  $k = 3$  minors, demonstrating KAG extends across the full input space regardless of training procedure.

## 4 Discussion

While our study is fundamentally observational, the patterns we observe connect to several strands of recent work on neural network structure. KAN networks [Liu et al., 2024] explicitly engineer Kolmogorov-Arnold representations using learnable univariate activations, treating KA structure as an architectural choice. The fact that vanilla MLPs spontaneously develop similar geometry during training suggests something different: perhaps KA structure is less about architectural design and more about what gradient descent naturally discovers when learning complex functions. This echoes the broader theme in deep learning of learned representations often matching or exceeding bespoke biases when sufficient data and capacity are available.

An intriguing observation emerges when comparing models trained with and without spatial augmentation. While both develop scale-agnostic KAG, the augmented model exhibits approximately 30-40% lower participation ratios despite maintaining comparable classification accuracy (96% vs 98%). This aligns with the notion of geometric ‘frustration’ where models trained on limited data variations (centered digits only) develop more extreme geometric structure as they struggle to learn from constrained distributions. Augmented training exposes the network to diverse spatial configurations, apparently reducing the need for such extreme geometric organization. This connects to frustration experiments from Freedman and Mulligan [2025] showing that networks trained on random or blank data develop extremely high PR values as they attempt to extract patterns from noise. The relationship between training distribution diversity and geometric structure warrants systematic investigation.

The relationship to neural collapse [Papayan et al., 2020] is particularly intriguing. Neural collapse describes how class-level representations become increasingly structured late in training, with

class means collapsing toward symmetric simplex configurations. Our work operates at a different scale: we examine the Jacobian mapping from inputs to hidden representations rather than final-layer class structure. It’s tempting to speculate that these phenomena might be connected. Where KAG in early layers provides the structured substrate that enables neural collapse in later layers, or conversely, that the pull toward collapsed final representations induces geometric organization earlier in the network. But these remain open questions requiring careful analysis of the joint evolution of geometry across all layers during training.

Of course, our analysis has certain limitations. We study only MNIST, a relatively simple grayscale recognition task, though this does represent a substantial step up in complexity from the 3D synthetic functions analyzed in prior work [Freedman and Mulligan, 2025]. Our networks are shallow (just two hidden layers) and whether these patterns persist in modern deep architectures remains to be tested. Further, while the RR metric shows clear signal, it exhibits higher baseline variance than participation ratio due to max-sampling effects, making it better suited for secondary confirmation than primary claims.

This points to clear next steps. The most critical is moving beyond correlation to causation. Frustration experiments (training on random labels before fine-tuning on real data) could test whether pre-established KAG accelerates learning. Explicit regularization experiments, adding loss terms that encourage or discourage KA signatures, could probe whether geometric structure actively aids generalization or merely accompanies it. These interventions would address the core question: Does KAG cause effective learning, or does learning cause KAG, or both?

Beyond causality, extending this analysis to modern architectures would clarify whether KAG is specific to fully-connected networks or a more universal principle. Comprehensive analysis across CNNs, ResNets, and attention-based models is needed to determine whether these geometric patterns are architecture-specific or general. Scaling to larger datasets (ImageNet for vision, language modeling datasets for transformers) would test whether these geometric patterns persist in contemporary settings or remain an artifact of simple tasks. Theoretical work explaining why gradient descent discovers KA structure could connect to broader questions about implicit regularization and the geometry of loss landscapes.

Ultimately, these results suggest that the geometric structure of learned representations may be richer and more systematic than previously appreciated. The spontaneous emergence of Kolmogorov-Arnold patterns, their consistency across spatial scales, and their correlation with network behavior all point toward geometric organization as a potentially important lens for understanding how neural networks learn. Whether this geometry is merely an interesting mathematical curiosity or plays a functional role in learning and generalization is an exciting next step.

## 5 Conclusion

We have extended the analysis of Kolmogorov-Arnold geometry from synthetic 3D functions to realistic 784-dimensional MNIST classification, revealing that this geometric structure emerges spontaneously during training and exhibits rich organizational properties. Unlike prior work on small synthetic datasets, our experiments on a high-dimensional classification task demonstrate that KAG is fundamentally scale-agnostic: it appears consistently from local neighborhoods as small as 7 pixels to the full  $28 \times 28$  image, and extends globally across the input space even when constrained to well-separated pixels.

Perhaps most strikingly, this scale-agnostic property proves robust to training procedures. Both standard training and training with spatial augmentation produce the same qualitative geometric patterns, though augmented training yields lower PR ratio. This suggests that training distribution



diversity modulates the strength of geometric structure—networks exposed to greater data variation require less extreme geometry to achieve comparable performance.

The spontaneous emergence of scale-invariant geometric structure is particularly intriguing. It suggests that KAG may reflect fundamental properties of how gradient descent organizes representations in overparameterized networks, rather than being specific to particular training regimes or datasets. Whether this structure actively aids learning or emerges as a consequence of optimization dynamics remains an important open question that intervention experiments could address.

Looking forward, extending this analysis to modern architectures and larger-scale tasks would clarify whether scale-agnostic KAG is a general principle of learned representations or specific to fully-connected networks on simple vision tasks. Systematic investigation of how training procedures shape geometric structure could yield insights into implicit regularization and inform the design of training protocols that encourage salubrious geometric properties.

## Acknowledgements

We thank Tammie Siew and Pamela Vagata (Pebblebed) for invaluable discussions that helped clarify the conceptual foundations of this work and sharpen our presentation of the key ideas.

## References

- Vladimir I Arnold. On functions of three variables. *Doklady Akademii Nauk SSSR*, 114:679–681, 1957.
- Michael H. Freedman and Michael Mulligan. Spontaneous kolmogorov-arnold geometry in shallow mlps. *arXiv preprint arXiv:2509.12326*, 2025.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.
- Andrey N Kolmogorov. On the representation of continuous functions of several variables by superpositions of continuous functions of a smaller number of variables. *Doklady Akademii Nauk SSSR*, 114:953–956, 1957.
- Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*, 2024.
- Vardan Papyan, XY Han, and David L Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663, 2020.

## A Detailed Experimental Configuration

This appendix provides complete implementation details for all experiments reported in the main paper, enabling full reproducibility of our results.

## A.1 MNIST MLP Training Details

### A.1.1 Architecture Specifications

Our 2-layer MLPs follow the architecture:

$$f(x) = W_{\text{out}}\sigma(W_1\sigma(W_0x + b_0) + b_1) \quad (6)$$

where  $x \in \mathbb{R}^{784}$  is a flattened MNIST image,  $W_0 \in \mathbb{R}^{h \times 784}$  and  $W_1 \in \mathbb{R}^{h \times h}$  are weight matrices for the two hidden layers,  $\sigma$  is the GELU activation function, and  $W_{\text{out}} \in \mathbb{R}^{10 \times h}$  maps to 10 output classes.

**Initialization:** We use Kaiming He initialization [He et al., 2015] for all weight matrices:

$$W_{ij} \sim \mathcal{N}\left(0, \frac{2}{n_{\text{in}}}\right) \quad (7)$$

where  $n_{\text{in}}$  is the number of input units to the layer.

### A.1.2 Training Hyperparameters

Parameter	Value
Optimizer	AdamW
$\beta_1, \beta_2$	0.9, 0.999
$\epsilon$	$10^{-8}$
Learning rate	$3 \times 10^{-4}$
LR schedule	Cosine annealing
Batch size	256
Epochs	200
Weight decay	$10^{-4}$
Loss function	Cross-entropy

Table 1: Complete hyperparameters for MNIST MLP training.

**Data normalization:** MNIST images are normalized using dataset statistics: mean = 0.1307, std = 0.3081.

**Hardware:** Single NVIDIA H100 GPU (80GB).

**Implementation:** PyTorch 2.0 with PyTorch Lightning for training loops.

**Random seeds:** For each configuration ( $h \in \{64, 128, 256\}$ ), we train 5 independent models with random seeds 0–4.

## A.2 Jacobian Computation Details

### A.2.1 Vectorized Jacobian Calculation

For a batch of inputs  $X \in \mathbb{R}^{B \times d}$  and hidden representation function  $\phi: \mathbb{R}^d \rightarrow \mathbb{R}^h$ , we compute the Jacobian for each sample using PyTorch’s automatic differentiation:

$$J_i = \frac{\partial \phi(x_i)}{\partial x_i} \in \mathbb{R}^{h \times d}, \quad i = 1, \dots, B \quad (8)$$

We use `torch.autograd.functional.jacobian` with vectorization enabled for efficiency. The resulting tensor has shape  $(B, h, d)$ , containing  $B$  Jacobian matrices.

### A.2.2 Memory-Efficient Minor Computation

We compute the **full Jacobian matrix**  $J \in \mathbb{R}^{h \times d}$  for each layer using vectorized automatic differentiation.

However, computing all possible  $k$ -minors from the Jacobian is prohibitively expensive, as it requires selecting all  $\binom{h}{k}$  row combinations and all  $\binom{d}{k}$  column combinations. For example, MNIST L0 ( $h = 256$ ,  $d = 784$ ) has  $\binom{256}{3} \times \binom{784}{3} \approx 2.2 \times 10^{15}$  possible 3-minors.

We therefore employ random sampling of row and column combinations:

**Row and column sampling:** For  $k > 1$ , we randomly sample up to 10,000 row combinations (from  $\binom{h}{k}$  possibilities) and up to 10,000 column combinations (from  $\binom{d}{k}$  possibilities). Sampling is deterministic (seed=42) for reproducibility.

**Chunked processing:** We split the sampled combinations into chunks of 500 and process them sequentially, clearing GPU memory between chunks to avoid out-of-memory (OOM) errors.

### A.2.3 Layer-wise Jacobian Analysis

For a 2-layer MLP, we compute Jacobians at each hidden layer independently:

- **Layer 1 (L1):**  $J_1 = \frac{\partial \phi_1}{\partial x}$  where  $\phi_1(x) = \sigma(W_0x + b_0) \in \mathbb{R}^h$
- **Layer 2 (L2):**  $J_2 = \frac{\partial \phi_2}{\partial x}$  where  $\phi_2(x) = \sigma(W_1\phi_1(x) + b_1) \in \mathbb{R}^h$

## A.3 Evaluation Protocol Details

### A.3.1 Timeline Analysis

We compute all KA geometry metrics every 10 epochs. Epoch 0 corresponds to random initialization and serves as the baseline.

### A.3.2 Random Initialization Baseline

For each trained model, we save a copy of the randomly initialized network (before any training) and compute the same Jacobian metrics. This allows us to measure the *change* in KA signatures due to training rather than properties of the initialization scheme.

### A.3.3 Statistical Robustness

We report mean  $\pm$  standard deviation across 5 random seeds for each configuration ( $h \in \{64, 128, 256\}$ ).

### A.3.4 Sampling Parameters for Metrics

To balance statistical accuracy with computational cost, we use the following sampling strategies:

## A.4 Computational Resources

Each seed run takes approximately 2–3 hours on a single H100 GPU (including training and Jacobian analysis).

## A.5 Software Versions

<b>Parameter</b>	<b>Value</b>
Evaluation batches	2 (512 samples)
Rotation samples	400
Row samples (for minors)	10,000
Column samples (for minors)	10,000
Max minor order $k$	3

Table 2: Sampling parameters for computing KA geometry metrics. Note: We compute the *full* Jacobian matrix but sample row and column combinations when computing minors (determinants).

<b>Package</b>	<b>Version</b>
Python	3.10
PyTorch	2.0.1
PyTorch Lightning	2.0.0
torchvision	0.15.2
CUDA	12.1

Table 3: Software versions used for all experiments.