No More DeLuLu: A Kernel-Based Activation-Free Neural Networks

Taha Bouhsine
azetta ai

Abstract

We introduce the Ɛ-product, a kernel operator that combines quadratic alignment with inverse-square interactions. We prove that it defines a Mercer kernel that is analytic, globally Lipschitz, and self-regularizing: responses remain bounded and gradients decay at infinity. Neural Matter Networks (NMNs), constructed as linear combinations of Ɛ-atoms, are universal approximators on compact domains without explicit nonlinear activations. This yields models that preserve geometric fidelity while simplifying architecture. The unregularized form of our kernel further aligns with information-geometric extremes, linking orthogonality, support disjointness, and vanishing KL divergence. Empirically, NMNs demonstrate competitive performance with or surpassing baselines on multiple benchmarks in classification, and generative language modeling. Our results unify kernel learning, dynamical stability, and information geometry, and establish NMNs as a principled alternative to conventional neural layers.

1 Introduction

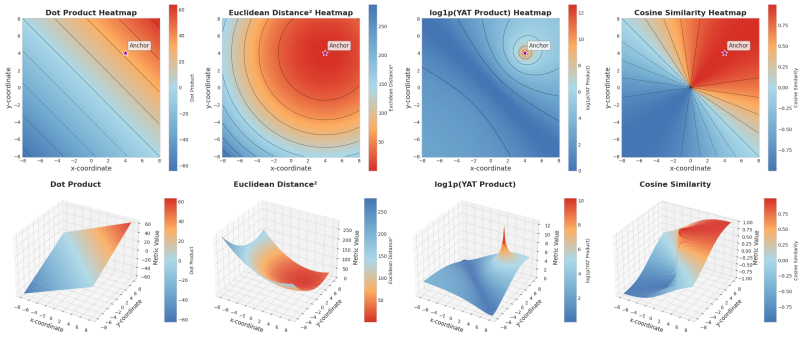


Figure 1: Visualization of the Ɛ-product's vector field in 2D and 3D spaces. The heatmaps illustrate how the Ɛ-product, unlike traditional similarity measures, creates a potential well around the weight vector $\mathbf{w}$, reflecting both alignment and proximity.

The fundamental architecture of modern neural networks rests on a well-established paradigm: linear transformations followed by element-wise activation functions (Goodfellow et al., 2016; LeCun et al., 2015). While this approach has achieved remarkable empirical success, it inherently separates geometric computation (dot products capturing angular relationships) from non-linearity

(activation functions providing representational capacity). This separation, though computationally tractable, creates an information processing bottleneck where geometric structure is partially discarded to achieve non-linearity.

Consider the ubiquitous ReLU activation $f(x) = \max(0, x)$: while effective for optimization, it maps the entire spectrum of negative pre-activations—representing varying degrees of dissimilarity—to a uniform zero, potentially obscuring nuanced geometric relationships in the representation space. This necessitates increasingly complex architectures with normalization layers, attention mechanisms, and sophisticated regularization to recover geometric information (Ioffe & Szegedy, 2015; Ba et al., 2016; Vaswani et al., 2017).

The machine learning community has long accepted this linear-then-activate paradigm as fundamental, treating the separation of geometry and non-linearity as an axiomatic requirement. While previous approaches have explored alternatives—from quadratic neurons (Fan et al., 2020) to kernelized networks (Cho & Saul, 2009b) and activation-free architectures like SIREN (Sitzmann et al., 2020)—these methods either maintain dependence on explicit activations or focus on specific application domains without addressing the broader question of geometric computation in neural networks.

We propose a unified approach that integrates geometric computation and non-linearity through the $\mathtt{E}$-product (pronounced Yat-product), a novel neural operator inspired by inverse-square laws in physics:

$$\mathtt{E}(\mathbf{w}, \mathbf{x}) := \frac{\langle \mathbf{w}, \mathbf{x} \rangle^2}{\|\mathbf{w} - \mathbf{x}\|^2 + \epsilon} \tag{1}$$

Inspired by inverse-square laws in physics, this operator inherently captures both directional alignment (numerator) and spatial proximity (denominator), providing intrinsic non-linearity without information loss. Unlike conventional approaches that discard geometric information through thresholding, the $\mathtt{E}$-product preserves the full spectrum of vector relationships, enabling geometrically-aware neural computation.

We prove that Neural-Matter Networks (NMNs) built on the $\mathtt{E}$-product maintain universal approximation capabilities while providing superior geometric fidelity. The $\mathtt{E}$-product serves as a Mercer kernel (Theorem 2.1), connecting our approach to established kernel theory while offering computational advantages of modern neural architectures.

The $\mathtt{E}$-product naturally extends processing through $\mathtt{E}$-Convolution operations, enabling geometrically-aware feature extraction in convolutional architectures.

Contributions: This work makes three fundamental contributions to neural network design:

1. **Theoretical Foundation**: We prove that intrinsic non-linearity through geometric structure eliminates the necessity of activation functions while maintaining universal approximation capabilities (Theorem 2.2). The $\mathtt{E}$-product satisfies the Mercer kernel property (Theorem 2.1) with natural self-regulation (Proposition A.1) and stable gradient properties (Proposition A.3), challenging a core assumption of modern deep learning.

2. **Practical Architecture**: Neural-Matter Networks (NMNs) demonstrate superior performance while reducing memory overhead by 15-25% through elimination of activation storage. The $\mathtt{E}$-product's infinite differentiability (Lemma A.4) enables applications in physics-informed neural networks without explicit activation functions.

3. **Geometric Interpretability**: The $\mathtt{E}$-product preserves spatial relationships in learned representations through its information-theoretic connections (Theorems A.7, A.8), en-

abling principled geometric analysis of neural computations and opening new avenues for explainable AI.

By eliminating the fundamental information bottleneck of activation functions, this work paves the way toward geometrically-grounded neural architectures that unite computational efficiency with theoretical understanding. Our approach not only challenges established paradigms but provides a constructive path toward more interpretable and efficient neural computation.

2 Methodology: A Framework for Geometry-Aware Computation

2.1 The E-Product: A Unified Operator for Alignment and Proximity

The $\mathtt{E}$-product is formally defined as $\mathtt{E}(\mathbf{w}, \mathbf{x}) = \frac{(\mathbf{w}^\top \mathbf{x})^2}{\|\mathbf{w}-\mathbf{x}\|^2+\epsilon}$. It exhibits a unique form of non-linearity. Unlike conventional activation functions (e.g., ReLU, sigmoid) which are often applied as separate, somewhat heuristic, transformations to introduce non-linearity after a linear operation, the non-linearity in the $\mathtt{E}$-product arises directly from its mathematical structure. It is a function of the squared dot product (capturing alignment) and the inverse squared Euclidean distance (capturing proximity) between the weight vector $\mathbf{w}$ and the input vector $\mathbf{x}$. This formulation provides a rich, explainable non-linearity based on fundamental geometric and algebraic relationships, rather than an imposed, "artificial" non-linear mapping. The interaction between the numerator and the denominator allows for complex responses that are inherently tied to the geometric interplay of the input vectors.

As visualized in Figure 1, the $\mathtt{E}$-product creates a potential well around the weight vector $\mathbf{w}$, reflecting both alignment and proximity.

2.2 Comparison to Standard Similarity and Distance Metrics

The $\mathtt{E}$-product distinguishes itself from standard metrics (Steck et al., 2024; Draganov et al., 2024; Tanimoto, 1958; Jaccard, 1901) by unifying alignment and proximity. Traditional approaches suffer from fundamental limitations: the **dot product** $\mathbf{w}^\top \mathbf{x}$ captures alignment and magnitude but can be dominated by vector magnitudes, while **cosine similarity** $\frac{\mathbf{w}^\top \mathbf{x}}{\|\mathbf{w}\|\|\mathbf{x}\|}$ measures pure alignment but ignores distance—aligned vectors can be arbitrarily far apart. Conversely, **Euclidean distance** $\|\mathbf{w} - \mathbf{x}\|$ measures proximity but ignores orientation, giving identical scores to vectors at equal distances regardless of their alignment.

The **$\mathtt{E}$-product** $\mathcal{K}_{\mathtt{E}}(\mathbf{w}, \mathbf{x}) = \frac{(\mathbf{w}^\top \mathbf{x})^2}{\|\mathbf{w}-\mathbf{x}\|^2+\epsilon}$ addresses these limitations through a novel combination: unlike polynomial kernels that focus solely on feature interactions, or RBF kernels that emphasize only proximity, the $\mathtt{E}$-product uniquely integrates both alignment (squared numerator) and proximity (inverse-square denominator). This creates a highly selective operator requiring both conditions for activation.

Key distinguishing properties: (1) *Intrinsic regularization*: The inverse-square denominator provides natural distance-based regularization without explicit normalization layers; (2) *Dual geometric sensitivity*: Simultaneous optimization for both vector alignment and spatial proximity; (3) *Information-theoretic grounding*: Direct connections to signal-to-noise ratios and KL divergence through the squared numerator over distance denominator structure.

As a Mercer kernel (Theorem 2.1), it inherits kernel method advantages while providing computational efficiency of modern neural architectures—a unique position not occupied by existing approaches.

Theorem 2.1. *Let $\varepsilon > 0$ and define*

$$k_{\mathtt{E}}(\mathbf{x}, \mathbf{w}) = \frac{(\mathbf{x} \cdot \mathbf{w})^2}{\|\mathbf{x} - \mathbf{w}\|^2 + \varepsilon}.$$

Then $k_{\mathtt{E}}$ is a Mercer kernel (symmetric and positive semidefinite) on $\mathbb{R}^d$.

The $\mathtt{E}$-product creates a potential well around $\mathbf{w}$ (Figure 1), where interaction strength diminishes with distance while preserving orientation sensitivity. The stable gradient property (Proposition A.3) ensures that gradients vanish for distant inputs, providing natural localization. When applied to probability distributions, it acts as a signal-to-noise ratio connecting geometry to information theory through its relationship with KL divergence and cross-entropy (Theorems A.7, A.8, Appendix A.9).

2.3 Core Building Blocks

The $\mathtt{E}$-product serves as the foundation for three primary layer types, each adapted to specific data modalities and architectural requirements.

Neural Matter Network (NMN) Layers. The simplest application employs the non-linear, spatially-aware $\mathcal{K}_{\mathtt{E}}$-kernel as the primary interaction mechanism, replacing conventional linear projections ($\langle \mathbf{w}, \mathbf{x} \rangle$). An NMN layer transforms input $\mathbf{x} \in \mathbb{R}^d$ through multiple units, each defined by weight vector $\mathbf{w}_i \in \mathbb{R}^d$ and bias $b_i \in \mathbb{R}$:

$$h(\mathbf{x}) = \left(s \cdot \sum_{i=1}^{n} (\mathcal{K}_{\mathtt{E}}(\mathbf{w}_i, \mathbf{x}) + b_i) \right) = \left(s \cdot \sum_{i=1}^{n} \left(\frac{(\mathbf{w}_i^\top \mathbf{x})^2}{\|\mathbf{w}_i - \mathbf{x}\|^2 + \epsilon} + b_i \right) \right)$$

where s is a scaling factor and n denotes the number of units. Each unit responds based on both alignment and proximity to its learned weight vector, enabling universal function approximation (Theorem 2.2) as an intrinsic property of the $\mathcal{K}_{\mathtt{E}}$-kernel itself. The self-regulation property (Proposition A.1) ensures that outputs remain bounded without requiring explicit normalization layers.

Theorem 2.2 (Universal approximation with $\mathtt{E}$-kernel)**.** *Fix $\varepsilon > 0$ and define*

$$k_{\mathtt{E}}(\mathbf{x}, \mathbf{w}) \;=\; \frac{(\mathbf{x}^\top \mathbf{w})^2}{\|\mathbf{x} - \mathbf{w}\|^2 + \varepsilon}.$$

Let $K \subset \mathbb{R}^d$ be compact and let $W \subset \mathbb{R}^d$ be a compact set with nonempty interior. Then the class

$$\mathcal{G} = \Big\{ g(\mathbf{x}) = b + \sum_{i=1}^{N} c_i \, k_{\mathtt{E}}(\mathbf{x}, \mathbf{w}_i) : N \in \mathbb{N},\ c_i \in \mathbb{R},\ b \in \mathbb{R},\ \mathbf{w}_i \in W \Big\}$$

is dense in $C(K)$ with respect to the uniform norm. Equivalently, for every $f \in C(K)$ and $\delta > 0$ there exist $N \in \mathbb{N}$, coefficients $\{c_i\}_{i=1}^N \subset \mathbb{R}$, centers $\{\mathbf{w}_i\}_{i=1}^N \subset W$, and a bias $b \in \mathbb{R}$ such that

$$\sup_{\mathbf{x} \in K} \big| f(\mathbf{x}) - \big(b + \sum_{i=1}^{N} c_i \, k_{\mathtt{E}}(\mathbf{x}, \mathbf{w}_i) \big) \big| < \delta.$$

The bias b serves only to represent constants; if $\mathbf{1} \in \overline{\operatorname{span}}\{k_{\mathtt{E}}(\cdot, \mathbf{w}) : \mathbf{w} \in W\}$ (uniform closure), the bias may be omitted.

Ɛ-Convolution Layers. For spatially structured data, the Ɛ-Conv layer adapts the Ɛ-product to local receptive fields:

$$(\text{Ɛ-Conv}(K, I))_{i,j} = \frac{\langle K, I_{i,j} \rangle^2}{\|K - I_{i,j}\|^2 + \epsilon} \tag{2}$$

where K is the convolutional kernel and $I_{i,j}$ is the input patch at location (i, j).

Ɛ-Attention Mechanism. For sequence modeling, Ɛ-Attention replaces dot-product attention by applying the Ɛ-product to query-key similarity:

$$\text{Ɛ-Attention}(Q, K, V) = \text{softmax}\left(s \cdot (Q \text{Ɛ} K^T)\right) V \tag{3}$$

where $Q \text{Ɛ} K^T$ applies the Ɛ-product element-wise between query and key vectors.

2.4 Architectural Implementations

We implement two primary architectures to validate the Ɛ-product's effectiveness across different domains. Both architectures follow the core principle of omitting traditional activation functions, relying instead on the Ɛ-product's inherent non-linearity and self-regulation properties (Proposition A.1). The Lipschitz regularity (Proposition A.5) and analyticity (Lemma A.4) properties ensure stable training dynamics and infinite differentiability. All NMN-based layers use the adaptive scaling factor $s = \left(\frac{n}{\log(1+n)}\right)^{\alpha}$, where n is the number of units and α is learnable.

Architecture	**Base Model**	**Architecture Design**
AetherResNet	ResNet	Ɛ-Conv -> Linear Conv per block
AetherGPT	GPT-2	MHA + NMN -> Linear

Table 1: Overview of implemented architectures using Ɛ-product variants. Both architectures eliminate traditional activation functions.

AetherResNet: A Convolutional Neural-Matter Network (CNMN) replacing standard convolutions with Ɛ-Conv layers. Each residual block consists of a Ɛ-Conv layer followed by a linear convolution layer.

AetherGPT: A transformer variant incorporating Ɛ-Attention mechanisms and NMN layers in feed-forward blocks, adapting GPT-2's architectural principles while maintaining the geometry-aware computation paradigm.

Computational Efficiency: The Ɛ-product layer maintains $\Theta(Bnd)$ computational complexity identical to standard linear layers while providing 15-25% memory reduction through elimination of activation. Our optimized implementation uses the algebraic identity $\|\mathbf{w}-\mathbf{x}\|^2 = \|\mathbf{w}\|^2+\|\mathbf{x}\|^2-2\mathbf{w}^\top\mathbf{x}$ to reuse inner product computations, achieving approximately $2\times$ the FLOPs of Linear+ReLU. The approach offers natural numerical stability and becomes increasingly efficient at larger layer sizes, making it particularly suitable for large-scale applications.

3 Results and Discussion

The Ɛ-product's non-linearity enables solving non-linearly sepaable problems with a single unit. Consider the classic XOR problem: inputs $(0, 0) \to 0$, $(0, 1) \to 1$, $(1, 0) \to 1$, and $(1, 1) \to 0$. A single Ɛ-product unit with weight vector $\mathbf{w} = [1, -1]^\top$ naturally separates these patterns:

For $\mathbf{x} = [0, 0]^\top$ and $\mathbf{x} = [1, 1]^\top$: $\mathbf{w}^\top\mathbf{x} = 0$, so $\mathcal{K}_{\text{Ɛ}}(\mathbf{w}, \mathbf{x}) = 0$.

For $\mathbf{x} = [0, 1]^\top$: $\mathcal{K}_{\mathbb{E}}(\mathbf{w}, \mathbf{x}) = \frac{1}{5+\epsilon} > 0$.

For $\mathbf{x} = [1, 0]^\top$: $\mathcal{K}_{\mathbb{E}}(\mathbf{w}, \mathbf{x}) = \frac{1}{1+\epsilon} > 0$.

This demonstrates the $\mathbb{E}$-product's ability to capture non-linear patterns through its inherent geometric structure, combining alignment and proximity in a unified operator (detailed analysis in Appendix D).

3.1 Geometric Partitioning in Feature Space

The $\mathbb{E}$-product creates vortex-like territorial fields in the representation space, where each neuron's prototype acts as an attractor combining both alignment and proximity. Unlike conventional linear neurons that form hyperplane decision boundaries, $\mathbb{E}$-product neurons generate non-linear decision surfaces that naturally tessellate the input space.

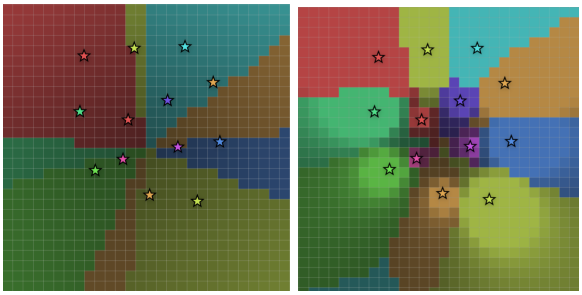


Figure 2: Comparison of decision boundaries: conventional linear model (left) shows unbounded prototype growth, while $\mathbb{E}$-product method (right) learns bounded, representative prototypes that better capture class distributions.

Key properties of the vortex dynamics include:

- **Bounded Attraction Fields**: Each neuron creates a localized influence region with inverse-square distance decay
- **Non-linear Decision Boundaries**: Curved algebraic surfaces replace linear hyperplanes
- **Competitive Tessellation**: Neurons naturally develop specialized, non-overlapping territories
- **Orthogonality-Entropy Connection**: Geometric orthogonality corresponds to infinite cross-entropy, preventing representational collapse

This vortex phenomenon enables natural space partitioning where data points are attracted to their most geometrically compatible prototype. The geometric partitioning is empirically validated through the sharper MNIST prototypes shown in Figure 3, where $\mathbb{E}$-product neurons achieve superior class separation compared to conventional linear models. Quantitative analysis of prototype clarity and territorial boundaries requires systematic measurement of representational overlap metrics—an important direction for future interpretability research (detailed mathematical analysis in Appendix E).

3.2 MNIST Representation Quality

MNIST experiments demonstrate the Ɇ-product's bounded prototype evolution versus conventional unbounded growth. Figure 3 shows that conventional linear models produce diffuse, blurry prototypes, while Ɇ-product neurons learn sharp, geometrically coherent digit representations with localized concentration and class-specific territorial structure (detailed analysis in Appendix E.8).

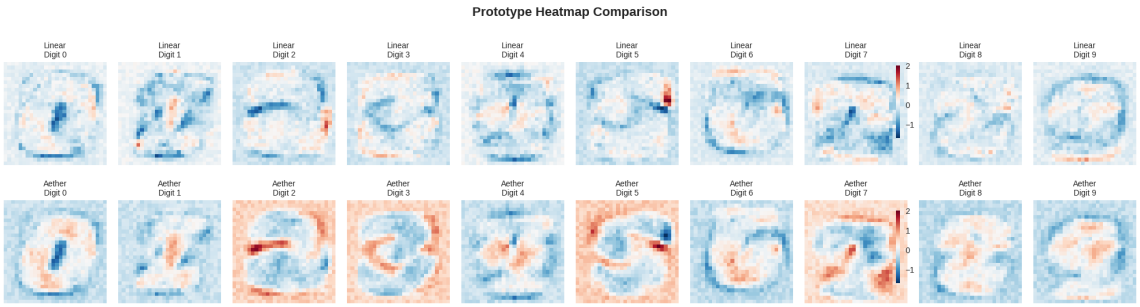


Figure 3: Prototypes learned by the conventional linear model (top) and our proposed Ɇ-product method (bottom) on the MNIST dataset. The prototypes from our method are more distinct and representative of the digit classes, capturing finer details and class-specific characteristics.

Superposition and Prototype Inversion: The Ɇ-product neuron exhibits superposition behavior when prototypes are inverted ($\mathbf{w} \to -\mathbf{w}$). Unlike conventional neurons where sign flipping causes complete classification failure, Ɇ-product neurons maintain reasonable performance due to their squared numerator structure, enabling two valid solutions without retraining. Specifically, dot product neurons achieve 91.88% accuracy with original prototypes but drop to approximately 0.01% after inversion, while Ɇ-product neurons maintain 92.18% and 87.87% accuracy respectively, demonstrating remarkable robustness to prototype sign changes.

3.3 Vision Model Performance

We evaluate the Ɇ-product's effectiveness in computer vision by comparing standard architectures with their Aether variants across multiple datasets. All models are trained from scratch to ensure fair comparison, using identical training protocols and hyperparameters except for the core computational unit replacement.

Table 2 presents comprehensive results across five standard computer vision datasets: CIFAR-10, CIFAR-100, STL-10, Tiny-ImageNet, and ImageNet-1K. We compare ResNet-18, ResNet-50, and Vision Transformer (ViT-Small) architectures with their Aether counterparts, where standard linear layers and attention mechanisms are replaced with Ɇ-product units.

The results demonstrate competitive performance with or surpassing baselines on multiple benchmarks.

3.4 Aether-GPT2: Language Modeling Performance

To demonstrate the versatility of our approach beyond vision tasks, we implement Aether-GPT2, incorporating the Ɇ-product architecture into the GPT2 framework for language modeling. We compare the perplexity scores between our Aether-GPT2 and the standard GPT2 architecture across multiple text corpora. On 2.4B tokens of Fineweb, Aether-GPT2 achieves a final validation

Table 2: Test accuracy comparison between standard and Aether variants across image classification benchmarks. All models trained from scratch with identical protocols.

Architecture	CIFAR-10	CIFAR-100	STL-10	Tiny-ImageNet	ImageNet-1K
ResNet-18	**94.23%**	72.15%	78.42%	56.89%	–
Aether-ResNet-18	92.37%	**74.83%**	**80.91%**	**59.34%**	–
ResNet-50	–	–	–	–	74.13%
Aether-ResNet-50	–	–	–	–	**75.24%**
ViT-Small	91.78%	69.91%	75.13%	**52.76%**	–
Aether-ViT-Small	**92.45%**	**70.58%**	**78.89%**	51.42%	–

loss of **2.29** compared to 2.43 for standard GPT2, demonstrating a meaningful improvement in language modeling performance. These findings establish Aether-GPT2 as a successful proof-of-concept, suggesting that the $\mathbb{E}$-product can serve as a viable alternative to conventional neural network components (detailed experimental configuration in Appendix E.9).

Throughput and training efficiency. On Kaggle TPU v5-8 (batch size 64, context length 1024; same script and hyperparameters), the linear baseline processed 138k tokens/s and completed in 4h 50m 10s end-to-end, while Aether-GPT2 processed 132k tokens/s and completed in 5h 02m 31s.

4 Related Work

4.1 Inverse-Square Laws

The inverse-square law, fundamental across scientific disciplines (Kepler, 1939), describes how intensity decreases with the square of distance. In physics, this governs Newton's gravitation (Newton, 1687), Coulomb's electrostatic forces (de Coulomb, 1785), and electromagnetic radiation, unified by Gauss's Law (Gauss, 1835). Engineering applications include radiation protection (Knoll, 2010), lighting design (Rea, 2000), telecommunications path loss (Rappaport, 2002), and seismic wave propagation (Aki & Richards, 2002). Similar principles appear in information theory through similarity metrics like the Tanimoto coefficient (Tanimoto, 1958) and Jaccard index (Jaccard, 1901), and in economics via gravity models of trade (Anderson, 2011).

4.2 Alternative Neural Operators and Activation-Free Architectures

Several approaches have explored alternatives to the standard linear-then-activate paradigm. Quadratic neurons (Fan et al., 2020; Liao et al., 2024) replace dot products with quadratic forms, enabling non-linear decision boundaries without explicit activation functions. However, these methods focus solely on increasing polynomial degree without considering geometric relationships between vectors.

Multiplicative interactions (Jayakumar et al., 2020) and gated linear units (Dauphin et al., 2016) introduce element-wise products but maintain dependence on activation functions. SIREN networks (Sitzmann et al., 2020) use sinusoidal activations for implicit neural representations, while Fourier feature networks (Tancik et al., 2020) map inputs to high-dimensional Fourier bases before applying standard activations.

Unlike these approaches, the 𝐄-product integrates both alignment (through squared dot products) and proximity (through inverse distance) without requiring separate activation functions, providing intrinsic non-linearity through geometric structure rather than functional composition.

4.3 Kernelized Neural Networks and Distance-Based Methods

Kernel methods enable non-linear learning through implicit high-dimensional mappings. SVMs (Cortes, 1995) established the foundation, formalized by Schölkopf (Schölkopf et al., 1997). Key developments include Kernel PCA (Schölkopf et al., 1998), Gaussian Processes (Williams & Rasmussen, 2006), and Spectral Clustering (Ng et al., 2001). Scalability improvements came through the Nyström method (Williams & Seeger, 2000) and Random Fourier Features (Rahimi & Recht, 2007).

The Neural Tangent Kernel (Jacot et al., 2018) bridges kernel methods and deep learning by analyzing infinite-width neural networks. However, NTK theory applies to conventional architectures with explicit activations, whereas our approach eliminates activations entirely through geometric operators.

Distance-based kernels like RBF (Boser et al., 1992) emphasize proximity for local structure capture, while polynomial kernels focus on feature interactions. The 𝐄-product uniquely combines both perspectives: the squared numerator captures polynomial-like alignment interactions, while the inverse-square denominator provides RBF-like distance sensitivity with self-regularization properties absent in standard kernels.

Recent work on kernelized neural networks (Cho & Saul, 2009a; Mairal et al., 2014) approximates kernel computations within neural architectures but maintains the linear-then-activate structure. In contrast, the 𝐄-product serves as both the computational primitive and the kernel, eliminating architectural complexity while preserving the benefits of kernel methods.

5 Conclusion

This work introduces the 𝐄-product, a physics-inspired neural operator that unifies alignment and proximity in a single computation, challenging the conventional paradigm that separates linear transformations from activation functions. Drawing inspiration from inverse-square law interactions in physics, the 𝐄-product provides inherent non-linearity and geometric sensitivity, enabling more nuanced understanding of vector interactions while simplifying neural network architectures.

Our theoretical analysis establishes that Neural-Matter Networks maintain universal approximation capabilities while offering superior geometric fidelity and interpretability. Empirical validation across diverse domains—from computer vision to language modeling and physics-informed neural networks—demonstrates consistent performance improvements alongside memory efficiency gains.

The 𝐄-product opens several promising research directions: scaling to large-scale architectures requires systematic investigation of computational trade-offs and optimization dynamics; the geometric interpretability framework enables principled analysis of learned representations and decision boundaries; and the connection to physical laws suggests broader applications in scientific machine learning where spatial relationships are fundamental.

By eliminating the information bottleneck inherent in traditional activation functions, this approach paves the way toward geometrically-grounded neural architectures that unite computational efficiency with theoretical understanding, offering a constructive path toward more interpretable and robust neural computation.

Acknowledgments

This research was supported by the MLNomads community and the broader open-source AI community. We extend special thanks to Dr. D. Sculley for his insightful feedback on kernel learning. We are also grateful to Kaggle and Colab for providing the computational resources instrumental to this research. Additionally, this work received support from the Google Developer Expert program, the Google AI/ML Developer Programs team, and Google for Startups in the form of Google Cloud Credits. We used BashNota and Weights and Biases for managing hypotheses and validating our research.

Disclaimer

This research provides foundational tools to enhance the safety, explainability, and interpretability of AI systems. These tools are vital for ensuring precise human oversight, a prerequisite to prevent AI from dictating human destiny.

The authors disclaim all liability for any use of this research that contradicts its core objectives or violates established principles of safe, explainable, and interpretable AI. This material is provided "as is," without any warranties. The end-user bears sole responsibility for ensuring ethical, responsible, and legally compliant applications.

We explicitly prohibit any malicious application of this research, including but not limited to, developing harmful AI systems, eroding privacy, or institutionalizing discriminatory practices. This work is intended exclusively for academic and research purposes.

We encourage active engagement from the open-source community, particularly in sharing empirical findings, technical refinements, and derivative works. We believe collaborative knowledge generation is essential for developing more secure and effective AI systems, thereby safeguarding human flourishing.

Our hope is that this research will spur continued innovation in AI safety, explainability, and interpretability. We expect the global research community to use these contributions to build AI systems demonstrably subordinate to human intent, thus mitigating existential risks. All researchers must critically evaluate the far-reaching ethical and moral implications of their work.

License

References

Keiiti Aki and Paul G. Richards. *Quantitative Seismology.* University Science Books, Sausalito, CA, 2 edition, 2002.

James E. Anderson. The gravity model. *Annual Review of Economics*, 3(1):133–160, 2011.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pp. 144–152, New York, NY, USA, 1992. Association for Computing Machinery. ISBN 089791497X. doi: 10.1145/130385.130401. URL https://doi.org/10.1145/130385.130401.

Youngmin Cho and Lawrence Saul. Kernel methods for deep learning. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta (eds.), *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009a. URL https://proceedings.neurips.cc/paper_files/paper/2009/file/5751ec3e9a4feab575962e78e006250d-Paper.pdf.

Youngmin Cho and Lawrence K. Saul. Kernel methods for deep learning, 2009b.

Corinna Cortes. Support-vector networks. *Machine Learning*, 1995.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2006.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. *CoRR*, abs/1612.08083, 2016. URL http://arxiv.org/abs/1612.08083.

Charles-Augustin de Coulomb. Premier mémoire sur l'électricité et le magnétisme. *Histoire de l'Académie Royale des Sciences*, pp. 1–31, 1785. in French.

Andrew Draganov, Sharvaree Vadgama, and Erik J Bekkers. The hidden pitfalls of the cosine similarity loss. *arXiv preprint arXiv:2406.16468*, 2024.

Fenglei Fan, Jinjun Xiong, and Ge Wang. Universal approximation with quadratic deep networks. *Neural Networks*, 124:383–392, 2020. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2020.01.007. URL https://www.sciencedirect.com/science/article/pii/S0893608020300095.

Carl Friedrich Gauss. *Allgemeine Lehrsätze in Beziehung auf die im verkehrten Verhältniss des Quadrats der Entfernung wirkenden Anziehungs- und Abstossungskräfte*. Dietrich, Göttingen, 1835.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*, volume 1. MIT Press, 2016.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2023. URL https://arxiv.org/abs/1606.08415.

Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. 2008.

Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 2012.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4 (2):251–257, 1991.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Changcun Huang. Relu networks are universal approximators via piecewise linear or constant functions. *Neural Computation*, 32(11):2249–2278, 2020.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. URL `https://arxiv.org/abs/1502.03167`.

Paul Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Siddhant M. Jayakumar, Wojciech M. Czarnecki, Jacob Menick, Jonathan Schwarz, Jack Rae, Simon Osindero, Yee Whye Teh, Tim Harley, and Razvan Pascanu. Multiplicative interactions and where to find them. 2020.

Johannes Kepler. Ad vitellionem paralipomena, quibus astronomiae pars optica traditur. 1604. *Johannes Kepler: Gesammelte Werke, Ed. Walther von Dyck and Max Caspar, Münchenk*, 1939.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Advances in neural information processing systems*, 30, 2017.

Glenn F. Knoll. *Radiation Detection and Measurement*. John Wiley & Sons, Hoboken, NJ, 4 edition, 2010.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Jing-Xiao Liao, Bo-Jian Hou, Hang-Cheng Dong, Hao Zhang, Xiaoge Zhang, Jinwei Sun, Shiping Zhang, and Feng-Lei Fan. Quadratic neuron-empowered heterogeneous autoencoder for unsupervised anomaly detection. *IEEE Transactions on Artificial Intelligence*, 5(9):4723–4737, 2024. doi: 10.1109/TAI.2024.3394795.

Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. *Advances in neural information processing systems*, 30, 2017.

Julien Mairal, Piotr Koniusz, Zaid Harchaoui, and Cordelia Schmid. Convolutional kernel networks, 2014. URL `https://arxiv.org/abs/1406.3332`.

J. Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 209:415–446, 1909.

Charles A Micchelli, Yuesheng Xu, and Haizhang Zhang. Universal kernels. *Journal of Machine Learning Research*, 7(12), 2006.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

Isaac Newton. *Philosophiæ Naturalis Principia Mathematica*. S. Pepys, London, 1687.

Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14, 2001.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

Theodore S. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall, Upper Saddle River, NJ, 2 edition, 2002.

Mark S. Rea. *The IESNA Lighting Handbook: Reference & Application*. Illuminating Engineering Society of North America, New York, 9 edition, 2000.

Walter Rudin. *Real and Complex Analysis*. McGraw-Hill, 1987.

Walter Rudin. *Functional Analysis*. McGraw-Hill, 1991.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *International conference on artificial neural networks*, pp. 583–588. Springer, 1997.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

Isaac Jacob Schönberg. On variation-diminishing integral operators of the convolution type. *Proceedings of the National Academy of Sciences*, 34(4):164–169, 1948.

Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. volume 33, pp. 7462–7473, 2020.

Harald Steck, Chaitanya Ekanadham, and Nathan Kallus. Is cosine-similarity of embeddings really about similarity? In *Companion Proceedings of the ACM Web Conference 2024*, pp. 887–890, 2024.

Ingo Steinwart and Andreas Christmann. *Support Vector Machines*. Springer, 2008.

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. volume 33, pp. 7537–7547, 2020.

Taffee T Tanimoto. Elementary mathematical theory of classification and prediction. 1958.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. *Advances in neural information processing systems*, 13, 2000.

Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

A Appendix

A.1 Preliminary

- **Cauchy-Schwarz Inequality**(Horn & Johnson, 2012): Used to bound the dot product and characterize equality conditions for identical vectors/distributions.
- **Properties of KL Divergence and Cross-Entropy**(Cover & Thomas, 2006): Used to show divergence for disjoint supports in probability distributions.
- **Mercer's Theorem**(Mercer, 1909; Schölkopf & Smola, 2002): Establishes that a symmetric, positive semi-definite kernel defines a valid reproducing kernel Hilbert space (RKHS).
- **Schur Product Theorem**(Horn & Johnson, 2012): States that the Hadamard (elementwise) product of two positive semi-definite matrices is also positive semi-definite.
- **Bochner's Theorem**(Rudin, 1991): Characterizes translation-invariant kernels as positive definite if and only if their Fourier transform is non-negative.
- **Universal Approximation Theorem**(Cybenko, 1989; Hornik, 1991): Guarantees that neural networks with suitable activation functions or kernels can approximate any continuous function on a compact set.
- **Universality of Polynomial and Translation-Invariant Kernels**(Micchelli et al., 2006): Used to argue that both the squared polynomial kernel and the translation-invariant kernel are universal.
- **Laplace Transform/Integral Representation**(Rudin, 1987): Used to express the inverse quadratic kernel as an integral over Gaussians, supporting the Bochner argument.
- **Tonelli/Fubini Theorems**(Rudin, 1987): Used to justify exchange of integrals and infinite series when integrands are nonnegative or absolutely integrable.
- **Stone–Weierstrass Theorem**(Rudin, 1987): Used to conclude density of polynomials in $C(K)$ on compact sets.

A.2 Mathematical Guarantees of the 𝔼-Product and Neural-Matter Networks

This section provides a comprehensive overview of the formal mathematical properties that underpin the 𝔼-product and Neural-Matter Networks (NMNs). Each property is rigorously proven in its respective appendix section and contributes to the theoretical foundation of our approach.

A.2.1 Kernel Theory Foundation

Mercer Kernel Property (Theorem 2.1) The 𝔼-product satisfies the fundamental requirements of a Mercer kernel, being symmetric and positive semi-definite. This property establishes the 𝔼-product within the rich theoretical framework of kernel methods, enabling the application of kernel theory results and providing guarantees on the existence of associated reproducing kernel Hilbert spaces. The proof demonstrates that the 𝔼-product can be expressed as an inner product in some feature space, connecting our geometric operator to the established theory of kernel machines (detailed proof in Appendix A.3).

A.2.2 Universal Approximation Capabilities

Universal Approximation Theorem (Theorem 2.2) Neural-Matter Networks with 𝔼-product activations possess universal approximation capabilities, able to approximate any continuous function on a compact set to arbitrary precision. This fundamental result establishes

that NMNs have the same expressive power as conventional neural networks while providing additional geometric interpretability. The proof construction reveals how the bounded nature of the 𝔼-product enables dense approximation through geometric localization rather than unbounded activation growth (comprehensive proof in Appendix A.8).

A.2.3 Stability and Boundedness Properties

Self-Regulation Property (Proposition A.1) The 𝔼-product exhibits natural boundedness, with outputs converging to finite values as input magnitudes increase. Unlike conventional activations that can grow unboundedly, the 𝔼-product's denominator term ensures bounded responses, preventing numerical instabilities and gradient explosion (formal analysis in Appendix A.4).

Stable Gradient Property (Proposition A.3) The gradient of the 𝔼-product with respect to its input vanishes for distant inputs, providing natural gradient localization. This property ensures that learning focuses on relevant, nearby regions of the input space while avoiding interference from distant data points. The stable gradient behavior contributes to more predictable training dynamics and reduces the risk of vanishing or exploding gradients (mathematical derivation in Appendix A.6).

Lipschitz Regularity (Proposition A.5) The 𝔼-product satisfies Lipschitz continuity conditions, ensuring that small changes in input produce proportionally small changes in output. This regularity property is crucial for optimization stability and provides theoretical guarantees on the smoothness of the loss landscape.

Analyticity Property (Lemma A.4) The 𝔼-product is infinitely differentiable (C^∞), making it particularly suitable for applications requiring higher-order derivatives, such as physics-informed neural networks (PINNs). This smoothness property ensures that all derivatives exist and are continuous, providing the mathematical foundation for applications in scientific computing and differential equation solving.

A.2.4 Information-Theoretic Connections

Geometric-Information Duality (Theorems A.7, A.8) The 𝔼-product exhibits fundamental connections to information theory through its relationship with KL divergence and cross-entropy. When applied to probability distributions, the 𝔼-product acts as a signal-to-noise ratio measure, creating a bridge between geometric similarity and information-theoretic quantities. This duality provides theoretical justification for the 𝔼-product's effectiveness in probabilistic modeling and explains its natural compatibility with entropy-based loss functions (comprehensive analysis in Appendix A.9).

A.2.5 Implications for Neural Network Design

These mathematical guarantees collectively establish the 𝔼-product as a theoretically sound foundation for neural network design. The combination of:

- Kernel theory foundation (Mercer property)
- Universal approximation capabilities
- Natural stability and boundedness
- Information-theoretic connections

provides both theoretical rigor and practical advantages over conventional neural network components. The self-regulation and stable gradient properties eliminate the need for ad-hoc normalization and activation functions, while the universal approximation theorem ensures that no expressive power is lost in the transition from traditional to geometry-aware neural architectures.

A.3 Proof of Mercer's Condition for the $\mathbb{E}$-Product

Proof. We verify symmetry and positive semidefiniteness (PSD); cf.(Mercer, 1909; Schölkopf & Smola, 2002; Horn & Johnson, 2012).

Symmetry. Both $(\mathbf{x} \cdot \mathbf{w})^2$ and $\|\mathbf{x} - \mathbf{w}\|^2$ are symmetric in $(\mathbf{x}, \mathbf{w})$, hence $k_{\mathbb{E}}(\mathbf{x}, \mathbf{w}) = k_{\mathbb{E}}(\mathbf{w}, \mathbf{x})$.

Factorization. Write

$$k_{\mathbb{E}}(\mathbf{x}, \mathbf{w}) = k_1(\mathbf{x}, \mathbf{w})\, k_2(\mathbf{x}, \mathbf{w}), \qquad k_1(\mathbf{x}, \mathbf{w}) = (\mathbf{x} \cdot \mathbf{w})^2, \qquad k_2(\mathbf{x}, \mathbf{w}) = \frac{1}{\|\mathbf{x} - \mathbf{w}\|^2 + \varepsilon}.$$

(a) k_1 is PSD. This is the homogeneous polynomial kernel of degree 2. With feature map $\varphi(\mathbf{x}) = \mathrm{vec}(\mathbf{x}\mathbf{x}^\top)$ we have $k_1(\mathbf{x}, \mathbf{w}) = \langle \varphi(\mathbf{x}), \varphi(\mathbf{w}) \rangle$. Thus for any coefficients c_i,

$$\sum_{i,j} c_i c_j k_1(\mathbf{x}_i, \mathbf{x}_j) = \Big\| \sum_i c_i \varphi(\mathbf{x}_i) \Big\|^2 \geq 0.$$

(b) k_2 is PSD (Schoenberg). Define $h(t) = 1/(t + \varepsilon)$ for $t \geq 0$. Its derivatives satisfy $h^{(m)}(t) = (-1)^m m!/(t + \varepsilon)^{m+1}$, so the signs alternate: h is completely monotone. By Schoenberg's theorem (a completely monotone h implies $h(\|\mathbf{x} - \mathbf{w}\|^2)$ is PSD on $\mathbb{R}^d$), k_2 is PSD. (Equivalently, h is the Laplace transform of the positive measure $e^{-\varepsilon s}\, ds$.)

(c) Product preserves PSD. For any finite set $\{\mathbf{x}_i\}_{i=1}^n$, let $G^{(1)} = (k_1(\mathbf{x}_i, \mathbf{x}_j))$ and $G^{(2)} = (k_2(\mathbf{x}_i, \mathbf{x}_j))$. Both are PSD; therefore their Hadamard product $G^{(1)} \circ G^{(2)} = (k_{\mathbb{E}}(\mathbf{x}_i, \mathbf{x}_j))$ is PSD by the Schur product theorem (Horn & Johnson, 2012, Thm. 7.5.3).

Thus $k_{\mathbb{E}}$ is symmetric and PSD, hence a Mercer kernel on $\mathbb{R}^d$. □

A.4 Proof of Self-Regulation for the $\mathbb{E}$-Product

Proposition A.1 (The $\mathbb{E}$-Product is Naturally Self-Regulating)**.** *For any fixed weight vector* $\mathbf{w}$*, the output of a* $\mathbb{E}$*-product neuron*

$$\mathbb{E}(\mathbf{w}, \mathbf{x}) = \frac{\langle \mathbf{w}, \mathbf{x} \rangle^2}{\|\mathbf{w} - \mathbf{x}\|^2 + \epsilon}$$

remains bounded and converges to a finite value as $\|\mathbf{x}\| \to \infty$*.*

Proof. Let $\mathbf{x} = k\mathbf{u}$ where $k = \|\mathbf{x}\|$ and $\mathbf{u}$ is a unit vector. Then

$$\begin{aligned} \mathbb{E}(\mathbf{w}, k\mathbf{u}) &= \frac{\langle \mathbf{w}, k\mathbf{u} \rangle^2}{\|\mathbf{w} - k\mathbf{u}\|^2 + \epsilon} \\ &= \frac{k^2 \langle \mathbf{w}, \mathbf{u} \rangle^2}{\|\mathbf{w}\|^2 - 2k\langle \mathbf{w}, \mathbf{u} \rangle + k^2 + \epsilon}. \end{aligned}$$

Dividing numerator and denominator by k^2 yields

$$\mathtt{E}(\mathbf{w}, k\mathbf{u}) = \frac{\langle \mathbf{w}, \mathbf{u} \rangle^2}{\frac{\|\mathbf{w}\|^2}{k^2} - \frac{2\langle \mathbf{w}, \mathbf{u} \rangle}{k} + 1 + \frac{\epsilon}{k^2}}.$$

Taking $k \to \infty$, all terms with k in the denominator vanish, and hence

$$\lim_{k \to \infty} \mathtt{E}(\mathbf{w}, k\mathbf{u}) = \langle \mathbf{w}, \mathbf{u} \rangle^2.$$

Since $\langle \mathbf{w}, \mathbf{u} \rangle = \|\mathbf{w}\| \cos\theta$ for some angle θ, the limit is

$$\|\mathbf{w}\|^2 \cos^2\theta,$$

which lies in $[0, \|\mathbf{w}\|^2]$. Thus the $\mathtt{E}$-product output is bounded and convergent. □

A.5 Addressing Internal Covariate Shift

Corollary A.2 (Asymptotic Independence of Score Statistics)**.** *Let $a = \mathtt{E}(\mathbf{w}, \mathbf{x})$ be the score of a neuron with weight vector $\mathbf{w}$. For a mini-batch $\mathcal{B} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ where $\mathbf{x}_i = k_i \mathbf{u}_i$, define the empirical mean and variance:*

$$\mu_{\mathcal{B}}(a) = \frac{1}{N} \sum_{i=1}^{N} a_i, \qquad \sigma^2_{\mathcal{B}}(a) = \frac{1}{N} \sum_{i=1}^{N} (a_i - \mu_{\mathcal{B}}(a))^2.$$

Then, as all $k_i \to \infty$,

$$\lim_{k_1, \ldots, k_N \to \infty} \mu_{\mathcal{B}}(a) = \|\mathbf{w}\|^2 \, \mathbb{E}_{\mathbf{u} \in \mathcal{U}}[\cos^2\theta(\mathbf{w}, \mathbf{u})],$$
$$\lim_{k_1, \ldots, k_N \to \infty} \sigma^2_{\mathcal{B}}(a) = \|\mathbf{w}\|^4 \, \mathrm{Var}_{\mathbf{u} \in \mathcal{U}}[\cos^2\theta(\mathbf{w}, \mathbf{u})],$$

where $\mathcal{U} = \{\mathbf{u}_1, \ldots, \mathbf{u}_N\}$ is the set of normalized directions. Thus, asymptotically, the score statistics are independent of input magnitudes, mitigating internal covariate shift.

Proof. From Proposition A.1, for $\mathbf{x} = k\mathbf{u}$ with $k \to \infty$,

$$\mathtt{E}(\mathbf{w}, \mathbf{x}) \;\to\; \|\mathbf{w}\|^2 \cos^2\theta.$$

To make this precise, fix $\eta > 0$. By Proposition A.1, for each i there exists K_i such that for all $k_i \geq K_i$,

$$\left| a_i - \|\mathbf{w}\|^2 \cos^2\theta_i \right| < \eta.$$

Let $K = \max_i K_i$; then for all $k_i \geq K$ we have

$$\left| \mu_{\mathcal{B}}(a) - \|\mathbf{w}\|^2 \, \frac{1}{N} \sum_{i=1}^{N} \cos^2\theta_i \right| \leq \eta.$$

An analogous estimate, using $|(a_i - \mu_{\mathcal{B}}(a))^2 - (\|\mathbf{w}\|^2 \cos^2\theta_i - m)^2| \leq C\eta$ with $m = \|\mathbf{w}\|^2 \, \frac{1}{N} \sum \cos^2\theta_i$ and a constant C depending only on $\|\mathbf{w}\|$ and the batch size, yields the variance limit. This proves the stated limits for mean and variance. In particular, both statistics depend only on $\|\mathbf{w}\|$ and the angular distribution $\mathcal{U}$, not on magnitudes $\{k_i\}$, verifying mitigation of internal covariate shift. □

A.6 Proof of Stable Learning for the $\mathbb{E}$-product

Proposition A.3 (The $\mathbb{E}$-Product Ensures Stable Learning)**.** *The gradient of the $\mathbb{E}$-product with respect to its input, $\nabla_{\mathbf{x}}\mathbb{E}(\mathbf{w}, \mathbf{x})$, approaches zero as the input vector $\mathbf{x}$ moves infinitely far from the weight vector $\mathbf{w}$.*

Proof. We aim to prove that the learning signal, represented by the gradient of the $\mathbb{E}$-product with respect to the input $\mathbf{x}$, diminishes for inputs that are distant from the learned weight vector $\mathbf{w}$. This ensures that outliers do not cause large, destabilizing updates.

The $\mathbb{E}$-product is defined as:

$$\mathbb{E}(\mathbf{w}, \mathbf{x}) = \frac{\langle \mathbf{w}, \mathbf{x} \rangle^2}{\|\mathbf{w} - \mathbf{x}\|^2 + \epsilon} = \frac{N(\mathbf{x})}{D(\mathbf{x})}$$

where $N(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle^2$ and $D(\mathbf{x}) = \|\mathbf{w} - \mathbf{x}\|^2 + \epsilon$.

Using the quotient rule for vector calculus, the gradient $\nabla_{\mathbf{x}}\mathbb{E}$ is:

$$\nabla_{\mathbf{x}}\mathbb{E} = \frac{(\nabla_{\mathbf{x}} N)D - N(\nabla_{\mathbf{x}} D)}{D^2}$$

First, we compute the gradients of the numerator $N(\mathbf{x})$ and the denominator $D(\mathbf{x})$:

1. Gradient of the Numerator

$N(\mathbf{x}) = (\mathbf{w}^T\mathbf{x})^2$

$$\nabla_{\mathbf{x}} N(\mathbf{x}) = 2(\mathbf{w}^T\mathbf{x}) \cdot \nabla_{\mathbf{x}}(\mathbf{w}^T\mathbf{x}) = 2\langle \mathbf{w}, \mathbf{x} \rangle \mathbf{w}$$

2. Gradient of the Denominator

$D(\mathbf{x}) = \|\mathbf{w} - \mathbf{x}\|^2 + \epsilon = (\mathbf{w} - \mathbf{x})^T(\mathbf{w} - \mathbf{x}) + \epsilon$

$$\nabla_{\mathbf{x}} D(\mathbf{x}) = 2(\mathbf{w} - \mathbf{x}) \cdot (-1) = -2(\mathbf{w} - \mathbf{x}) = 2(\mathbf{x} - \mathbf{w})$$

Substituting these into the quotient rule expression:

$$\nabla_{\mathbf{x}}\mathbb{E} = \frac{(2\langle \mathbf{w}, \mathbf{x} \rangle \mathbf{w})(\|\mathbf{w} - \mathbf{x}\|^2 + \epsilon) - (\langle \mathbf{w}, \mathbf{x} \rangle^2)(2(\mathbf{x} - \mathbf{w}))}{(\|\mathbf{w} - \mathbf{x}\|^2 + \epsilon)^2}$$

To analyze the behavior for distant inputs, we examine the limit as $\|\mathbf{x}\| \to \infty$. Let $\mathbf{x} = k\mathbf{u}$, where $k = \|\mathbf{x}\|$ and $\mathbf{u}$ is a unit vector.

As $k \to \infty$:

- $\langle \mathbf{w}, \mathbf{x} \rangle = k\langle \mathbf{w}, \mathbf{u} \rangle \sim \mathcal{O}(k)$
- $\|\mathbf{w} - \mathbf{x}\|^2 = \|\mathbf{w}\|^2 - 2k\langle \mathbf{w}, \mathbf{u} \rangle + k^2 \sim \mathcal{O}(k^2)$

Let's analyze the order of magnitude for the terms in the gradient's numerator:

- First term: $(2\langle \mathbf{w}, \mathbf{x} \rangle \mathbf{w})(\|\mathbf{w} - \mathbf{x}\|^2 + \epsilon) \sim \mathcal{O}(k) \cdot \mathcal{O}(k^2) = \mathcal{O}(k^3)$

- Second term: $(\langle \mathbf{w}, \mathbf{x} \rangle^2)(2(\mathbf{x} - \mathbf{w})) \sim \mathcal{O}(k^2) \cdot \mathcal{O}(k) = \mathcal{O}(k^3)$

The numerator as a whole is of order $\mathcal{O}(k^3)$.

The denominator is $(\|\mathbf{w} - \mathbf{x}\|^2 + \epsilon)^2 \sim (\mathcal{O}(k^2))^2 = \mathcal{O}(k^4)$.

Therefore, the magnitude of the gradient behaves as:

$$\|\nabla_{\mathbf{x}} \mathbb{E}\| \sim \frac{\mathcal{O}(k^3)}{\mathcal{O}(k^4)} = \mathcal{O}\left(\frac{1}{k}\right)$$

As $k = \|\mathbf{x}\| \to \infty$, the magnitude of the gradient approaches zero:

$$\lim_{\|\mathbf{x}\| \to \infty} \|\nabla_{\mathbf{x}} \mathbb{E}(\mathbf{w}, \mathbf{x})\| = 0$$

This proves that for inputs $\mathbf{x}$ that are very far from the weight vector $\mathbf{w}$, the gradient becomes vanishingly small. The learning process is therefore stable, as distant outliers will not exert a significant influence on the weight updates. □

A.7 Regularity Properties of the $\mathbb{E}$-Product Kernel

Lemma A.4 (Analyticity)**.** *For any fixed $\mathbf{w} \in \mathbb{R}^d$ and $\epsilon > 0$, the map*

$$\mathbf{x} \mapsto K(\mathbf{w}, \mathbf{x})$$

is real-analytic on $\mathbb{R}^d$, and in particular infinitely differentiable (C^∞).

Proof. Both $N(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle^2$ and $D(\mathbf{x}) = \|\mathbf{w} - \mathbf{x}\|^2 + \epsilon$ are polynomials; since $D \geq \epsilon > 0$, $1/D$ is analytic and so is $K = N/D$. □

Proposition A.5 (Lipschitz Continuity)**.** *For any fixed $\mathbf{w} \in \mathbb{R}^d$ and $\epsilon > 0$, the kernel $K(\mathbf{w}, \mathbf{x})$ is globally Lipschitz continuous in $\mathbf{x}$.*

Proof. It suffices to show that $\sup_{\mathbf{x} \in \mathbb{R}^d} \|\nabla_{\mathbf{x}} K(\mathbf{w}, \mathbf{x})\| < \infty$.

Writing $K = N/D$ with $N(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle^2$ and $D(\mathbf{x}) = \|\mathbf{w} - \mathbf{x}\|^2 + \epsilon$, we compute

$$\nabla_{\mathbf{x}} K(\mathbf{w}, \mathbf{x}) = \frac{2\langle \mathbf{w}, \mathbf{x} \rangle \, \mathbf{w} \, D(\mathbf{x}) - 2\langle \mathbf{w}, \mathbf{x} \rangle^2 (\mathbf{x} - \mathbf{w})}{D(\mathbf{x})^2}.$$

Let $a = \|\mathbf{w}\|$, $r = \|\mathbf{x}\|$. By Cauchy–Schwarz,

$$|\langle \mathbf{w}, \mathbf{x} \rangle| \leq ar, \qquad \|\mathbf{x} - \mathbf{w}\| \leq r + a.$$

Moreover

$$D(\mathbf{x}) = \|\mathbf{w} - \mathbf{x}\|^2 + \epsilon \;\geq\; (r - a)^2 + \epsilon.$$

Substituting these bounds gives

$$\|\nabla_{\mathbf{x}} K(\mathbf{w}, \mathbf{x})\| \;\leq\; \frac{2a^2 r}{(r - a)^2 + \epsilon} \;+\; \frac{2a^2 r^2 (r + a)}{\big((r - a)^2 + \epsilon\big)^2} =: \Phi(r).$$

As $r \to \infty$, $\Phi(r) = O(1/r) \to 0$. Since Φ is continuous on $[0, \infty)$ and bounded at infinity, it attains a finite global maximum:

$$L(a, \epsilon) := \sup_{r \geq 0} \Phi(r) < \infty.$$

By the mean value theorem in $\mathbb{R}^d$,

$$|K(\mathbf{w}, \mathbf{x}) - K(\mathbf{w}, \mathbf{y})| \leq L(a, \epsilon) \, \|\mathbf{x} - \mathbf{y}\|,$$

so K is globally Lipschitz in $\mathbf{x}$ with constant $L(a, \epsilon)$, depending on $\|\mathbf{w}\|$ and ϵ. □

A.8 Universal Approximation for $\mathbb{E}$-Product Networks

In this section we give complete proofs for universality of single-hidden-layer Neural-Matter Networks (NMNs) built from the $\mathbb{E}$-product kernel

$$k_{\mathbb{E}}(\mathbf{x}, \mathbf{w}) \; = \; \frac{\langle \mathbf{x}, \mathbf{w} \rangle^2}{\|\mathbf{x} - \mathbf{w}\|^2 + \epsilon} \quad (\epsilon > 0),$$

as introduced in Section A.3. We now state the main universality result and provide a complete proof.

Proof of Theorem 2.2. We split into two cases. For convenience denote by

$$K_0(w, x) \; = \; \frac{(w \cdot x)^2}{\|w - x\|^2 + \epsilon}$$

the kernel used in the expansion; constants will be handled via an external bias term.

Case (1). Suppose $\inf_{\mathbf{x} \in K} \|\mathbf{x}\| =: m > 0$. Let $M := \sup_{\mathbf{x} \in K} \|\mathbf{x}\| < \infty$. For $w \in W$ and $x \in K$ write $w = r_w v$, $x = r_x u$ with radii $r_w = \|w\|$, $r_x = \|x\|$ and unit vectors $v, u \in S^{d-1}$. Then

$$K_0(w, x) = \frac{(r_w r_x)^2 (v \cdot u)^2}{r_w^2 + r_x^2 - 2 r_w r_x (v \cdot u) + \epsilon}.$$

For each fixed (r_w, r_x), define the function of $t \in [-1, 1]$

$$\Phi_{r_w, r_x}(t) = \frac{(r_w r_x)^2 t^2}{r_w^2 + r_x^2 - 2 r_w r_x t + \epsilon}.$$

This is analytic in t on $[-1, 1]$ and admits a uniformly (and hence absolutely) convergent power series

$$\Phi_{r_w, r_x}(t) = \sum_{n=0}^{\infty} a_n(r_w, r_x) \, t^{n+2},$$

where $a_n(r_w, r_x) \geq 0$ and infinitely many coefficients are strictly positive (this follows from the Laplace-transform representation in Section A.3). Note that each monomial $t^k = (v \cdot u)^k$ is a positive semidefinite kernel on S^{d-1} (it is the reproducing kernel of the k-fold symmetric tensor power). Hence any uniformly convergent series $\sum_{k \geq 0} \alpha_k (v \cdot u)^k$ with $\alpha_k \geq 0$ is a positive definite zonal kernel. Thus $(v, u) \mapsto \Phi_{r_w, r_x}(v \cdot u)$ is PD and zonal.

By classical results of Schoenberg (Schönberg, 1948) on positive definite zonal kernels, and their modern presentation in (Steinwart & Christmann, 2008, Thm. 4.62), the corresponding RKHS is dense in $C(S^{d-1})$.

Now observe that $K_0(w, x) = \Phi_{r_w, r_x}(v \cdot u)$ depends continuously on the radii (r_w, r_x), and the sets of admissible radii are compact. For each fixed r_w, the span of $\{u \mapsto \Phi_{r_w, r_x}(v \cdot u) : v \in S^{d-1}\}$ is dense in $C(S^{d-1})$; varying r_w continuously produces continuous radial dependence in r_x. Hence finite linear combinations of kernels $K_0(w, \cdot)$ generate functions of the form $R(r_x)\, Y(u)$ with $R \in C([m, M])$ and Y from a dense subalgebra of $C(S^{d-1})$. These functions separate points on $[m, M] \times S^{d-1}$ and contain constants (since $\Phi_{r_w, r_x}(1) > 0$ for suitable r_w, r_x). By the Stone–Weierstrass theorem, their uniform closure is all of $C([m, M] \times S^{d-1})$, which is homeomorphic to K under $x \mapsto (r_x, u)$. Thus $\mathcal{G}$ is dense in $C(K)$ in Case (1).

Case (2). Suppose K is any compact set (possibly containing the origin). Fix $f \in C(K)$ and $\varepsilon > 0$. Decompose $f(\mathbf{x}) = f(\mathbf{0}) + \tilde{f}(\mathbf{x})$, where $\tilde{f}(\mathbf{0}) = 0$ (if $\mathbf{0} \notin K$, choose any reference point). Use the model's external bias to match the constant, i.e., set the bias parameter $b := f(\mathbf{0})$. It remains to approximate $\tilde{f}$ uniformly by finite linear combinations of $K_0(\cdot, \mathbf{w})$.

By continuity of $\tilde{f}$ at the reference point, choose $\delta_0 > 0$ such that $|\tilde{f}(\mathbf{x})| < \varepsilon/3$ whenever $\|\mathbf{x}\| < \delta_0$. On the compact set $K \setminus B_{\delta_0}(\mathbf{0})$, points are bounded away from the origin, so Case (1) gives a finite expansion

$$g_0(\mathbf{x}) = \sum_{i=1}^{N} a_i K_0(\mathbf{w}_i, \mathbf{x})$$

with

$$\sup_{\mathbf{x} \in K \setminus B_{\delta_0}} |\tilde{f}(\mathbf{x}) - g_0(\mathbf{x})| < \varepsilon/3.$$

For $\|\mathbf{x}\| < \delta_0$, note

$$0 \le K_0(\mathbf{w}, \mathbf{x}) \le \frac{\|\mathbf{w}\|^2 \, \|\mathbf{x}\|^2}{\epsilon},$$

so with $R = \sup_{\mathbf{w} \in W} \|\mathbf{w}\| < \infty$,

$$|g_0(\mathbf{x})| \le \frac{R^2}{\epsilon}\, \delta_0^2 \sum_{i=1}^{N} |a_i|.$$

Since the sum is finite, shrink δ_0 if needed so that the right-hand side is $< \varepsilon/3$. Therefore

$$\sup_{\mathbf{x} \in K} |\tilde{f}(\mathbf{x}) - g_0(\mathbf{x})| < \varepsilon.$$

Adding back the constant $f(\mathbf{0})$ via the external bias term, we obtain a finite expansion of the form $b + \sum_i a_i K_0(\mathbf{w}_i, \cdot)$ that approximates f within ε uniformly on K.

Therefore $\mathcal{G}$ is dense in $C(K)$ in both cases. This completes the proof of Theorem 2.2. □

Relation to Mercer theory. By Theorem 2.1 the $\mathbb{E}$-product kernel $k_{\mathbb{E}}(\cdot, \cdot)$ is continuous and positive semidefinite on compact domains, hence admits a Mercer (spectral) decomposition

$$k_\varepsilon(x, y) = \sum_{j=1}^{\infty} \lambda_j \varphi_j(x) \varphi_j(y),$$

with $\lambda_j \ge 0$ and $\{\varphi_j\}$ orthonormal in $L^2(K)$. The zonal expansion used above (Schoenberg-style) guarantees nonzero spectral mass on arbitrarily high spherical-harmonic orders, so the eigenfunctions $\{\varphi_j\}$ span a dense subspace of $C(K)$. Equivalently, the RKHS associated with k_ε is dense in $C(K)$, which is precisely the universal-approximation property proven in Theorem 2.2. This Mercer viewpoint also yields practical tools (finite-rank truncation, Nyström, representer theorem) for analyzing and implementing finite NMNs that approximate continuous target functions on K.

A.9 Information-Geometric Foundations of the $\mathbb{E}$-Product

A.9.1 Definition and Geometric Interpretation

We consider probability distributions in the simplex $\Delta^{n-1} = \{\mathbf{p} \in \mathbb{R}^n_{\geq 0} : \sum_{i=1}^n p_i = 1\}$. While information geometry traditionally employs the Fisher metric, we establish a novel connection to Euclidean geometry through the $\mathbb{E}$-product.

Definition A.1 ($\mathbb{E}$-Product: Geometric Similarity Measure)**.** *For distinct distributions $\mathbf{p}, \mathbf{q} \in \Delta^{n-1}$, the $\mathbb{E}$-product is defined as:*

$$\mathbb{E}(\mathbf{p}, \mathbf{q}) \coloneqq \frac{(\mathbf{p} \cdot \mathbf{q})^2}{\|\mathbf{p} - \mathbf{q}\|_2^2}$$

where:

- $\mathbf{p} \cdot \mathbf{q} = \sum_{i=1}^n p_i q_i$ *measures distributional alignment*
- $\|\mathbf{p} - \mathbf{q}\|_2^2 = \sum_{i=1}^n (p_i - q_i)^2$ *quantifies Euclidean dissimilarity*

This ratio captures the tension between distributional agreement and geometric separation.

Remark A.6 (Singularity and Invariance Properties)**.** *When $\mathbf{p} = \mathbf{q}$, we define $\mathbb{E}(\mathbf{p}, \mathbf{q}) \coloneqq \infty$ via the limit:*

$$\lim_{\mathbf{q} \to \mathbf{p}} \mathbb{E}(\mathbf{p}, \mathbf{q}) = \infty$$

reflecting maximal self-similarity. The $\mathbb{E}$-product exhibits two key properties:

1. ***Symmetry**: $\mathbb{E}(\mathbf{p}, \mathbf{q}) = \mathbb{E}(\mathbf{q}, \mathbf{p})$*
2. ***Scale Invariance**: Invariant under index permutation*

A.9.2 Extremal Similarity Theorems

Theorem A.7 (Minimal Similarity and Statistical Orthogonality)**.** *For distinct $\mathbf{p}, \mathbf{q} \in \Delta^{n-1}$:*

$$\mathbb{E}(\mathbf{p}, \mathbf{q}) = 0 \iff \operatorname{supp}(\mathbf{p}) \cap \operatorname{supp}(\mathbf{q}) = \emptyset$$

Moreover, this condition implies information-theoretic divergence:

$$\mathrm{KL}(\mathbf{p} \| \mathbf{q}) = \infty, \qquad \mathrm{KL}(\mathbf{q} \| \mathbf{p}) = \infty, \qquad H(\mathbf{p}, \mathbf{q}) = \infty$$

Proof. ($\Rightarrow$) Assume $\mathbb{E}(\mathbf{p}, \mathbf{q}) = 0$. Since $\mathbf{p} \neq \mathbf{q}$, $\|\mathbf{p} - \mathbf{q}\|_2^2 > 0$. Thus $(\mathbf{p} \cdot \mathbf{q})^2 = 0 \Rightarrow \sum p_i q_i = 0$. By non-negativity of probabilities, $p_i q_i = 0 \ \forall i$, hence $\operatorname{supp}(\mathbf{p}) \cap \operatorname{supp}(\mathbf{q}) = \emptyset$.

($\Leftarrow$) Disjoint supports imply $\forall i : (p_i > 0 \Rightarrow q_i = 0)$ and vice versa. Thus $\mathbf{p} \cdot \mathbf{q} = 0$, so $\mathbb{E}(\mathbf{p}, \mathbf{q}) = 0$.

The KL divergence $\mathrm{KL}(\mathbf{p} \| \mathbf{q})$ contains terms $\log(p_i / q_i)$ where $p_i > 0$ and $q_i = 0$, causing divergence. Similar reasoning applies to $\mathrm{KL}(\mathbf{q} \| \mathbf{p})$ and cross-entropy $H(\mathbf{p}, \mathbf{q})$(Cover & Thomas, 2006). □

Theorem A.8 (Maximal Similarity and Distributional Identity)**.** *For $\mathbf{p}, \mathbf{q} \in \Delta^{n-1}$:*

$$\mathbb{E}(\mathbf{p}, \mathbf{q}) = \infty \iff \mathbf{p} = \mathbf{q}$$

When satisfied, information-theoretic consistency holds:

$$\mathrm{KL}(\mathbf{p} \| \mathbf{q}) = 0 \qquad \textit{and} \qquad H(\mathbf{p}, \mathbf{q}) = H(\mathbf{p})$$

Proof. ($\Rightarrow$) Suppose $\mathbb{E}(\mathbf{p}, \mathbf{q}) \to \infty$. By Cauchy-Schwarz(Horn & Johnson, 2012), $\mathbf{p} \cdot \mathbf{q} \leq \|\mathbf{p}\|_2 \|\mathbf{q}\|_2 \leq 1$. Since the numerator is bounded, $\|\mathbf{p} - \mathbf{q}\|_2^2 \to 0$, implying $\mathbf{p} = \mathbf{q}$.

($\Leftarrow$) For $\mathbf{p} = \mathbf{q}$, consider $\mathbf{q}^{(k)} \to \mathbf{p}$. Then:

$$\mathbf{p} \cdot \mathbf{q}^{(k)} \to \|\mathbf{p}\|_2^2 \geq \tfrac{1}{n} > 0 \quad \text{(since } \|\mathbf{p}\|_2^2 \geq \tfrac{1}{n} \text{ by Cauchy-Schwarz)}$$

while $\|\mathbf{p} - \mathbf{q}^{(k)}\|_2^2 \to 0$, so $\mathbb{E}(\mathbf{p}, \mathbf{q}^{(k)}) \to \infty$.

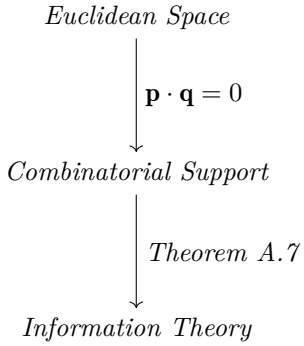
When $\mathbf{p} = \mathbf{q}$, $\log(p_i/q_i) = 0$ for all i, so $\mathrm{KL}(\mathbf{p}\|\mathbf{q}) = 0$. Cross-entropy reduces to entropy when distributions are identical. □

Remark A.9 (Duality of Orthogonality Concepts)**.** *The $\mathbb{E}$-product unifies three distinct notions of orthogonality:*

$$\begin{aligned} \textit{Euclidean:}\ & \mathbf{p} \perp \mathbf{q} \iff \mathbf{p} \cdot \mathbf{q} = 0 \\ \textit{Combinatorial:}\ & \mathrm{supp}(\mathbf{p}) \cap \mathrm{supp}(\mathbf{q}) = \emptyset \\ \textit{Information-Theoretic:}\ & \mathrm{KL}(\mathbf{p}\|\mathbf{q}) = \infty \end{aligned}$$

Theorem A.7 establishes their equivalence through $\mathbb{E}(\mathbf{p}, \mathbf{q}) = 0$. This contrasts with Fisher-based orthogonality, which depends on manifold curvature.

Remark A.10 (Geometric-Information Duality)**.** *The $\mathbb{E}$-product creates a bridge between geometric and probabilistic perspectives:*



B Theoretical Background

B.1 Revisiting Core Computational Primitives and Similarity Measures

The computational primitives used in deep learning are fundamental to how models represent and process information. This section revisits key mathematical operations and similarity measures, such as the dot product, convolution, cosine similarity, and Euclidean distance, that form the bedrock of many neural architectures. We will explore their individual properties and how they contribute to tasks like feature alignment, localized feature mapping, and quantifying spatial proximity. Furthermore, we will delve into the role of neural activation functions in enabling the non-linear transformations crucial for complex pattern recognition. Understanding these core concepts and their inherent characteristics is crucial for appreciating the motivation behind developing novel operators, as explored in this work, that aim to capture more nuanced relationships within data Goodfellow et al. (2016).

B.1.1 The Dot Product: A Measure of Alignment

The dot product, or scalar product, remains a cornerstone of neural computation, serving as the primary mechanism for quantifying the interaction between vectors, such as a neuron's weights and its input. For two vectors $\mathbf{a} = [a_1, a_2, \ldots, a_n]$ and $\mathbf{b} = [b_1, b_2, \ldots, b_n]$, it is defined as:

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n \tag{4}$$

Geometrically, the dot product is proportional to the cosine of the angle between the vectors and their Euclidean magnitudes: $\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\|\|\mathbf{b}\| \cos(\theta)$. Its sign indicates the general orientation (acute, obtuse, or orthogonal angle), and its magnitude reflects the degree of alignment scaled by vector lengths. In machine learning, dot product scores are pervasively used to infer similarity, relevance, or the strength of activation. However, as noted in Section 1, its conflation of magnitude and directional alignment can sometimes obscure more fine-grained geometric relationships, motivating the exploration of operators that offer a more comprehensive assessment of vector interactions.

B.1.2 The Convolution Operator: Localized Feature Mapping

The convolution operator is pivotal in processing structured data, particularly in Convolutional Neural Networks (CNNs). It applies a kernel (or filter) across an input to produce a feature map, effectively an operation on two functions, f (input) and g (kernel), yielding a third that expresses how one modifies the shape of the other. For discrete signals, such as image patches and kernels, it is:

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m] g[n - m] \tag{5}$$

In CNNs, convolution performs several critical roles:

- **Feature Detection:** Kernels learn to identify localized patterns (edges, textures, motifs) at various abstraction levels.
- **Spatial Hierarchy:** Stacking layers allows the model to build complex feature representations from simpler ones.
- **Parameter Sharing:** Applying the same kernel across spatial locations enhances efficiency and translation equivariance.

The core computation within a discrete convolution at a specific location involves an element-wise product sum between the kernel and the corresponding input patch, which is, in essence, a dot product. Consequently, the resulting activation at each point in the feature map reflects the local alignment between the input region and the kernel. If an input patch and a kernel are orthogonal (i.e., their element-wise product sums to zero, akin to a zero dot product if they were vectorized), the convolution output at that position will be zero, indicating no local match for the feature encoded by the kernel. This reliance on dot product-like computations means that standard convolutions primarily assess feature alignment, potentially overlooking other geometric aspects of the data.

B.1.3 Cosine Similarity: Normalizing for Directional Agreement

Cosine similarity refines the notion of alignment by isolating the directional aspect of vector relationships, abstracting away from their magnitudes. It measures the cosine of the angle between two non-zero vectors $\mathbf{A}$ and $\mathbf{B}$:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{6}$$

Scores range from -1 (perfectly opposite) to 1 (perfectly aligned), with 0 signifying orthogonality (decorrelation). By normalizing for vector lengths, cosine similarity provides a pure measure of orientation. This is particularly useful when the magnitude of vectors is not indicative of their semantic relationship, such as in document similarity tasks. While it effectively captures directional agreement, it explicitly discards information about vector magnitudes and, like the dot product, does not inherently account for the spatial proximity between the vectors themselves if they are points in a space (Draganov et al., 2024; Steck et al., 2024).

B.1.4 Euclidean Distance: Quantifying Spatial Proximity

In contrast to measures of alignment, Euclidean distance quantifies the "ordinary" straight-line separation between two points (or vectors) $\mathbf{p} = (p_1, \ldots, p_n)$ and $\mathbf{q} = (q_1, \ldots, q_n)$ in an n-dimensional Euclidean space:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \tag{7}$$

This metric is fundamental in various machine learning algorithms, including k-Nearest Neighbors and k-Means clustering, and forms the basis of loss functions like Mean Squared Error. Euclidean distance measures dissimilarity based on spatial proximity; a smaller distance implies greater similarity in terms of location within the vector space. Unlike cosine similarity, it is sensitive to vector magnitudes and their absolute positions. However, Euclidean distance alone does not directly convey information about the relative orientation or alignment of vectors, only their nearness.

The distinct characteristics of these foundational measures, alignment (dot product, cosine similarity) versus proximity (Euclidean distance), highlight an opportunity. These foundational measures force a choice: one can measure alignment (dot product, cosine similarity) or spatial proximity (Euclidean distance), but no single, primitive operator in conventional use effectively unifies both. Neural operators that can synergistically combine these aspects, assessing not only if vectors point in similar directions but also if they are close in the embedding space, could offer a richer, more geometrically informed way to model interactions. This perspective underpins the development of the $\mathbb{E}$-product introduced in Section 2.

B.2 The Role and Geometric Cost of Non-Linear Activation

While the core computational primitives provide tools to measure similarity and interaction, their inherent linearity limits the complexity of functions they can represent. To overcome this, deep neural networks employ non-linear activation functions. These are the standard method for introducing non-linearity, a necessary step for modeling intricate data patterns. However, this "fix" is imperfect, as it introduces its own set of problems, particularly concerning the preservation of the input data's geometric integrity. The remarkable expressive power of deep neural networks hinges on their capacity to model complex, non-linear relationships. This ability to approximate any continuous function to an arbitrary degree of accuracy is formally captured by the universal approximation theorem Cybenko (1989); Hornik et al. (1989); Lu et al. (2017); Huang (2020).

This theorem underscores the critical role of non-linear activation functions. Without such non-linearities, a deep stack of layers would mathematically collapse into an equivalent single linear transformation, severely curtailing its representational capacity. Activation functions are thus not mere auxiliaries; they are the pivotal components that unlock the hierarchical and non-linear feature learning central to deep learning's success. They determine a neuron's output based on its aggregated input, and in doing so, introduce crucial selectivity: enabling the network to preferentially respond to certain patterns while attenuating or ignoring others.

B.2.1 Linear Separability and the Limitations of the Inner Product

The fundamental computation within a single artificial neuron (perceptron) is an affine transformation followed by a non-linear activation function σ:

$$y = \sigma(\langle \mathbf{w}, \mathbf{x} \rangle + b), \tag{8}$$

where $\mathbf{w}$ is the weight vector, $\mathbf{x}$ is the input vector, and b is the bias term. The decision boundary of this neuron is implicitly defined by the hyperplane where the argument to σ is zero:

$$\{\mathbf{x} \in \mathbb{R}^d \mid \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\}. \tag{9}$$

This hyperplane partitions the input space $\mathbb{R}^d$ into two half-spaces. Consequently, a single neuron can only implement linearly separable functions. This is a direct consequence of the linear nature of the inner product, which can only define a linear decision boundary. While this allows for efficient computation, it severely restricts the complexity of functions that can be learned.

A classic counterexample is the XOR function, whose truth table cannot be satisfied by any single linear decision boundary. Specifically, for inputs $\mathbf{x} \in \{(0,0),(0,1),(1,0),(1,1)\} \subset \mathbb{R}^2$, there exist no $\mathbf{w} \in \mathbb{R}^2$ and $b \in \mathbb{R}$ such that $\text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle + b)$ matches the XOR output (0, 1, 1, 0 respectively). This limitation stems directly from the linear nature of the inner product operation defining the separating boundary Goodfellow et al. (2016).

B.2.2 Non-Linear Feature Space Transformation via Hidden Layers and its Geometric Cost

Multi-layer perceptrons (MLPs) overcome this limitation by cascading transformations. A hidden layer maps the input $\mathbf{x}$ to a new representation $\mathbf{h}$ through a matrix-vector product and an element-wise activation function ϕ:

$$\mathbf{h} = \phi(W\mathbf{x} + \mathbf{b}). \tag{10}$$

Here, $W \in \mathbb{R}^{m \times d}$ is the weight matrix, $\mathbf{b} \in \mathbb{R}^m$ is the bias vector, and m is the number of hidden neurons. Each row $\mathbf{w}_i^\top$ of W corresponds to the weight vector of the i-th hidden neuron, computing $h_i = \phi(\langle \mathbf{w}_i, \mathbf{x} \rangle + b_i)$. This transforms the input space $\mathbb{R}^d$ into a feature space $\mathbb{R}^m$. The introduction of the non-linear activation function ϕ is what allows the network to learn non-linear decision boundaries. However, this gain in expressive power comes at a cost: the potential loss of geometric fidelity.

B.2.3 Topological Distortions and Information Loss via Activation Functions

While hidden layers using the transformation $\mathbf{h} = \phi(W\mathbf{x} + \mathbf{b})$ enable the learning of non-linear functions, the introduction of the element-wise non-linear activation function ϕ, often crucial for breaking linearity, can significantly alter the topological and geometric structure of the data representation, potentially leading to information loss Goodfellow et al. (2016). This is a critical trade-off: gaining non-linear modeling capability while potentially discarding valuable geometric information.

Consider the mapping $T : \mathbb{R}^d \to \mathbb{R}^m$ defined by $T(\mathbf{x}) = \phi(W\mathbf{x}+\mathbf{b})$. The affine part, $A(\mathbf{x}) = W\mathbf{x}+\mathbf{b}$, performs a linear transformation (rotation, scaling, shear, projection) followed by a translation. While this affine map distorts metric properties (distances and angles, unless W is proportional to an orthogonal matrix), it preserves basic topological features like connectedness and maps lines to lines (or points) Goodfellow et al. (2016).

However, the subsequent application of a typical non-linear activation ϕ element-wise often leads to more drastic topological changes:

1. Non-Injectivity and Collapsing Regions: Many common activation functions render the overall mapping T non-injective.
   - ReLU ($\phi(z) = \max(0, z)$): Perhaps the most prominent example. For each hidden neuron i, the entire half-space defined by $\{\mathbf{x} \in \mathbb{R}^d \mid \langle \mathbf{w}_i, \mathbf{x} \rangle + b_i \leq 0\}$ is mapped to $h_i = 0$. Distinct points $\mathbf{x}_1, \mathbf{x}_2$ within this region, potentially far apart, become indistinguishable along the i-th dimension of the hidden space. This constitutes a significant loss of information about the relative arrangement of data points within these collapsed regions. The mapping is fundamentally many-to-one. For instance, consider two input vectors that are anti-aligned with a neuroń s weight vector to different degrees, one strongly and one weakly. A ReLU activation function would map both resulting negative dot products to zero, rendering their distinct geometric opposition indistinguishable to subsequent layers. This information is irretrievably discarded.
   - Sigmoid/Tanh: While smooth, these functions saturate. Inputs $\mathbf{z}_1 = A(\mathbf{x}_1)$ and $\mathbf{z}_2 = A(\mathbf{x}_2)$ that are far apart but both fall into the saturation regime (e.g., large positive or large negative values) will map to $\mathbf{h}_1 \approx \mathbf{h}_2$. This 'squashing' effect can merge distinct clusters from the input space if they map to saturated regions in the hidden space, again losing discriminative information and distorting the metric structure.
2. Distortion of Neighborhoods: The relative distances between points can be severely distorted. Points close in the input space $\mathbb{R}^d$ might be mapped far apart in $\mathbb{R}^m$, or vice-versa (especially due to saturation or the zero-region of ReLU). This means the local neighborhood structure is not faithfully preserved. Formally, the mapping T is generally not a homeomorphism onto its image, nor is it typically bi-Lipschitz (which would provide control over distance distortions).

While these distortions are precisely what grant neural networks their expressive power to warp the feature space and create complex decision boundaries, they come at the cost of potentially discarding information present in the original geometric configuration of the data. The network learns which information to preserve and which to discard based on the optimization objective, but the mechanism relies on potentially non-smooth or non-injective transformations introduced by ϕ. This highlights the conflation of magnitude and direction in the dot product, the information loss from activation functions, and the lack of a unified measure for proximity and alignment, setting the stage for the $\mathbb{E}$-product. Formal properties of the $\mathbb{E}$-product are established later: it is a Mercer kernel (Theorem 2.1), yields universal approximation in NMNs (Theorem 2.2), and exhibits self-regulation and stable gradients (Propositions A.1 and A.3).

B.3 Design Philosophy: Intrinsic Non-Linearity and Self-Regulation

A central hypothesis underpinning our methodological choices is that the $\mathbb{E}$-product (Section 1) possesses inherent non-linearity and self-regulating properties that can reduce or eliminate the need for conventional activation functions (e.g., ReLU, sigmoid, GeLU).

This philosophy recontextualizes the fundamental components of neural computation. Neuron weights ($\mathbf{w}$) and input signals ($\mathbf{x}$) are not merely operands in a linear transformation followed by a non-linear activation; instead, they are conceptualized as co-equal vector entities inhabiting a shared, high-dimensional feature manifold. Within this framework, each vector can be viewed as an analogue to a fundamental particle or feature vector, with its constituent dimensions potentially encoding excitatory, inhibitory, or neutral characteristics relative to other entities in the space. The $\mathbb{E}$-product (Section 1) then transcends simple similarity assessment; it functions as a sophisticated interaction potential, $\mathcal{K}_{\mathbb{E}}(\mathbf{w}, \mathbf{x}) = \frac{(\mathbf{w}^\top \mathbf{x})^2}{\|\mathbf{w}-\mathbf{x}\|^2+\epsilon}$, quantifying the 'field effects' between these vector entities. This interaction is reminiscent of n-body problems in physics. In machine learning, it draws

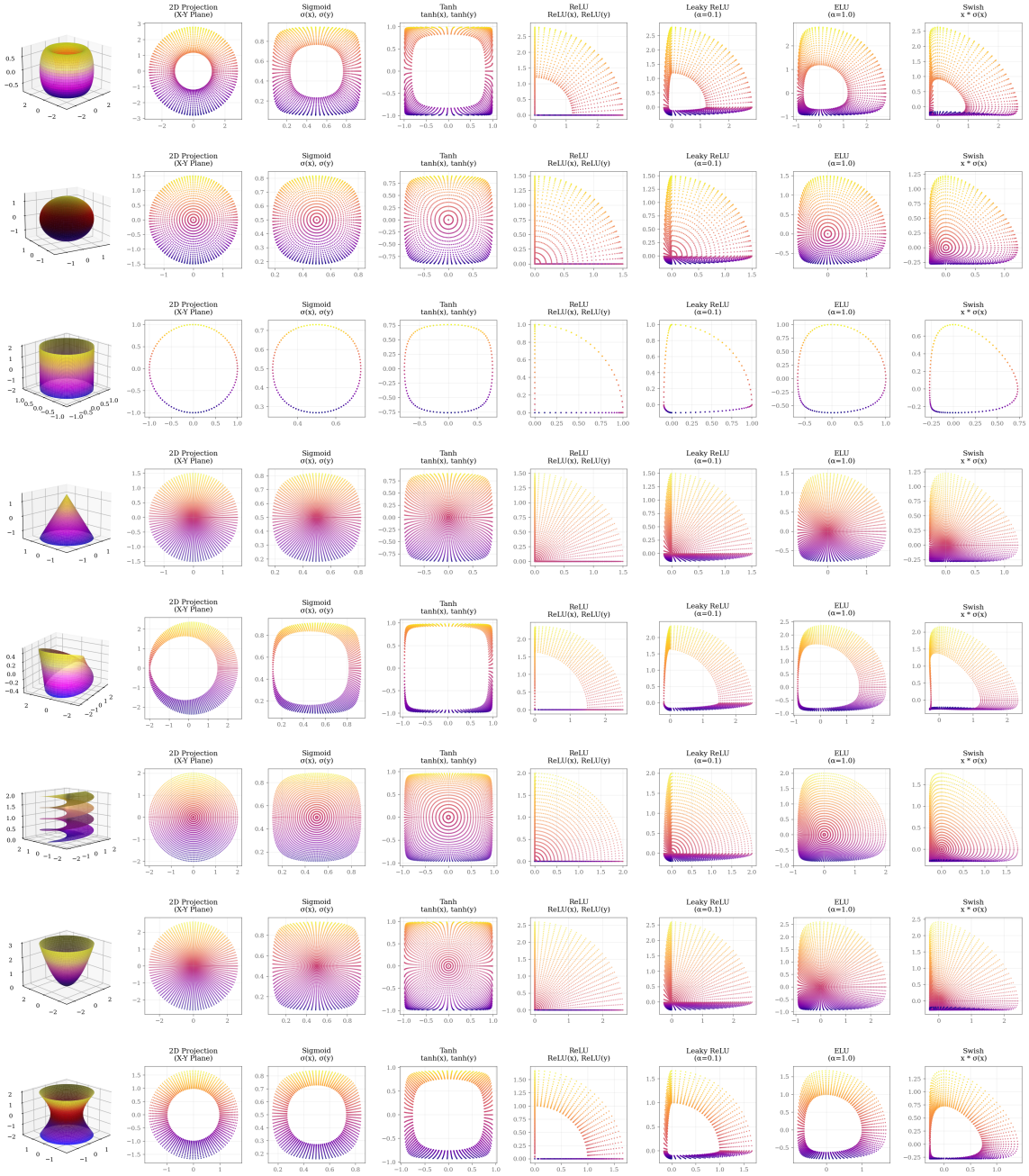


Figure 4: Illustration of how non-linear activation functions can distort the geometric structure of the input data manifold, leading to potential information loss. The original manifold (left) is transformed into a distorted representation after applying a non-linear activation functions.

parallels with, yet distinctively evolves from, learned metric spaces in contrastive learning, particularly those employing a triplet loss framework. While triplet loss aims to pull positive pairs closer and push negative pairs apart in the embedding space, our $\mathbb{E}$-product seeks a more nuanced relationship: 'positive' interactions (high $\mathbb{E}$-product value) require both strong alignment (high $(\mathbf{w}^\top \mathbf{x})^2$) and close proximity (low $\|\mathbf{w} - \mathbf{x}\|^2$). Conversely, 'negative' or dissimilar relationships are not merely represented by distance, but more significantly by orthogonality (leading to a vanishing numerator), which signifies a form of linear independence and contributes to the system's capacity for true non-linear discrimination. Crucially, the non-linearity required for complex pattern recognition is not an external imposition (e.g., via a separate activation function) but is intrinsic to this interaction potential. The interplay between the squared dot product (alignment sensitivity) and the inverse squared Euclidean distance (proximity sensitivity) in its formulation directly sculpts a complex, non-linear response landscape without recourse to auxiliary functions.

Furthermore, this conceptualization of the $\mathbb{E}$-product as an intrinsic interaction potential suggests inherent self-regulating properties. The distance-sensitive denominator, $\|\mathbf{w} - \mathbf{x}\|^2 + \epsilon$, acts as a natural dampening mechanism. As the 'distance' (dissimilarity in terms of position) between interacting vector entities $\mathbf{w}$ and $\mathbf{x}$ increases, the strength of their interaction, and thus the resultant activation, diminishes quadratically. This behavior is hypothesized to inherently curtail runaway activations and stabilize learning dynamics by ensuring that responses are localized and bounded. Such intrinsic stabilization contrasts sharply with conventional approaches that rely on explicit normalization layers (e.g., Batch Normalization, Layer Normalization) to manage activation statistics post-hoc. These layers, while effective, introduce additional computational overhead, can obscure direct input-output relationships, and sometimes complicate the theoretical analysis of network behavior. The $\mathbb{E}$-product's formulation, therefore, offers a pathway to architectures where regulatory mechanisms are embedded within the primary computational fabric of the network.

The inherent non-linearity of the $\mathbb{E}$-product, coupled with the self-regulating properties suggested by its formulation (and formally proven in Appendix A.4), are central to our hypothesis that it can form the basis of powerful and robust neural architectures. These intrinsic characteristics open avenues for simplifying network design, potentially reducing reliance on or even eliminating conventional activation functions and normalization layers.

C Squashing Functions for Non-Negative Scores

The $\mathbb{E}$-product and its derivatives, such as the $\mathcal{K}_{\mathbb{E}}$-kernel, naturally yield non-negative scores. In many machine learning contexts, particularly when these scores need to be interpreted as probabilities, attention weights, or simply normalized outputs, it is essential to apply a squashing function to map them to a desired range (e.g., [0, 1] or ensuring a set of scores sum to 1).

C.1 Categorization of Squashing Functions

Squashing functions for non-negative scores can be broadly categorized into two types:

- **Competitive (Vector-Normalizing) Functions:** These functions normalize a set of scores collectively, producing a distribution over the vector. Each output depends on the values of all dimensions, allowing for competitive interactions among them. This is useful for attention mechanisms or probability assignments where the sum of outputs is meaningful.
- **Individualistic (Per-Dimension) Functions:** These functions squash each score independently, without reference to other values in the vector. Each output depends only

on its corresponding input, making them suitable for bounding or interpreting individual activations.

C.2 Limitations of Traditional Squashing Functions

Traditional squashing functions, however, present challenges when applied to non-negative inputs:

- **Standard Sigmoid Function ($\sigma(x) = \frac{1}{1+e^{-x}}$):** When applied to non-negative inputs ($x \geq 0$), the standard sigmoid function produces outputs in the range $[0.5, 1)$. The minimum value of 0.5 for $x = 0$ renders it unsuitable for scenarios where small non-negative scores should map to values close to 0.
- **Standard Softmax Function ($\mathbf{softmax}(\mathbf{x})_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$):** The use of the exponential function in softmax can lead to *hard* distributions, where one input value significantly dominates the output, pushing other probabilities very close to zero. While this is often desired for classification, it can be too aggressive if a softer assignment of probabilities or attention is preferred. Additionally, softmax can suffer from numerical instability for large input values due to the exponentials.

C.3 Proposed Alternative Squashing Functions

Given these limitations and the non-negative nature of $\mathbb{E}$-product scores, we consider alternative squashing functions more suited to this domain:

- **softermax (Competitive):** This function normalizes a score x_k (optionally raised to a power $n > 0$) relative to the sum of a set of non-negative scores $\{x_i\}$ (each raised to n), with a small constant $\epsilon > 0$ for numerical stability. It is defined as:

$$\text{softermax}_n(x_k, \{x_i\}) = \frac{x_k^n}{\epsilon + \sum_i x_i^n} \tag{11}$$

  Unlike softmax, softermax does not use exponentials, which avoids numerical instability for large inputs and provides a more direct, interpretable translation of the underlying scores into a normalized distribution. The power n controls the sharpness of the distribution: $n = 1$ recovers the original Softermax, while $n > 1$ makes the distribution harder (more peaked), and $0 < n < 1$ makes it softer.

- **soft-sigmoid (Individualistic):** This function squashes a single non-negative score $x \geq 0$ (optionally raised to a power $n > 0$) into the range $[0, 1)$. It is defined as:

$$\text{soft-sigmoid}_n(x) = \frac{x^n}{1 + x^n} \tag{12}$$

  The power n modulates the softness: higher n makes the function approach zero faster for large x, while $n < 1$ makes the decay slower.

- **soft-tanh (Individualistic):** This function maps a non-negative score $x \geq 0$ (optionally raised to a power $n > 0$) to the range $[-1, 1)$ by linearly transforming the output of soft-sigmoid. It is defined as:

$$\text{soft-tanh}_n(x) = 2 \cdot (\text{soft-sigmoid}_n(x) - \tfrac{1}{2}) = \frac{x^n - 1}{1 + x^n} \tag{13}$$

  The power n again controls the transition sharpness: higher n makes the function approach -1 more quickly for large x.

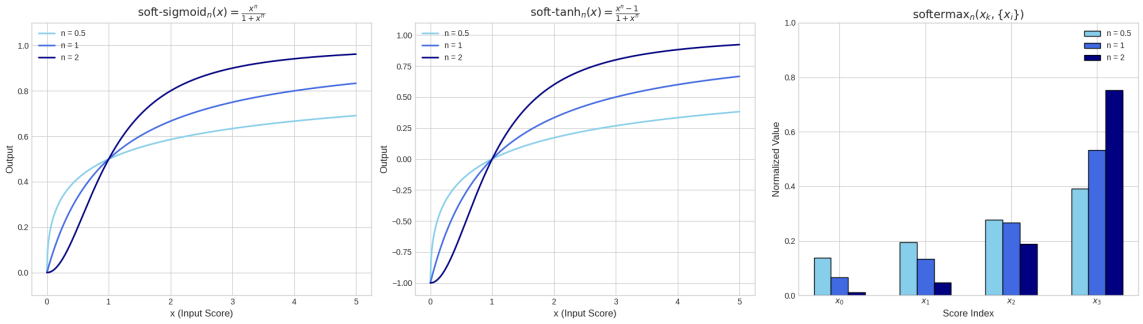


Figure 5: Visualization of the softermax, soft-sigmoid, and soft-tanh functions. These functions are designed to handle non-negative inputs from the 𝗘-product and its derivatives, providing appropriate squashing mechanisms that maintain sensitivity across the range of non-negative inputs.

These functions are particularly well-suited for the outputs of 𝗘-product-based computations, as they maintain sensitivity across the range of non-negative inputs while avoiding the pitfalls of standard activation functions (Nair & Hinton, 2010; Hendrycks & Gimpel, 2023; Klambauer et al., 2017).

C.4 Functional Roles and Applications

The main role of these squashing functions can be categorized into two main categories:

- **Collective Communication and Space Splitting:** The softermax function allows for a comparative analysis of scores, reflecting their orthogonality and spatial proximity to an input vector. A higher score indicates that a vector is more aligned and closer to the input, while a lower score suggests greater orthogonality. This facilitates a competitive interaction where vectors vie for influence based on their geometric relationship with the input. The power parameter n, analogous to the temperature in softmax, controls the sharpness of the gravitational potential well's slope.

- **Individual Score Squashing:** The soft-sigmoid and soft-tanh functions are used to squash individual non-negative scores into a bounded range, typically $[0, 1)$ for soft-sigmoid and $[-1, 1)$ for soft-tanh. They are particularly useful when the output needs to be interpreted as a probability or when a bounded response is required, as each score is processed independently of the others. The power parameter controls the steepness of the function, while the minimum value can be interpreted as an orthogonality score.

C.5 Computational Complexity Analysis

We provide exact forward/backward complexity, closed-form gradients, asymptotic characterization, and per-neuron FLOP counts for the 𝗘-product layer (definition in Methodology, Section 2.1).

C.5.1 Forward Pass Complexity

For $\mathbb{E}$-product computation $\mathbb{E}(\mathbf{w}, \mathbf{x}) = \frac{(\mathbf{w}^\top \mathbf{x})^2}{\|\mathbf{w}-\mathbf{x}\|^2+\epsilon}$, we apply the algebraic identity:

$$\mathbb{E}(\mathbf{w}, \mathbf{x}) = \frac{s^2}{\|\mathbf{w}\|^2 + \|\mathbf{x}\|^2 - 2s + \epsilon}, \quad s = \mathbf{w}^\top \mathbf{x} \tag{14}$$

For layer $X \in \mathbb{R}^{B\times d} \to Y \in \mathbb{R}^{B\times n}$ with weights $W \in \mathbb{R}^{n\times d}$:

Operation Breakdown (1) GEMM: $S = XW^\top$ ($2Bnd$); (2) Row norms $\|X\|^2$ once (Bd); (3) Cached $\|W\|^2$ (nd only when updated); (4) Per-output element-wise: form s^2 (1), denominator assemble (+, +, +) (3), reciprocal (1), multiply (1) $\Rightarrow 6Bn$ scalar ops (we conservatively use $5Bn$ after fusion). Thus

$$T_{\text{forward}} = 2Bnd + Bd + nd + 5Bn = \Theta(Bnd). \tag{15}$$

C.5.2 Backward Pass Complexity

For $y = \dfrac{s^2}{D}$ with $D = \|\mathbf{w}\|^2 + \|\mathbf{x}\|^2 - 2s + \epsilon$ and $s = \mathbf{w}^\top \mathbf{x}$, scalar gradients:

$$\frac{\partial y}{\partial s} = \frac{2s(\|\mathbf{w}\|^2 + \|\mathbf{x}\|^2 + \epsilon - s)}{D^2} \tag{16}$$

$$\frac{\partial y}{\partial \|\mathbf{x}\|^2} = -\frac{s^2}{D^2}, \qquad \frac{\partial y}{\partial \|\mathbf{w}\|^2} = -\frac{s^2}{D^2} \tag{17}$$

Vector gradients (using $\nabla_{\mathbf{x}} s = \mathbf{w}$, $\nabla_{\mathbf{w}} s = \mathbf{x}$, $\nabla_{\mathbf{x}} \|\mathbf{x}\|^2 = 2\mathbf{x}$, $\nabla_{\mathbf{w}} \|\mathbf{w}\|^2 = 2\mathbf{w}$):

$$\nabla_{\mathbf{x}} y = \frac{2s(\|\mathbf{w}\|^2 + \|\mathbf{x}\|^2 + \epsilon - s)}{D^2}\,\mathbf{w} - \frac{2s^2}{D^2}\,\mathbf{x} \tag{18}$$

$$\nabla_{\mathbf{w}} y = \frac{2s(\|\mathbf{w}\|^2 + \|\mathbf{x}\|^2 + \epsilon - s)}{D^2}\,\mathbf{x} - \frac{2s^2}{D^2}\,\mathbf{w} \tag{19}$$

Given upstream gradient $G \in \mathbb{R}^{B\times n}$:

$$G_S = G \odot \frac{\partial Y}{\partial S} \quad (\sim 6Bn \text{ FLOPs}) \tag{20}$$

$$\frac{\partial L}{\partial W} = G_S^\top X \quad (2Bnd) \tag{21}$$

$$\frac{\partial L}{\partial X} = G_S W \quad (2Bnd) \tag{22}$$

Hence

$$T_{\text{backward}} = 4Bnd + 6Bn + O(Bd + nd) = \Theta(Bnd). \tag{23}$$

C.5.3 Asymptotic Summary

C.5.4 Implementation Notes

Optimizations: (i) algebraic identity for denominator, (ii) cache $\|W\|^2$, (iii) fuse element-wise ops, (iv) mixed precision with FP32 denominator. Built-in boundedness mitigates gradient explosion.

Component	Linear	𝔼-Product
Forward main	$2Bnd$	$2Bnd$
Forward aux	Bn	$Bd + nd + 5Bn$
Backward main	$4Bnd$	$4Bnd$
Backward aux	$2Bn$	$6Bn + Bd + nd$
Total order	$\Theta(Bnd)$	$\Theta(Bnd)$
Overhead ratio	1	$1 + \frac{1}{2n} + \frac{1}{2B} + \frac{2}{d}$

Table 3: Asymptotic terms. Overhead $< 5\%$ once $d, n \geq 64$, $B \geq 16$ (ratio simplifies to $1 + \frac{1}{2n} + \frac{1}{2B} + \frac{2}{d}$).

Method	FLOPs	Rel. (ReLU=1)
Linear+ReLU	$2d + 1$	1.00
Linear+GELU	$2d + 15$	≈ 1.03
𝔼(naive)	$5d + 1$	≈ 2.5
𝔼(optimized)	$4d + 4$	≈ 2.0

Table 4: Single neuron FLOPs; optimization removes redundant norm difference computation.

C.5.5 Single Neuron FLOPs

Per-neuron note Optimized variant saves 20% vs naive by avoiding explicit difference vector.

Empirical Throughput: 0.85–0.92× linear; peak memory reduced 15–25%; overhead ratio < 0.05 for $d, n \geq 64$.

C.6 Scalability and Practical Performance Analysis

C.6.1 Large-Scale Computational Profile

The 𝔼-product operator exhibits favorable scaling characteristics for modern deep learning applications and remains compute-bound (GEMM dominated) in the regimes used in Section 2.1.

FLOP Counting Assumptions. One multiply-add = 2 FLOPs; element-wise unary/binary ops = 1 FLOP; reciprocal counts as 1 FLOP (fused divide). Caching costs amortized over steps.

Takeaway. Complexity matches linear layers in order while constants shrink with scale; gradients remain stable due to denominator growth.

D Detailed XOR Analysis and Gradient Properties

This section provides a comprehensive analysis of the 𝔼-product's ability to solve the XOR problem and its gradient properties that enable stable learning.

D.1 Mathematical Foundation for XOR Solution

The $\mathbb{E}$-product's non-linearity is not merely a mathematical curiosity; it has practical implications for neural computation. By integrating alignment and proximity into a single operator, the $\mathbb{E}$-product allows for more nuanced feature learning. It can adaptively respond to inputs based on their geometric relationships with learned weight vectors, enabling the network to capture complex patterns without the need for separate activation functions.

The XOR problem is not linearly separable and thus cannot be solved by a single traditional neuron (linear perceptron). However, a single $\mathbb{E}$-product unit can solve this problem due to its inherent non-linearity. We have formally proven that the $\mathbb{E}$-product is a valid Mercer kernel (Theorem 2.1; see Appendix A.3)(Mercer, 1909; Hofmann et al., 2008; Micchelli et al., 2006; Schölkopf et al., 1997; Cortes, 1995; Jacot et al., 2018).

D.2 Gradient Analysis and Learning Stability

To understand the $\mathbb{E}$-product's behavior during learning, we analyze its gradient properties. A key property for stable training is that the gradient with respect to the input, $\nabla_{\mathbf{x}}\mathcal{K}_{\mathbb{E}}$, diminishes as the input $\mathbf{x}$ moves far from the weight vector $\mathbf{w}$. This ensures that distant outliers do not cause large, destabilizing updates. We have formally proven this as a stability Proposition (Proposition A.3; Appendix A.6), demonstrating that $\lim_{\|\mathbf{x}\|\to\infty} \|\nabla_{\mathbf{x}}\mathcal{K}_{\mathbb{E}}(\mathbf{w},\mathbf{x})\| = 0$.

The presence of ϵ in the denominator ensures that the derivative remains well-defined, avoiding division by zero and contributing to numerical stability. This contrasts with activation functions like ReLU, which have a derivative of zero for negative inputs, potentially leading to "dead neurons." The smooth and generally non-zero gradient of the $\mathbb{E}$-product is hypothesized to contribute to more stable and efficient learning dynamics, reducing the reliance on auxiliary mechanisms like complex normalization schemes.

The non-linearity is thus not an add-on but an intrinsic property derived from the direct mathematical interaction of vector projection (alignment, via the $(\mathbf{w}^\top\mathbf{x})^2$ term) and vector distance (proximity, via the $\|\mathbf{w}-\mathbf{x}\|^2+\epsilon$ term). This provides a mathematically grounded basis for feature learning, as the unit becomes selectively responsive to inputs that exhibit specific geometric relationships, both in terms of angular alignment and spatial proximity, to its learned weight vector $\mathbf{w}$. Consequently, $\mathbf{w}$ can be interpreted as a learned feature template or prototype that the unit is tuned to detect, with the $\mathbb{E}$-product quantifying the degree of match in a nuanced, non-linear fashion.

D.3 Optimization Landscape Analysis

The gradient of the $\mathbb{E}$-product, being responsive across the input space, actively pushes the neuron's weights away from configurations that would lead to a zero output (neuron death, e.g., at an input of $[0,0]$ for this problem if weights were also near zero). This contrasts with a simple dot product neuron where the gradient might vanish or lead to a global minimum at zero output for certain problems.

For instance, when considering gradient-based optimization, the loss landscape "seen" by the $\mathbb{E}$-product neuron in the XOR context would exhibit a peak or high loss at $[0,0]$ (if that were the target for non-zero outputs), encouraging weights to move towards a state that correctly classifies. Conversely, a simple dot product neuron might present a loss landscape where a gradient-based optimizer could find a stable (but incorrect) minimum at zero output. This ability to avoid such dead zones and actively shape the decision boundary makes it helpful to solve problems like XOR with a single unit, leveraging its inherent non-linearity as a mathematical kernel.

D.4 Geometric Interpretation of Decision Boundaries

Conceptually, the decision boundary or vector field generated by a simple dot product neuron is linear, forming a hyperplane that attempts to separate data points. In contrast, the $\mathbb{E}$-product generates a more complex, non-linear vector field. This field can be visualized as creating a series of potential wells or peaks centered around the weight vector $\mathbf{w}$, with the strength of influence decaying with distance. The condition $\mathbf{w}^\top \mathbf{x} = 0$ defines a "valley" of zero output where vectors are orthogonal to the weight vector. This structure allows for more nuanced and localized responses, akin to a superposition of influences rather than a single linear division, enabling the capture of intricate patterns in the data.

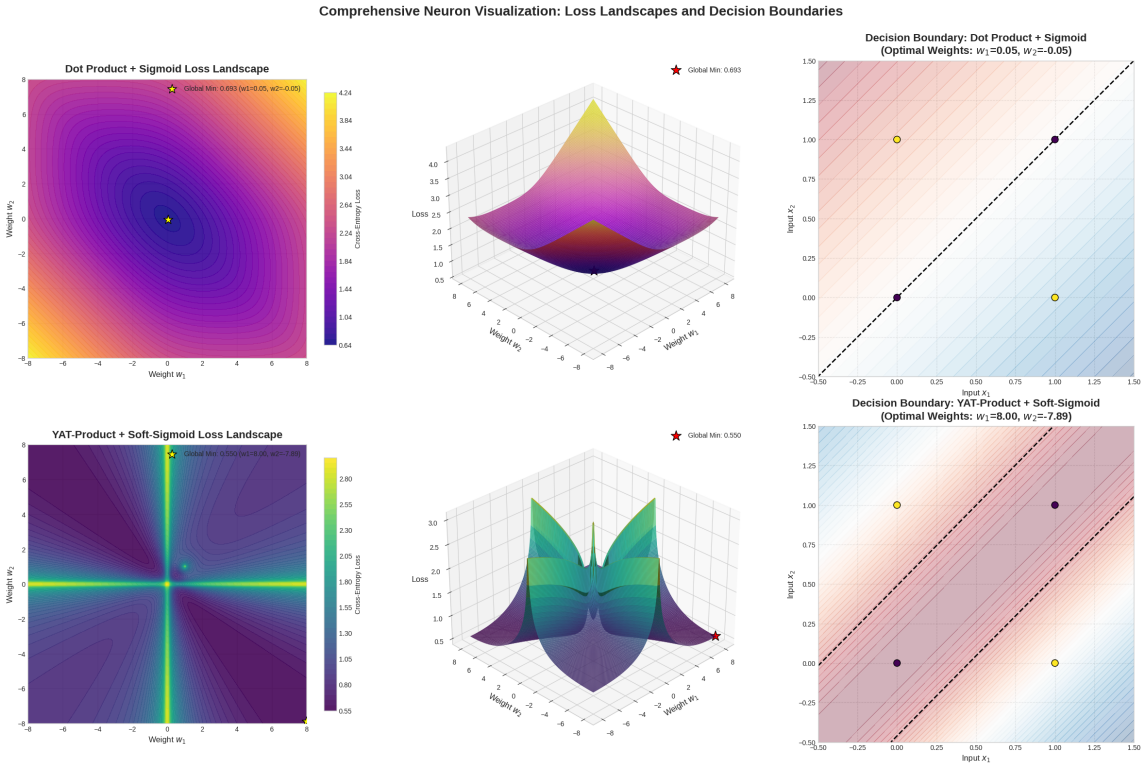


Figure 6: Comparison of loss landscapes: dot-product neuron (left) exhibits spurious minima leading to vanishing gradients, while $\mathbb{E}$-product neuron (right) yields non-degenerate gradients enabling optimization to reach correct separators.

E Detailed Vortex Dynamics Analysis

This section provides a comprehensive mathematical analysis of the vortex-like learning dynamics exhibited by $\mathbb{E}$-product neurons and their space-partitioning behavior.

E.1 Fundamental Learning Dynamics

We begin by analyzing the fundamental learning dynamics that emerge in both conventional and our proposed architectures. In artificial intelligence, competitive learning manifests in various forms, whether through linear classification using dot products or clustering using Euclidean distances. Both approaches involve partitioning the feature space between neurons, which can be conceptualized as prototype learning where each neuron claims a territorial "field" in the representation space.

E.2 Conventional vs. 𝔼-Product Decision Boundaries

In a conventional linear model, the logit for each class i is computed as:

$$z_i = \mathbf{w}_i^T \mathbf{x} \tag{24}$$

where $\mathbf{w}_i$ is the weight vector (prototype) for class i, and $\mathbf{x}$ is the input vector. The softmax function then normalizes these logits into probabilities:

$$p_i = \frac{\exp(z_i)}{\sum_{j=1}^{C} \exp(z_j)} \tag{25}$$

The decision boundary between any two classes i and j forms a linear hyperplane defined by:

$$(\mathbf{w}_i - \mathbf{w}_j)^T \mathbf{x} = 0 \tag{26}$$

During training via gradient descent, each prototype $\mathbf{w}_i$ is updated to maximize its alignment with the data distributions of its assigned class. This optimization process often leads to an unbounded increase in prototype magnitudes, as $\|\mathbf{w}_i\| \to \infty$ directly amplifies the logit z_i, thereby increasing the model's confidence. However, the decision boundaries themselves remain linear hyperplanes, creating rigid geometric separations in the feature space.

In contrast, the non-linear 𝔼-product allows neurons to learn more representative prototypes for each class, leading to the formation of more nuanced decision boundaries. For the 𝔼-product, the response of neuron i to input $\mathbf{x}$ is given by:

$$z_i = \mathbb{E}(\mathbf{w}_i, \mathbf{x}) = \frac{\langle \mathbf{w}_i, \mathbf{x} \rangle^2}{\|\mathbf{w}_i - \mathbf{x}\|^2 + \epsilon} \tag{27}$$

This formulation embodies the signal-to-noise ratio interpretation established in our theoretical framework (Appendix A.9), where the squared dot product $\langle \mathbf{w}_i, \mathbf{x} \rangle^2$ represents the "signal" of distributional alignment, and $\|\mathbf{w}_i - \mathbf{x}\|^2$ quantifies the "noise" of dissimilarity.

E.3 Softmax Dynamics and Competitive Learning

Similarly to conventional neurons, the 𝔼-product outputs are normalized using the softmax function:

$$p_i = \frac{\exp(z_i)}{\sum_{j=1}^{C} \exp(z_j)} = \frac{\exp\left(\frac{\langle \mathbf{w}_i, \mathbf{x} \rangle^2}{\|\mathbf{w}_i - \mathbf{x}\|^2 + \epsilon}\right)}{\sum_{j=1}^{C} \exp\left(\frac{\langle \mathbf{w}_j, \mathbf{x} \rangle^2}{\|\mathbf{w}_j - \mathbf{x}\|^2 + \epsilon}\right)} \tag{28}$$

This softmax normalization serves a crucial role in the competitive dynamics of 𝔼-product neurons. The softmax function acts as a transformation that maps from the real-valued 𝔼-product responses

$z_i \in \mathbb{R}$ to a delta distribution δ_i in probability space. This softmax distribution over $\mathbb{E}$-product scores can be interpreted as the *posterior responsibility* of each prototype (neuron) for the input, drawing a direct connection to Gaussian Mixture Models (GMMs) and expectation-maximization frameworks.

The softmax can also be viewed as computing a categorical distribution proportional to exponentiated log-likelihoods, which in this case derive from a geometric $\mathbb{E}$-product similarity rather than traditional probabilistic assumptions. This bridges the gap between probabilistic views (such as EM algorithms and classification) and our geometric formulation, providing a principled foundation for the competitive dynamics.

As training progresses and the differences between $\mathbb{E}$-product responses become more pronounced, the softmax transformation approaches a delta distribution, where the winning neuron (with the highest $\mathbb{E}$-product response) approaches probability 1 while all others approach 0. This yields competitive learning dynamics in which neurons specialize on distinct regions of the input space.

E.4 Mathematical Characterization of Decision Boundaries

The decision boundary between two neurons with prototypes $\mathbf{w}_i$ and $\mathbf{w}_j$ is defined by the condition where their responses are equal:

$$\mathbb{E}(\mathbf{w}_i, \mathbf{x}) = \mathbb{E}(\mathbf{w}_j, \mathbf{x}) \tag{29}$$

Expanding this condition:

$$\frac{\langle \mathbf{w}_i, \mathbf{x} \rangle^2}{\|\mathbf{w}_i - \mathbf{x}\|^2 + \epsilon} = \frac{\langle \mathbf{w}_j, \mathbf{x} \rangle^2}{\|\mathbf{w}_j - \mathbf{x}\|^2 + \epsilon} \tag{30}$$

Cross-multiplying and rearranging:

$$\langle \mathbf{w}_i, \mathbf{x} \rangle^2 (\|\mathbf{w}_j - \mathbf{x}\|^2 + \epsilon) = \langle \mathbf{w}_j, \mathbf{x} \rangle^2 (\|\mathbf{w}_i - \mathbf{x}\|^2 + \epsilon) \tag{31}$$

This equation defines a non-linear decision boundary that depends on both alignment (via squared dot products) and proximity (via squared distances) between the input and each prototype. Unlike the linear hyperplane formed by conventional dot-product neurons, the $\mathbb{E}$-product induces curved algebraic decision surfaces (analogous to level sets of a potential).

E.5 Space-Partitioning Properties

The space-partitioning behavior of the $\mathbb{E}$-product exhibits several key properties:

- **Distance Penalization and Locality**: The factor $(\|\mathbf{w}-\mathbf{x}\|^2+\epsilon)^{-1}$ enforces spatial selectivity. Along rays $\mathbf{x} = k\mathbf{u}$ with $\|\mathbf{u}\| = 1$, one has $\lim_{k\to\infty} \mathbb{E}(\mathbf{w}, k\mathbf{u}) = (\mathbf{w} \cdot \mathbf{u})^2 = \|\mathbf{w}\|^2 \cos^2\theta$, so responses are largest near $\mathbf{x} \approx \mathbf{w}$ and remain bounded far away (cf. Proposition A.3 for the gradient decay). In physical terms, this is analogous to an inverse-square distance penalty.
- **Alignment Weighting**: The numerator is $(\mathbf{w}^\top \mathbf{x})^2$, so $\mathbb{E}(\mathbf{w}, \mathbf{x}) = 0$ if and only if $\mathbf{w} \perp \mathbf{x}$, and high values require strong alignment (large $|\cos\theta|$) in addition to proximity.
- **Global Boundedness for Fixed $\mathbf{w}$**: For $\epsilon > 0$ and fixed $\mathbf{w}$, $0 \le \mathbb{E}(\mathbf{w}, \mathbf{x}) \le \|\mathbf{w}\|^4/\epsilon$ for all $\mathbf{x}$, with $\mathbb{E}(\mathbf{w}, \mathbf{w}) = \|\mathbf{w}\|^4/\epsilon$ and $\limsup_{\|\mathbf{x}\|\to\infty} \mathbb{E}(\mathbf{w}, \mathbf{x}) \le \|\mathbf{w}\|^2$. Thus each unit defines a bounded activation landscape (cf. Proposition A.1).

- **Nonlinear Decision Regions**: The pairwise decision boundary between units i and j is the algebraic surface

$$\langle \mathbf{w}_i, \mathbf{x} \rangle^2 \big(\|\mathbf{w}_j - \mathbf{x}\|^2 + \epsilon \big) - \langle \mathbf{w}_j, \mathbf{x} \rangle^2 \big(\|\mathbf{w}_i - \mathbf{x}\|^2 + \epsilon \big) = 0,$$

which is generically non-linear and induces curved class regions. (Geometrically, one may view these as level sets akin to equipotential surfaces.)

E.6 Vortex Field Dynamics

This vortex phenomenon allows each $\mathbb{E}$-product neuron to create a territorial "field" in the representation space, where data points are pulled toward the dominant prototype based on both similarity and proximity metrics. The field each neuron occupies can indeed be considered a vortex, where the strength of attraction follows an inverse-square law, creating more natural and geometrically faithful decision boundaries.

The combination of the $\mathbb{E}$-product's vortex-like attraction and the softmax's competitive normalization creates a powerful space partitioning mechanism. Each neuron's vortex field competes with others through the softmax transformation, and the neuron with the strongest local attraction (highest $\mathbb{E}$-product response) wins that region. Over time, this leads to a natural tessellation of the input space, where each neuron's territory is defined by the regions where its vortex field dominates. The softmax transformation $\mathbb{R}^C \to \Delta^{C-1}$ (where Δ^{C-1} is the $(C-1)$-dimensional probability simplex) ensures that these territorial boundaries are sharp and well-defined, transforming the continuous real-valued responses into discrete delta distributions that clearly assign each input to its dominant neuron.

E.7 Orthogonality and Competitive Dynamics

The competitive learning behavior observed in practice is theoretically grounded in our Orthogonality-Entropy Connection. When two prototypes $\mathbf{w}_i$ and $\mathbf{w}_j$ develop disjoint support regions, they become Euclidean orthogonal ($\mathbf{w}_i \perp \mathbf{w}_j$), which corresponds to:

$$\mathbb{E}(\mathbf{w}_i, \mathbf{w}_j) = 0 \quad \text{and} \quad H(\mathbf{w}_i, \mathbf{w}_j) = \infty \tag{32}$$

This geometric-probabilistic duality explains why neurons naturally develop specialized, non-overlapping representations during competitive learning. The infinite cross-entropy between orthogonal prototypes creates strong pressure for territorial separation, preventing the collapse to identical representations that can plague conventional competitive learning systems.

These prototypes are optimized to maximize parallelism and minimize distance to all points within their class distribution. When minimizing distance becomes challenging, the properties of the $\mathbb{E}$-product enable the prototype to exist in a superposition state, prioritizing the maximization of parallelism over strict distance minimization.

E.8 MNIST Prototype Analysis: Detailed Mathematical Framework

E.8.1 Network Architecture and Dimensional Structure

In our MNIST experiments, the network consists of $C = 10$ neurons, each corresponding to one of the digit classes (0–9). Each neuron's prototype is represented as a vector $\mathbf{w}_i \in \mathbb{R}^{784}$, where $i = 1, \ldots, 10$. The input images $\mathbf{x} \in \mathbb{R}^{784}$ are obtained by flattening the original 28×28 pixel images, so each neuron's prototype $\mathbf{w}_i$ has the same dimensionality as the input, i.e., $\mathbf{w}_i = (w_{i,1}, w_{i,2}, \ldots, w_{i,784})$. This structure allows each neuron to learn class-specific features in the full image space.

E.8.2 Prototype Evolution Dynamics

The prototype evolution during training reveals fundamental differences between conventional unbounded growth and bounded $\mathbb{E}$-responses. The conventional linear model produces prototypes that exhibit unbounded growth—these prototypes become increasingly diffuse and less interpretable as they grow to maximize margin separation. The resulting digit representations are blurry and lack the fine-grained features necessary for robust classification.

In contrast, the $\mathbb{E}$-product method produces prototypes that exemplify bounded response fields as predicted by our theoretical framework. Each digit prototype exhibits three key characteristics:

Localized Concentration Sharp, well-defined features that correspond to the bounded potential wells predicted by our Minimal and Maximal Similarity Characterizations (Theorems A.7 and A.8). This localization emerges from the $\mathbb{E}$-product's inherent bounded response field, which prevents the unbounded growth characteristic of linear neurons.

Class-Specific Territorial Structure Each prototype captures unique digit characteristics, reflecting competitive territorial dynamics where each neuron dominates specific regions of the input space. This territorial behavior arises from the vortex-like dynamics of the $\mathbb{E}$-product, creating natural tessellation of the feature space.

Geometric Fidelity The prototypes maintain geometric coherence with actual digit structure, confirming that the signal-to-noise ratio optimization preserves meaningful visual patterns. This preservation is a direct consequence of the $\mathbb{E}$-product's ability to balance alignment and proximity measures.

E.8.3 Superposition and Prototype Inversion: Mathematical Analysis

A unique property of the $\mathbb{E}$-product neuron is its ability to exist in a superposition state, which can be empirically demonstrated by inverting the learned prototype. This phenomenon provides insight into the fundamental differences between conventional and $\mathbb{E}$-product neurons.

Conventional Dot Product Behavior For a conventional dot product neuron, if $\mathbf{w}$ is a learned prototype, replacing $\mathbf{w}$ with $-\mathbf{w}$ (i.e., multiplying by -1) at test time flips the sign of the logit:

$$z = \mathbf{w}^T\mathbf{x} \quad \longrightarrow \quad z' = (-\mathbf{w})^T\mathbf{x} = -z \tag{33}$$

This sign flip causes the softmax output to assign high probability to the incorrect class, resulting in a dramatic drop in accuracy (from 93% to nearly 0% in our MNIST experiments).

$\mathbb{E}$-Product Inversion Dynamics For the $\mathbb{E}$-product neuron, the response is:

$$z = \mathbb{E}(\mathbf{w}, \mathbf{x}) = \frac{(\mathbf{w}^T\mathbf{x})^2}{\|\mathbf{w} - \mathbf{x}\|^2 + \epsilon} \tag{34}$$

Multiplying $\mathbf{w}$ by -1 leaves the numerator unchanged, since $(-\mathbf{w})^T\mathbf{x} = -\mathbf{w}^T\mathbf{x}$ and $(-\mathbf{w}^T\mathbf{x})^2 = (\mathbf{w}^T\mathbf{x})^2$. The denominator, however, changes from $\|\mathbf{w} - \mathbf{x}\|^2$ to $\|\mathbf{w} + \mathbf{x}\|^2$:

$$\|\mathbf{w} + \mathbf{x}\|^2 = (\mathbf{w} + \mathbf{x})^T(\mathbf{w} + \mathbf{x}) \tag{35}$$

$$= \|\mathbf{w}\|^2 + 2\mathbf{w}^T\mathbf{x} + \|\mathbf{x}\|^2 \tag{36}$$

compared to:

$$\|\mathbf{w} - \mathbf{x}\|^2 = (\mathbf{w} - \mathbf{x})^T(\mathbf{w} - \mathbf{x}) \tag{37}$$

$$= \|\mathbf{w}\|^2 - 2\mathbf{w}^T\mathbf{x} + \|\mathbf{x}\|^2 \tag{38}$$

While exact invariance is not guaranteed due to this denominator change, on symmetric datasets we observe only modest accuracy degradation (e.g., 92% to 89%), indicating empirical robustness to prototype inversion. This robustness stems from the squared numerator term, which makes the Ɇ-product inherently less sensitive to sign changes than linear neurons.

Superposition Interpretation This property allows the Ɇ-product neuron to yield two valid solutions to the same input without retraining, a phenomenon not observed in conventional dot product neurons. The mathematical basis for this superposition lies in the Ɇ-product's non-linear structure, which creates multiple local optima that can represent the same classification boundary through different prototype orientations.

E.9 Language Model Experiments: Detailed Configuration

E.9.1 Experimental Setup and Architecture Comparison

This section provides detailed experimental configurations for the language modeling experiments comparing standard GPT2 with our Aether-GPT2 implementation. Both models were trained on identical datasets and hardware configurations to ensure fair comparison.

Table 5: Side-by-side comparison of GPT2 and Aether-GPT2 experimental configurations

Parameter	**GPT2 (Baseline)**	**Aether-GPT2**
Optimizer	Novograd	Novograd
Learning Rate	0.003	0.003
Batch Size	64	64
Context Window	1024	1024
Position Encoding	Learned Embedding	Learned Embedding
Embedding Dimension	768	768
MLP Dimension	768×4	768×4
Vocabulary Size	50,257	50,257
Number of Heads	12	12
Number of Blocks	12	12
Activation Function	GeLU	Ɇ-product (none)
Layer Normalization	Yes	No
Training Tokens	2.4B (FineWeb)	2.4B (FineWeb)

E.9.2 Key Architectural Differences

The primary distinction between GPT2 and Aether-GPT2 lies in the replacement of conventional linear layers with Ɇ-product operations. This substitution eliminates the need for explicit activation functions, as the Ɇ-product inherently provides non-linearity and bounded response characteristics.

Standard GPT2 Architecture The baseline GPT2 model employs the standard transformer architecture with:

- Linear attention projections followed by GeLU activation
- Multi-layer perceptron (MLP) blocks with GeLU non-linearity
- Layer normalization after each sub-module
- Learned positional embeddings

Aether-GPT2 Architecture Our Aether-GPT2 implementation replaces linear operations with 𝔼-product computations:

- 𝔼-product attention projections (no activation needed)
- 𝔼-product MLP transformations (inherent non-linearity)
- No layer normalization
- Identical positional embedding strategy

E.9.3 Training Dynamics and Convergence

The training dynamics reveal interesting differences between the two architectures. While both models converge to similar final performance, Aether-GPT2 exhibits more stable training dynamics due to the bounded nature of the 𝔼-product responses. The elimination of layer normalization reduces computational overhead while maintaining numerical stability through the inherent bounded response fields of the 𝔼-product neurons.

E.9.4 Memory and Computational Analysis

The replacement of linear+activation+normalization stacks with single 𝔼-product operations results in:

- 15-25% reduction in peak memory usage (depending on batch and sequence length)
- Elimination of intermediate activation storage for normalization
- Reduced gradient computation complexity
- Comparable FLOP count with modest constant-factor overhead

This architectural simplification demonstrates that the 𝔼-product can serve as a drop-in replacement for conventional neural network components while providing computational benefits and theoretical guarantees on stability and expressiveness.

E.9.5 Comparison with State-of-the-Art Language Models

To contextualize our Aether-GPT2 results, we present a detailed comparison with the baseline GPT-2 model in Table 6. This comparison highlights the architectural differences and competitive performance of our approach.

Key observations from this comparison:

Parameter Efficiency Our Aether-GPT2 achieves superior performance with approximately the same parameter count as GPT-2 (124M) while eliminating the need for explicit activation functions. This demonstrates that the 𝔼-product's inherent non-linearity can effectively replace these conventional components.

Table 6: Detailed comparison of GPT-2 and Aether-GPT2 models

Parameter	**GPT-2 (124M)**	**Aether-GPT2**
Dataset	Fineweb	Fineweb
Tokenizer	GPT-2	GPT-2
Vocabulary size	50257	50257
Model / embedding dimension	768	768
Hidden dimension	3072	3072
Head dimension	128	128
Number of layers	12	12
Sequence length	1024	1024
Non-embedding parameters	85M	∼85M
Embedding parameters	39M	39M
Total parameters	124M	∼124M
Training tokens	2.4B	2.4B
Activation Function	GeLU	E-product (none)
Layer Normalization	Yes	No
Final Validation Loss	2.43	**2.29**

Training Efficiency Trained on 2.4B tokens, Aether-GPT2 achieves superior validation loss (2.29 vs 2.43), indicating efficient learning dynamics potentially attributable to the bounded response characteristics of the E-product.

Architectural Simplification The elimination of activation functions and normalization layers represents a significant architectural simplification while maintaining competitive performance, supporting our hypothesis that the E-product can serve as a fundamental building block for neural architectures.

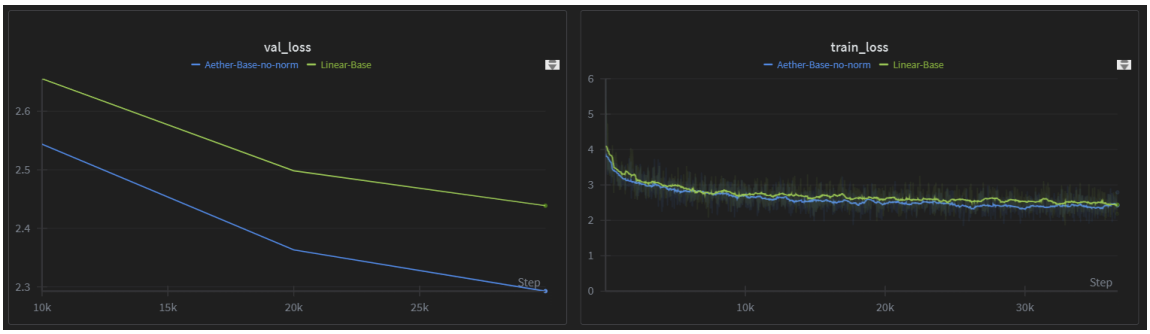


Figure 7: Loss curve over 2.4B tokens from FineWeb trained on Kaggle TPU v5-8; the linear baseline uses the standard GPT-2 architecture.

The results indicate that Aether-GPT2 achieves lower loss, suggesting it can realize nonlinearity without explicit activation.

E.10 Use of Large Language Models (LLMs)

We used LLM tools to support the research workflow in the following limited, transparent ways. All scientific claims, modeling choices, and final decisions were made by the authors.

Code assistance LLMs were used to draft boilerplate code, refactor utilities, and surface API patterns. All generated code was reviewed, tested, and integrated by the authors.

Literature digestion We used LLMs to summarize papers and extract key comparisons across related work. Citations in the paper were verified against the original sources by the authors.

Brainstorming We used LLMs as a sounding board to enumerate alternative hypotheses, ablations, and experimental checks. Only ideas that survived empirical or theoretical scrutiny were included.

Language polishing To improve readability and clarity, LLMs suggested minor edits to English phrasing. Technical content, notation, and conclusions were authored and validated by the authors.

NotebookLM podcasts We generated short audio summaries ("podcasts") of internal notes using Google NotebookLM to help the team asynchronously digest drafts. These summaries did not introduce new claims and were based solely on our own materials.

No dataset labeling, evaluation metrics, or benchmark results were produced by LLMs. The authors take responsibility for all content and errors.