NON-EUCLIDEAN HARMONIC LOSSES

Anonymous authors
Paper under double-blind review

ABSTRACT

Cross-entropy loss has long been the standard choice for training deep neural networks, yet it suffers from interpretability limitations, unbounded weight growth, and inefficiencies that can contribute to costly training dynamics. Recent work introduced *harmonic loss*, a distance-based alternative grounded in Euclidean geometry, which improves interpretability and mitigates phenomena such as *grokking*, also known as delayed generalization on the test set. However, the study of harmonic loss remains narrow: only Euclidean distance is explored, and no systematic evaluation of computational efficiency or sustainability was conducted. In this paper, we extend harmonic loss by systematically investigating a broad spectrum of distance metrics as replacements for the Euclidean distance. We comprehensively evaluate *distance-tailored harmonic losses* on both vision backbones and large language models. Our analysis is framed around a three-way evaluation of *model performance*, *interpretability*, and *sustainability*. On vision tasks, cosine distances provide the most favorable trade-off, consistently improving accuracy while lowering carbon emissions, whereas Bray-Curtis and Mahalanobis further enhance interpretability at varying efficiency costs. On language models, cosine-based harmonic losses markedly improve gradient and learning stability, strengthen representation structure, and reduce emissions relative to cross-entropy and Euclidean heads. Our code is available at: `https://anonymous.4open.science/r/rethinking-harmonic-loss-5BAB/`.

1 INTRODUCTION

Cross-entropy is the *de facto* loss function for classification tasks. However, it has shortcomings in terms of model interpretability and training dynamics. Cross-entropy training provides no inherent meaning to the learned weight vectors (they serve as abstract parameters rather than intuitive prototypes) and can drive those weights to grow without bound in pursuit of confident predictions Baek et al. (2025). This unbounded weight growth can lead to phenomena like *grokking*: a delayed generalization where the model only closes the train–test performance gap after extensive overtraining Power et al. (2022). Moreover, in high-stakes applications where transparency is critical (e.g., healthcare or finance), the opaque nature of cross-entropy–trained models poses challenges for trust and error diagnosis. These issues motivate the exploration of alternative loss functions that may yield more *interpretable*, *efficient*, and *robust* model behavior.

Recently, *harmonic loss* was proposed as an alternative training objective to address some of these concerns Baek et al. (2025). Harmonic loss replaces the conventional inner-product logits and softmax normalization with a distance-based formulation: model predictions are derived from the distances between the sample's representation and *class prototype* vectors (learned weight vectors for each class). Intuitively, this means that a model is trained to bring each sample closer to its correct class center in the feature space rather than simply increasing a classification score. This approach endows the learning process with two key properties: *i) scale invariance* – distance comparisons do not depend on vector norm, and *ii) finite convergence point* – training aims for a distance of zero to the correct prototype, rather than driving logits to $\pm\infty$ as in cross-entropy. As a result, each class weight converges to an anchor point that can be interpreted as the *center* of that class's feature distribution. Empirically, Baek et al. (2025) demonstrated that harmonic loss can close the train–test gap faster and yield more interpretable representations than cross-entropy. For example, the learned weight vectors in a harmonic-loss model directly reflect class prototypes, making them semantically meaningful. Models trained with harmonic loss were shown to require less data to generalize and to

mitigate grokking, all while achieving competitive or better accuracy on vision and language benchmarks Baek et al. (2025). These findings suggest that *distance-based loss functions* are a promising direction for improving both performance and transparency in deep learning.

However, research on harmonic loss has been limited in scope so far. Baek et al. (2025) focused exclusively on Euclidean distance as the metric for their loss function and did not examine the broader impacts on computational efficiency or energy consumption. On the other hand, distance-based metrics have been explored in other contexts and problems. Notably, Coil et al. (2025) investigated a wide range of distance measures for a problem of change point detection in concept-drift scenarios for anomaly detection. Their study found that the choice of distance metric can drastically affect both the accuracy and efficiency of detecting distribution shifts. For instance, replacing a costly metric (e.g., Wasserstein) with simpler alternatives yielded comparable detection performance at substantially lower computational cost. This evidence that "metric matters" in learning algorithms raises a natural question: *might other distance measures offer advantages over Euclidean in a harmonic-loss setting?* To date, no work has evaluated harmonic loss with distance metrics beyond Euclidean, nor benchmarked their impacts across different domains.

In this paper, we present the first comprehensive study of *custom distance-based loss functions* in deep learning classification, extending the harmonic loss framework to a variety of distance measures across multiple problem domains. We experiment with a rich set of distance metrics, including Manhattan, Euclidean, Chebyshev, Minkowski, and cosine distance, as well as specialized metrics such as Hamming, Canberra, Bray-Curtis, and Mahalanobis. These metrics are integrated as drop-in replacements for Euclidean distance in the harmonic loss formulation.

We evaluate harmonic loss with each distance metric on two heterogeneous task families: *image classification* (MLP, ResNet, PVT) and *language modeling* with transformer-based LLMs (GPT-2, BERT, and others). This diversity enables us to assess whether certain distance-based losses consistently outperform cross-entropy and Euclidean harmonic loss on metrics of *effectiveness*, *efficiency*, and *explainability*. Specifically, we pursue the following research questions:

RQ1 (Model Performance): Do distance-based loss functions offer higher accuracy or faster convergence compared to cross-entropy and Euclidean harmonic loss?

RQ2 (Interpretability): Do models trained with distance-based losses exhibit more interpretable representations than those trained with cross-entropy?

RQ3 (Efficiency & Sustainability): If a custom distance-based loss outperforms cross-entropy, does it do so without incurring higher computational cost? We track training time, resource utilization, and energy consumption to assess the *Green AI* perspective Schwartz et al. (2019).

By addressing these questions, our aim is to explore a *three-way trade-off* between *accuracy, interpretability, and sustainability* in the training process of deep learning models. Previous work has typically optimized one or two of these aspects in isolation: for instance, improving accuracy at the cost of enormous compute, known as "Red AI" (Schwartz et al., 2019), or simplifying models for interpretability while losing accuracy. In contrast, we seek solutions that can improve predictive performance while also yielding lower energy usage and more transparent models.

Contributions. This paper introduces *distance-tailored harmonic losses* and provides an extensive empirical and analytical evaluation of their merits. To our knowledge, this is the first work to: i) extend the harmonic loss beyond Euclidean distance and benchmark a wide spectrum of metrics on both vision and NLP tasks, ii) assess the carbon footprint and resource usage of different loss functions in a controlled setting, and iii) investigate interpretability outcomes of distance-based losses. We also offer preliminary theoretical insights into how different distance metrics influence the geometry of the learned model (e.g., relating L_1 losses to median-based class centers vs. L_2 to mean-based centers), which could inform the selection of an optimal loss for a given objective.

2 HARMONIC LOSS

Harmonic loss replaces the conventional inner-product logits and softmax normalization with a distance-based formulation: model predictions are derived from distances between the sample's representation and *class prototype* vectors (the learned weight vectors for each class). Intuitively,

this means a model is trained to bring each sample closer to its correct class center in the feature space, rather than simply pushing up a classification score.

From Baek et al. (2025), given the training set $\{(x_i, y_i)\}_{i=1}^n$ with $y_i \in \{1, ..., K\}$ and class prototypes $\{\mathbf{w}_c\}_{c=1}^K \in \mathbb{R}^d$, the harmonic logit is the ℓ_2 distance between $\mathbf{w}_i$ and $\mathbf{x}$, i.e., $d_i = \|\mathbf{w}_i - \mathbf{x}\|_2$. Then, the harmonic probabilities are given by:

$$p_k(x_i) \;=\; \frac{d_i^{-n}}{\sum_{j=1}^K d_i^{-n}}, \tag{1}$$

where the harmonic exponent n is a hyperparameter that controls the heavy-tailedness of the probability distribution. The Harmonic loss is then given by:

$$\mathcal{L}(\{\mathbf{w}_k\}) = -\sum_{i=1}^{n} \log p_k(x_i). \tag{2}$$

This approach endows the learning process with two key properties: i) *scale invariance:* distance comparisons do not depend on the overall norm of $\mathbf{h}$ or $\mathbf{w}_c$, in contrast to inner-product logits; and ii) *finite convergence point:* optimization seeks a distance of zero to the correct prototype, rather than driving logits to $\pm\infty$ as in cross-entropy.

As a result, each class weight converges to an anchor point that can be interpreted as the *center* of that class's feature distribution. Empirically, Baek et al. (2025) demonstrated that harmonic loss can close the train–test gap faster and yield more interpretable representations than cross-entropy. For example, the learned weight vectors in a harmonic-loss model directly reflect class prototypes, making them semantically meaningful. Models trained with harmonic loss were also shown to require less data to generalize and to mitigate grokking, all while achieving competitive or better accuracy on both vision and language benchmarks. These findings suggest that *distance-based loss functions* are a promising direction for improving performance and transparency in deep learning.

3 Non-Euclidean Harmonic Losses

Our framework introduces *non-Euclidean harmonic losses* as a generalization of the harmonic loss, and as a replacement for conventional cross-entropy training. The idea is that, in Eq. (1), the Euclidean distance $d_i = \|\mathbf{w}_i - \mathbf{x}\|_2$ is replaced by a non-Euclidean distance.

3.1 Class Prototypes, Distances, and Distance-Based Harmonic Loss function

Each class $c \in \{1, \ldots, K\}$ is associated with a *prototype vector* $\mathbf{w}_c \in \mathbb{R}^d$. Given a sample $\mathbf{h}$, we compute its distance to all prototypes via a chosen metric $d(\cdot, \cdot)$. We extend the Euclidean formulation of harmonic loss (Baek et al., 2025) with the following distances:

Euclidean. $d_{\text{euclidean}}(\mathbf{h}, \mathbf{w}) = \|\mathbf{h} - \mathbf{w}\|_2$. Baseline Euclidean distance between feature and prototype.

Manhattan (L1). $d_{\text{manhattan}}(\mathbf{h}, \mathbf{w}) = \|\mathbf{h} - \mathbf{w}\|_1$. Emphasizes absolute differences, making it more robust to outliers (Keeling & Kunisch, 2016; Ye et al., 2012; Giloni & Padberg, 2003). It can stabilize training and reduce unnecessary computations, thereby lowering energy costs.

Chebyshev (L∞). $d_{\text{chebyshev}}(\mathbf{h}, \mathbf{w}) = \|\mathbf{h} - \mathbf{w}\|_\infty$. Captures the maximum coordinate deviation, offering a highly interpretable measure of the most discriminative feature dimension. Its simplicity makes it computationally efficient.

Minkowski (Lp). $d_{\text{minkowski}}(\mathbf{h}, \mathbf{w}; p) = \|\mathbf{h} - \mathbf{w}\|_p$. Generalizes both L1 and L2, with tunable p enabling a trade-off between robustness and sensitivity. This flexibility allows tailoring the loss to dataset complexity, improving accuracy while balancing sustainability.

Cosine. $d_{\text{cosine}}(\mathbf{h}, \mathbf{w}) = 1 - \frac{\mathbf{h}^\top \mathbf{w}}{\|\mathbf{h}\|_2 \|\mathbf{w}\|_2}$. Ignores magnitude and instead measures angular similarity, making it particularly effective in high-dimensional embeddings (e.g., CNNs, Transformers) (Reimers & Gurevych, 2019; Deng et al., 2019; Wang et al., 2018; Sun et al., 2016; Karpukhin et al., 2020). This often improves generalization with minimal computational overhead.

Hamming. $d_{\text{hamming}}(\mathbf{h}, \mathbf{w}) = \frac{1}{d}\sum_{i=1}^{d} \mathbf{1}_{\{h_i \neq w_i\}}$. Counts mismatches directly, providing highly interpretable signals. With soft or gumbel relaxations, it becomes suitable for continuous embeddings and can reduce emissions when binary approximations are leveraged.

Canberra. $d_{\text{canberra}}(\mathbf{h}, \mathbf{w}) = \sum_{i=1}^{d} \frac{|h_i - w_i|}{|h_i| + |w_i| + \varepsilon}$. Normalizes differences by feature magnitudes, enhancing sensitivity to small but meaningful variations. This can improve performance on fine-grained tasks while stabilizing optimization.

Bray–Curtis. $d_{\text{bray-curtis}}(\mathbf{h}, \mathbf{w}) = \frac{\sum_{i=1}^{d} |h_i - w_i|}{\sum_{i=1}^{d} (|h_i| + |w_i|) + \varepsilon}$. Captures proportional differences across feature vectors, making it efficient and interpretable for compositional data (Fuschi et al., 2025; Chao et al., 2010; Song et al., 2020). It often balances accuracy with sustainability better than covariance-based measures.

Mahalanobis. $d_{\text{mahalanobis}}(\mathbf{h}, \mathbf{w}; \Sigma) = \sqrt{(\mathbf{h} - \mathbf{w})^\top \Sigma^{-1} (\mathbf{h} - \mathbf{w})}$. Incorporates feature correlations, offering superior accuracy in complex datasets and deep CNNs (Pang et al., 2018; Lee et al., 2018; Gómez-Silva et al., 2021; Omara et al., 2021). Although covariance estimation may increase computational cost, its interpretability and classification power justify the trade-off in high-capacity models.

In our work, we generalize harmonic loss by replacing the Euclidean distance used to calculate the harmonic logit with some other distance measure. Overall, compared to cross-entropy, these distance-based harmonic losses reduce reliance on probabilistic normalization and can lower the number of required operations. This translates into potential accuracy gains, reduced carbon emissions, and improved interpretability, depending on the chosen distance and backbone.

A formal treatment of our distance–based probabilistic layer is provided in Appendix A. There we generalize the harmonic-loss analysis to broad distance families and prove: i) *scale invariance* and the existence of *finite* minimizers under 1–homogeneous distances (Theorem 1), and ii) a *margin-style PAC–Bayes generalization bound* whose finiteness follows from the finite–norm solution (Theorem 2). These results clarify when geometry choices are well-posed and why the resulting classifiers admit standard generalization guarantees.

4 Experiments and Discussion

4.1 Training and Evaluation

Datasets. We evaluate on three *vision* benchmarks (MNIST, CIFAR-10, CIFAR-100) and one *language* corpus (OpenWebText).

Vision. We consider a **Simple MLP** with two hidden layers (512, 256, ReLU), a **Simple CNN** (two 3×3 conv blocks with $[32, 64]$ channels and 2×2 max-pooling, then a 128-dim FC), **ResNet-50** (standard $[3, 4, 6, 3]$ bottleneck stages; for small inputs we remove the initial max-pool and use a 3×3 stride-1 stem), and **PVTv2-B0** (four hierarchical stages with overlapping patch embeddings; output pooled to a 256-dim vector).

Language. We study three Transformer families: **GPT**-style (decoder-only causal LM), **BERT** (encoder-only masked LM with 15% masking), and **Qwen2**-style decoders.

Optimization. Unless noted, models are trained from scratch with Adam/AdamW-style optimizers (weight decay, (β_1, β_2) as configured), cosine learning-rate decay with linear warmup, mixed precision (FP16/BF16 when available), and gradient accumulation. We apply gradient clipping, dataset-specific schedulers, and early stopping with *dataset-specific patience* and a minimum improvement threshold ($\Delta_{\min}$). For fairness, all harmonic heads and the baseline share the same backbone, batch size, scheduler, and data order. Additional details about optimization are reported in Appendix C.1.

Model Performance. For vision tasks, we report average Accuracy and F1. For language task, we report Gradient Stability (higher values indicate smoother training with lower variance in gradient norms), Model Health ($-\Delta$ `model_collapse_score`): higher values means stronger resistance to representation collapse, Clipping Quality (higher scores indicate healthier gradient flow without extreme values requiring intervention), and Learning Quality (captures both how well the model learned and how much it improved).

Interpretability. We probe whether learned prototypes/weights act as class centers and whether features become more structured by computing **PCA explained variance** on the penultimate features: (i) **PC2 EV** (variance explained by the top two PCs), and (ii) **PCA@90%** (dimensions required to reach 90% variance). Lower PCA@90% and higher PC2 EV indicate more concentrated, low-dimensional structure. For language, we report *PCA5*: Δ variance explained by the top 5 principal components of final hidden states (causal LM: last token; MLM: masked positions); higher values implies more concentrated, low-dimensional structure.

Sustainability. We instrument training with *CodeCarbon* to log *duration*, *energy*, and CO_2 *emissions*. Emissions are reported per run and *differentially* vs. the cross-entropy baseline (grams CO_2; negative means greener-than-baseline). We aggregate by (dataset, backbone, distance) and also report cumulative figures across seeds. For language, we also report Speed ($-\Delta$`time_to_90_percent`): higher values denotes fewer steps to reach 90% of final performance.

To isolate the effect of the *loss geometry*, we *only* swap the classifier head (linear vs. distance-based) while keeping: backbone weights initialization scheme, data preprocessing/augmentation, optimizer and LR schedule, batch size, number of epochs, early-stopping rule, and randomness controls (seeds). For ResNet-50/PVT we use identical augmentation; for LLMs we use the same context length L, optimizer, and schedule across heads. We run multiple seeds and report means. Exact architectures and preprocessing pipelines are detailed in Appendix C.1. Full hyperparameter grids (including head-specific parameters Θ, e.g., p for Minkowski or covariance settings for Mahalanobis) are provided in Appendix D.

This unified protocol lets us *systematically* test how replacing the Euclidean harmonic head with alternative distances impacts: i) final model performance, ii) representation structure and prototype semantics, and iii) measured energy and carbon footprint.

4.2 Vision: Radar Plots

Figure 1 provides a multi-criteria comparison of distance-based harmonic loss variants across MNIST, CIFAR-10, and CIFAR-100 with MLP, CNN, and ResNet50 backbones. The radar plots allow us to visualize trade-offs among model performance, interpretability, and sustainability.

RQ1: Model Performance (F1, Test Accuracy). Across architectures, **cosine-based** harmonic losses are the most reliable performers. On MNIST and CIFAR-10, cosine typically achieves the highest or near-highest accuracy and F1 and reaches those plateaus with smooth training dynamics. On CIFAR-100, where capacity and data difficulty are more significant, cosine remains competitive and frequently leads in CNN and ResNet50. Moreover, it is often among the top curves for PVT as well. Euclidean and cross-entropy baselines are stable references, but are rarely the best once a distance-based head is available. Other distances (e.g., Bray–Curtis, Mahalanobis) can match or exceed cosine in isolated settings, but do so less consistently.

RQ2: Interpretability (PC2 EV, PCA 90%). Non-Euclidean distances shape the geometry of the learned representation in a consistent way. **Bray–Curtis** and **Chebyshev (standard)** produce the *largest* explained variance (PC2 EV) and reduce the number of principal components required to cover 90% of the variance, indicating compact, prototype-aligned feature spaces. **Cosine** generally provides substantial EV gains while preserving strong performance, yielding a favorable accuracy–interpretability balance. **Mahalanobis** often achieves extreme concentration (very high EV), reflecting strong alignment to class structure, but this effect can co-occur with less stable optimization on harder datasets.

RQ3: Sustainability (Duration/Epoch, Emissions). Distance choice also influences efficiency. **Cosine** tends to be neutral-to-favorable on emissions relative to Euclidean/cross-entropy across backbones, with modest or no penalties in Duration/Epoch. **Mahalanobis** variants frequently incur higher emissions, reflecting covariance-related computation, while **Bray–Curtis** shows mixed but often moderate cost. Changes in Duration/Epoch are smaller than changes in accuracy or EV (the final layer is light compared to the backbone), yet cumulative emissions still separate distances meaningfully over many epochs and runs.

Three significant regularities emerge: i) **Cosine** is the best *all-around* choice: consistently high accuracy/F1, clear gains in EV, and neutral-to-lower emissions across MLP, CNN, ResNet50, and PVT;

ii) **Bray–Curtis** (standard/normalized) is the best option for *interpretability*: it reliably increases EV and reduces PCA 90% dimensionality; accuracy effects are positive but more configuration-dependent; iii) **Mahalanobis** emphasizes *representation clarity* (very high EV) at a potential sustainability cost; it is attractive when prototype sharpness is prioritized over efficiency.

4.3 LANGUAGE: RADAR PLOTS

Figure 2 summarizes the effect of distance-tailored harmonic losses on *BERT*, *GPT*, and *Qwen*-style decoders across the three perspectives. Scores are normalized so that larger areas indicate more desirable behavior.

RQ1: Model Performance (Health, Stability, Clipping). Across architectures, **cosine-based** harmonic losses are the most consistently strong on stability-oriented axes. In BERT, cosine heads improve *Gradient Stability* and *Learning Stability* while maintaining high *Model Health*; *Clipping Quality* remains favorable, indicating fewer extreme updates. In GPT, cosine with a *very low* temperature can overconstrain the logits and slightly degrade learning stability, whereas *cosine (simple)* and *Minkowski (*p*=2)* provide a steadier training signal. Qwen exhibits similar trends: *Minkowski* and *Euclidean* remain competitive in stability, but cosine variants typically match or exceed them in the combined performance envelope. Overall, non-Euclidean distances reduce gradient volatility and collapse symptoms relative to the Euclidean head and the cross-entropy baseline.

RQ2: Interpretability (PCA Structure). Non-Euclidean distances tend to *concentrate* the token representations. Cosine and Minkowski enlarge the *PCA Structure* slice (greater variance explained by a few components), suggesting more organized, prototype-aligned hidden spaces. BERT benefits the most, reflecting its bidirectional encoding and MLM target, while GPT and Qwen still show clear gains over cross-entropy and Euclidean. These results mirror the vision findings: distances that compare *directions* (cosine) or emphasize ℓ_p *geometry* (Minkowski) yield more low-dimensional structure.

RQ3: Sustainability (Emissions). The *Emissions* wedges indicate that replacing the linear classifier with a distance head introduces little overhead and can even be *greener* than cross-entropy. In GPT and Qwen, the cross-entropy baseline typically occupies the largest emissions footprint, whereas cosine and Minkowski are neutral-to-favorable. Extremely sharp cosine temperatures may trade off stability for small emission gains; moderate settings avoid this. Since the head is lightweight relative to the Transformer, differences accumulate through faster convergence and steadier optimization rather than per-step cost.

Overall, Cosine is the most reliable all-around choice for LLMs, improving stability and representation structure with neutral or reduced emissions. Minkowski (p=2) offers a robust alternative when cosine (with overly low temperature) harms stability, preserving strong PCA structure. The cross-entropy baseline tends to be the least sustainable, while Euclidean harmonic loss is a solid reference but rarely dominates once distance-based heads are allowed.

5 RELATED WORK

Loss functions for classification. The majority of classification models are trained with cross-entropy loss due to its empirical effectiveness and probabilistic interpretation. However, it only cares about separating classes, not about how the representations are separated, often yielding features that are separable but not necessarily interpretable. Over the years, alternative loss functions have been proposed to address these limitations. Metric learning losses such as contrastive and triplet loss train models to preserve distances between examples, but require sampling strategies that add training complexity. Boudiaf et al. (2020) propose a unifying mutual information framework connecting cross-entropy to standard pairwise losses, showing that cross-entropy implicitly bounds pairwise distance objectives. These insights motivate a deeper theoretical understanding of distance-based training. Regularization-based approaches such as *center loss* (Wen et al., 2016) explicitly encourage compact intra-class clusters and large inter-class separation. These works foreshadow the idea that directly leveraging distances to class prototypes can improve representation quality. Angular margin losses such as AMC-Loss in Choi et al. (2020) introduce geometric constraints on angular separations to enhance interpretability via hyperspherical metrics. Orthogonal Projection Loss (OPL) introduced by Ranasinghe et al. (2021) encourages inter-class orthogonality and intra-

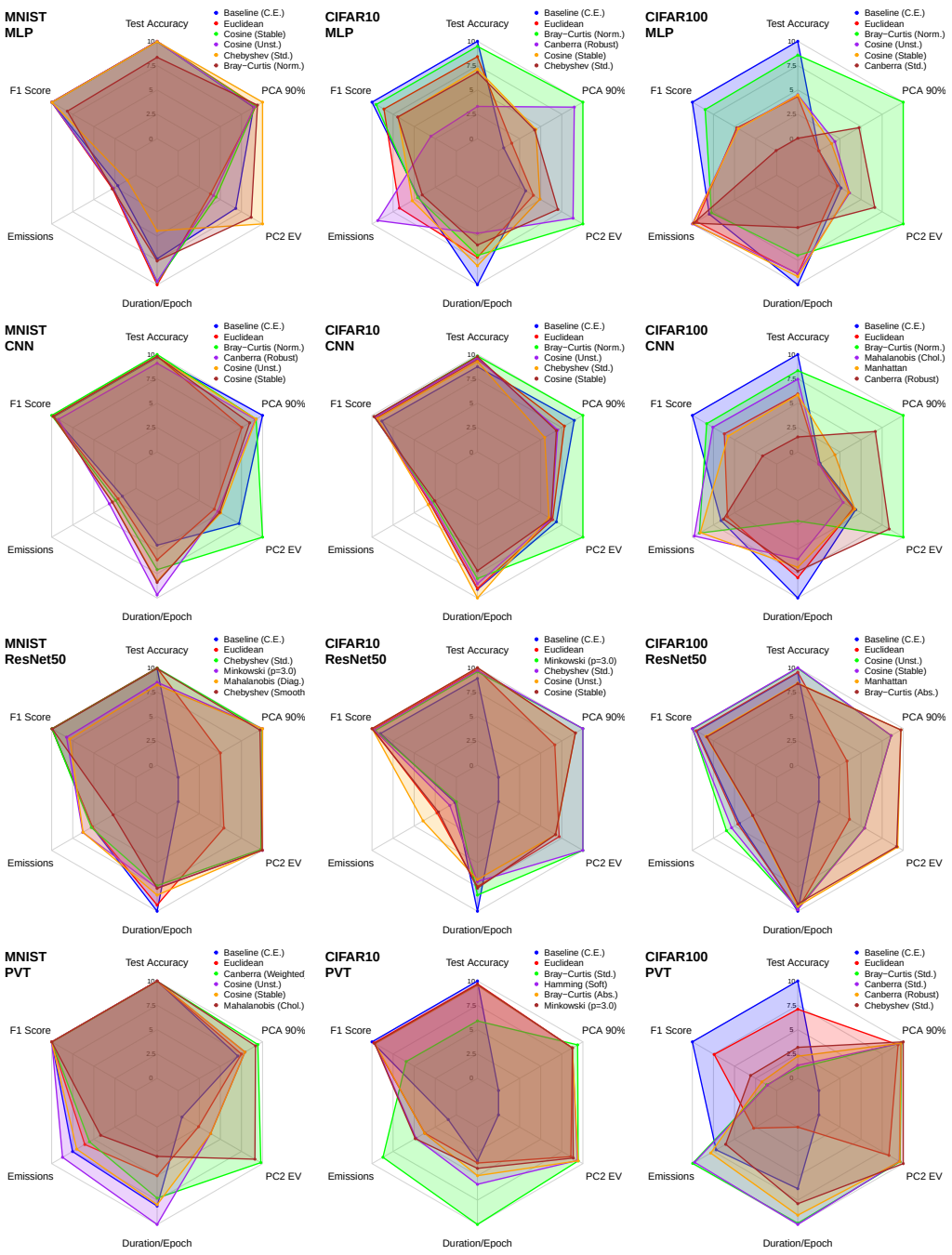

Figure 1: Vision: Radar plots: 1) *Model Performance* (F1, Accuracy); 2) *Interpretability* (PC2 EV, PCA 90%), and 3) *Sustainability* (Duration/Epoch, Emissions). Plots feature Baseline (Cross-Entropy), Euclidean harmonic, and the four top-performing non-Euclidean harmonic losses.

class cohesion without sampling overhead. Several studies have assessed how loss functions affect neural network performance. Miller et al. (2021) introduce *Class Anchor Clustering* (CAC) loss that encourages tight class clusters centered on anchored prototypes, enhancing distance-based open-set classification performance. This approach aligns with the prototype-centered philosophy underlying harmonic loss. Cho et al. (2019) analyzed how eight loss functions impact neural network accuracy and convergence speed, finding that additive-margin softmax loss resulted in the fastest convergence and highest performance on multiple datasets. Janocha & Czarnecki (2017) assessed 12 loss

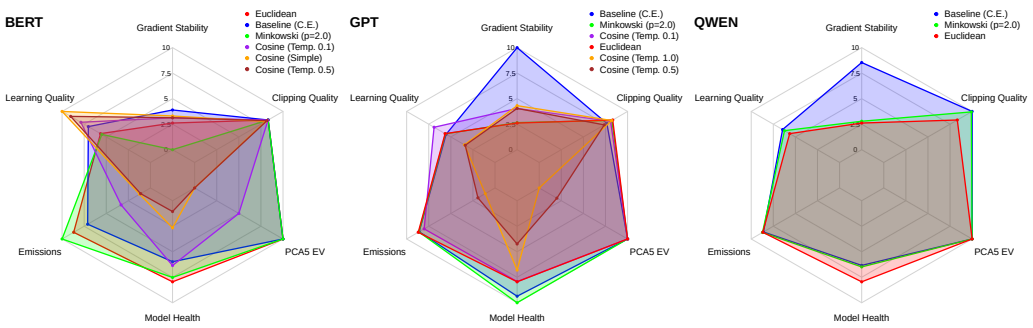

Figure 2: Language: Radar plots: 1) *Model Performance* (Model Health, Gradient Stability, Learning Quality, Clipping Quality); 2) *Interpretability* (PCA5 EV), and 3) *Sustainability* (Emissions). Plots feature Baseline (Cross-Entropy), Euclidean harmonic, and the top-performing non-Euclidean harmonic losses.

functions for classification, finding that choice of loss function impacted learning speed and testing accuracy. Gonzalez & Miikkulainen (2020) used genetic programming to develop Baikal loss, which not only led to networks achieving higher accuracy than networks trained with cross-entropy loss, but also faster training and higher performance in low-data settings. These studies demonstrate a large focus on the impact of loss function on neural networks performance. Our work builds on the discussion of the importance of loss function choice by drilling deeper on harmonic loss, examining how distance metric choice impacts the effectiveness of neural networks. Our focus is not on comparing harmonic loss with other loss functions, which was done by Baek et al. (2025), but rather to shed light on performance of a generalized harmonic loss.

Efficiency and Green AI. Green AI is an emerging initiative that calls for efficiency and energy usage to be treated as first-class evaluation criteria (Schwartz et al., 2019). Many works on green AI focus on model compression (Paula et al., 2025; Rafat et al., 2023), comparing multiple models (Verma et al., 2024) or fine-tuning strategies (Wang et al., 2023), or hyperparameter optimization for carbon emission reduction Wang et al. (2025). While prior works on new loss functions rarely report sustainability metrics, we incorporate carbon footprint analysis into our evaluation due to claims that models trained with harmonic loss are more data efficient and have less grokking (Baek et al., 2025).

Interpretability in neural networks. Neural networks are complex and not inherently interpretable, but a substantial amount of effort was done to improve interpretability (Zhang et al., 2021). The push for *interpretable by design* models argues that transparency should be built into model architectures and losses rather than added post-hoc (Rudin, 2019). Harmonic loss aligns with this vision by structurally linking model weights to class prototypes. The study by Saphra et al. (2024) discusses how internal model components reveal human-understandable circuits and features in LLMs. Techniques such as activation patching, sparse autoencoders, transcoders, and crosscoders enable structural interpretations of model behavior. Parallel to our interpretability focus, Wen et al. (2025) introduced *InterpGN*, a framework combining interpretable models with deep networks for time-series tasks, preserving understandable reasoning where possible. Though not loss-centric, it reflects the growing emphasis on transparency in deep learning research. Some work has focused on using loss functions specifically to improve model interpretability. Liu et al. (2022) combine sparse coding constraints with cross-entropy to produce concise, interpretable word-level attributions. Dong et al. (2017) introduced *interpretative loss* to improve interpretability of learned features during video captioning tasks. Within classification tasks, Zhang et al. (2018) designed a loss function to improve CNN filter interpretabiltiy. Methods such as the one proposed by Hagos et al. (2023) augment standard losses with distance-based penalties that align model attributions with user-provided annotations, strengthening interpetability.

Distance metrics in learning algorithms. Beyond supervised classification, the choice of distance measure is known to be crucial. Coil et al. (2025) compared twelve distance metrics in anomaly detection for concept drift. Their results highlighted that performance depends heavily on the chosen metric and that efficient alternatives can sometimes match the performance of more costly distances. A variety of other works have shown the importance of distance metric choice in different contexts.

Amaya-Tejera et al. (2024) used a kernel for SVMs that could support a variety of kernels, finding that distance metric choice impacted performance. Kalra et al. (2022) and Hu et al. (2016) both found that distance metric choice impacted performance of k-nearest neighbors algorithms on a variety of datasets. These result highlights the importance of systematically exploring metrics in different contexts. To our knowledge, our paper is the first to bring this comparative perspective into loss functions for deep neural networks.

6 Conclusion

This work examined *distance–based harmonic losses* as drop–in replacements for cross–entropy across image classification (MNIST, CIFAR-10, CIFAR-100) with four vision backbones (MLP, CNN, ResNet50, PVT) and LLM pretraining (GPT, BERT, Qwen), leveraging a broad family of distances (cosine, Euclidean, Bray-Curtis, Mahalanobis, Minkowski, Chebyshev, Canberra, *etc.*).

What we learned. i) **Geometry matters for optimization.** Across vision and language, *cosine (stable)* consistently delivered smoother training and the most reliable final performance; Euclidean remained a solid reference; Bray-Curtis was often competitive but architecture–sensitive; Mahalanobis showed the largest variance: occasionally strong early learning, yet less stable plateaus. Adding transformer–style PVT corroborated these vision trends. ii) **Sustainability depends on distance and architecture.** On vision tasks, several distances are *carbon–negative per step* relative to cross–entropy for CNN/ResNet50 (largest gains on deeper CNNs), mixed on MLP, and frequently *carbon–positive* on PVT. For LLM pretraining, the classifier head is lightweight, so differences accrue mainly via convergence: the cross–entropy baseline typically exhibits the largest cumulative emissions, while *cosine* and *Minkowski* are neutral–to–favorable. These results motivate reporting both per–step deltas and *energy–to–target–quality*. iii) **Interpretability can be quantified.** PCA–based probes (PC concentration and PCA@90%) provide reproducible evidence that distance–tailored heads yield more structured representations; this holds for image features and for token representations in LLMs (causal last–token and MLM masked–token states).

Language: Cosine–based harmonic losses markedly improve *gradient/learning stability* and *representation structure* (higher PCA concentration) for GPT, BERT, and Qwen, while reducing emissions relative to both cross–entropy and Euclidean heads. Mahalanobis is less reliable for pretraining due to covariance overheads and stability sensitivity.

Vision: For accuracy-focused workloads, *cosine (stable)* is preferred; Bray–Curtis is a viable secondary option; Mahalanobis should be used when its inductive bias yields clear benefits. For sustainability on CNN/ResNet50, several distances reduce per–step CO_2; on PVT and LLMs, the lightest geometries (cosine/Euclidean) should be favored, or cross-entropy should be retaind unless distance-based losses reduce steps-to-target enough to offset higher per–step cost.

Future Work. Promising directions include scalable covariance estimation for Mahalanobis–style heads, token–prototype curricula for language modeling, principled temperature control, and extending our sustainability analysis to instruction–tuning and multi–node regimes.

Reproducibility Statement. We took several steps to facilitate exact and statistical reproducibility. The main paper specifies the learning objectives, training protocol, model families, and evaluation metrics used in all studies. The *Appendix* contains: i) complete hyperparameter and backbone-specific settings; ii) dataset descriptions and end-to-end preprocessing pipelines (including splits and any filtering); iii) detailed experimental studies and analyses; iv) technical details with code snippets to integrate our non-Euclidean harmonic losses in conventional deep learning pipelines. An *anonymous* code repository (see Abstract) provides: ready-to-run scripts for data acquisition and preprocessing; configuration files for every experiment; training/evaluation entry points; instructions for reproducing results. Together, these materials are intended to enable independent researchers to audit, rerun, and extend our findings with minimal effort.

References

Nazhir Amaya-Tejera, Margarita Gamarra, Jorge I Vélez, and Eduardo Zurek. A distance-based kernel for classification via support vector machines. *Frontiers in Artificial Intelligence*, 7:1287875, 2024.

David D Baek, Ziming Liu, Riya Tyagi, and Max Tegmark. Harmonic loss trains interpretable ai models. *arXiv preprint arXiv:2502.01628*, 2025.

Martin Bengtsson et al. Compressing large language models with pca without fine-tuning, 2025. URL https://arxiv.org/abs/2508.04307.

Malik Boudiaf, Jérôme Rony, Imtiaz Masud Ziko, Eric Granger, Marco Pedersoli, and Ismail Ben Ayed. A unifying mutual information view of metric learning: Cross-entropy vs. pairwise losses. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 548–564. Springer, 2020. doi: 10.1007/978-3-030-58568-6_33.

Anne Chao, Robin L Chazdon, Robert K Colwell, and Tsung-Jen Shen. An additive decomposition formula for the Bray–Curtis dissimilarity and their ecological meaning. *Ecological Modelling*, 221(9):1275–1283, 2010.

Kwantae Cho, Jong-hyuk Roh, Youngsam Kim, and Sangrae Cho. A performance comparison of loss functions. In *2019 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 1146–1151. IEEE, 2019.

Jongmin Choi, Minsung Cho, and Seong-Whan Lee. Am-loss: Angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 298–307, 2020. doi: 10.1109/WACV45572.2020.9093485.

Collin Coil, Kamil Faber, Bartlomiej Sniezynski, and Roberto Corizzo. Distance-based change point detection for novelty detection in concept-agnostic continual anomaly detection. *Journal of Intelligent Information Systems*, pp. 1–39, 2025.

Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. ArcFace: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4690–4699, 2019.

Yinpeng Dong, Hang Su, Jun Zhu, and Bo Zhang. Improving interpretability of deep neural networks with semantic information. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4306–4314, 2017.

Nelson Elhage, Neel Nanda, et al. Toy models of superposition. 2022. Transformer Circuits Thread.

FAR AI. Uncovering latent human wellbeing in llm embeddings. https://far.ai/news/uncovering-latent-human-wellbeing-in-llm-embeddings, 2023. Shows first principal component of GPT-3 embeddings correlates with ethics/well-being labels.

Carolina Fuschi, Davide Delfino, Thomas Klammsteiner, Franco Biasioli, Gino Fernandez, Sabina Angeli, and Raffaella Causin. Microbiome data: tell me which metrics and I will tell you which communities. *Scientific Reports*, 15(1):1–14, 2025.

Arie Giloni and Manfred Padberg. The finite sample breakdown point of ℓ_1-regression. In *SIAM Journal on Optimization*, volume 14, pp. 608–620. SIAM, 2003.

María José Gómez-Silva, Arturo de la Escalera, and José María Armingol. Back-propagation of the Mahalanobis distance through a deep triplet learning model for person re-identification. *Integrated Computer-Aided Engineering*, 28(3):277–288, 2021.

Santiago Gonzalez and Risto Miikkulainen. Improved training speed, accuracy, and data utilization through loss function optimization. In *2020 IEEE congress on evolutionary computation (CEC)*, pp. 1–8. IEEE, 2020.

Elias Hagos, Dahai Lin, and Xinyi Wu. Distance-aware explanation based learning for interpretable neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 12012–12020, 2023.

Yutong He et al. Can transformers perform pca?, 2024. URL https://openreview.net/forum?id=mjDNVksC5G. OpenReview preprint.

Li-Yu Hu, Min-Wei Huang, Shih-Wen Ke, and Chih-Fong Tsai. The distance function effect on k-nearest neighbor classification for medical datasets. *SpringerPlus*, 5(1):1304, 2016.

Robert Huben, Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *International Conference on Learning Representations (ICLR), Poster*, 2024. URL https://openreview.net/forum?id=F76bwRSLeK.

Katarzyna Janocha and Wojciech Marian Czarnecki. On loss functions for deep neural networks in classification. *arXiv preprint arXiv:1702.05659*, 2017.

Ole Jorgensen. *Understanding and Controlling the Activations of Language Models*. PhD thesis, Imperial College London, 2023. URL https://ojorgensen.github.io/assets/pdfs/Imperial_Dissertation.pdf.

Vandana Kalra, Indu Kashyap, and Harmeet Kaur. Effect of distance measures on k-nearest neighbour classifier. In *2022 Second International Conference on Computer Science, Engineering and Applications (ICCSEA)*, pp. 1–7. IEEE, 2022.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, 2020.

S. L. Keeling and K. Kunisch. Robust ℓ_1 approaches to computing the geometric median and principal and independent components. *Journal of Mathematical Imaging and Vision*, 56(2):286–300, 2016.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in Neural Information Processing Systems*, 31, 2018.

Tian Liu, Han Chen, and Xiang Ren. Sparsity and interpretability: Improving attribution for neural text classifiers. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 1517–1530, 2022. doi: 10.18653/v1/2022.naacl-main.109.

David Miller, Garrett Stewart, and Fernando de la Torre. Cac: Class anchor clustering for open-set recognition. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 16326–16339, 2021.

Ibrahim Omara, Xue-juan Wu, Huan Zhang, Yingying Du, and Wangmeng Zuo. A novel approach for ear recognition: learning Mahalanobis distance features from deep CNNs. *Machine Vision and Applications*, 32(2):1–17, 2021.

Tianyu Pang, Chao Du, Yinpeng Dong, and Jun Zhu. Max-Mahalanobis linear discriminant analysis networks. In *International Conference on Machine Learning*, pp. 4016–4025. PMLR, 2018.

Eileen Paula, Jayesh Soni, Himanshu Upadhyay, and Leonel Lagos. Comparative analysis of model compression techniques for achieving carbon efficient ai. *Scientific Reports*, 15(1):23461, 2025.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *CoRR*, abs/2201.02177, 2022.

Kazi Rafat, Sadia Islam, Abdullah Al Mahfug, Md Ismail Hossain, Fuad Rahman, Sifat Momen, Shafin Rahman, and Nabeel Mohammed. Mitigating carbon footprint for knowledge distillation based deep learning model compression. *Plos one*, 18(5):e0285668, 2023.

Kanchana Ranasinghe, Mehrtash Harandi, and Fatih Porikli. Orthogonal projection loss for learning discriminative features in face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13369–13378, 2021. doi: 10.1109/CVPR46437.2021.01318.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019. doi: 10.1038/s42256-019-0048-x.

Naomi Saphra, Tim Lieberum, Arthur Conmy, Abhinav Sharma, Yushi Wu, Nelson Elhage, Catherine Olsson, Nicholas Joseph, Ethan Perez, Lawrence Chan, et al. Mechanistic interpretability: Methods and applications. *arXiv preprint arXiv:2408.13296*, 2024.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green ai. In *Communications of the ACM*, volume 63, pp. 54–63, 2019. doi: 10.1145/3381831.

Ling Song, Peter Langfelder, and Steve Horvath. Systematic comparisons for composition profiles, taxonomic levels, and machine learning methods for microbiome-based disease prediction. *Frontiers in Molecular Biosciences*, 7:618573, 2020.

Yantao Sun, Yuxin Chen, Xiaogang Wang, and Xiaoou Tang. Learning discriminative CNN features and similarity metrics for image retrieval. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 1013–1018. IEEE, 2016.

Arthur Templeton et al. Sparse autoencoders find highly interpretable directions in language models. https://www.alignmentforum.org/posts/Qryk6FqjtZk9FHHJR/sparse-autoencoders-find-highly-interpretable-directions-in, 2023.

Alex Turntrout. Steering GPT-2-XL by adding an activation vector. https://turntrout.com/gpt2-steering-vectors, 2023.

Anil Verma, Sumit Kumar Singh, Rupesh Kumar Sah, Rajiv Misra, and TN Singh. Performance comparison of deep learning models for co2 prediction: Analyzing carbon footprint with advanced trackers. In *2024 IEEE International Conference on Big Data (BigData)*, pp. 4429–4437. IEEE, 2024.

Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. CosFace: Large margin cosine loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5265–5274, 2018.

Irene Wang, Newsha Ardalani, Mostafa Elhoushi, Daniel Jiang, Samuel Hsia, Ekin Sumbul, Divya Mahajan, Carole-Jean Wu, and Bilge Acun. Carbon aware transformers through joint model-hardware optimization. *arXiv preprint arXiv:2505.01386*, 2025.

Xiaorong Wang, Clara Na, Emma Strubell, Sorelle Friedler, and Sasha Luccioni. Energy and carbon considerations of fine-tuning bert. *arXiv preprint arXiv:2311.10267*, 2023.

Haoran Wen, Zhi Li, Yue Wu, and Tianqi Chen. Interpgn: Combining interpretable models with deep networks for time-series. In *International Conference on Learning Representations (ICLR)*, 2025.

Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 499–515. Springer, 2016. doi: 10.1007/978-3-319-46478-7_31.

Jinfeng Ye, Tao Li, Tao Xiong, and Ravi Janardan. A pure L_1-norm principal component analysis. *Computational Statistics & Data Analysis*, 56(12):4474–4486, 2012.

Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Interpretable convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8827–8836, 2018.

Yu Zhang, Peter Tiňo, Aleš Leonardis, and Ke Tang. A survey on neural network interpretability. *IEEE transactions on emerging topics in computational intelligence*, 5(5):726–742, 2021.

Yuxin Zhou et al. Exploring concept depth: How large language models acquire concepts across layers, 2024. URL https://arxiv.org/abs/2404.07066.

APPENDIX

A THEORETICAL PROPERTIES OF DISTANCE-BASED PROBABILISTIC LAYERS

Setup. Let $\{(x_i, y_i)\}_{i=1}^n$ be the training set with $y_i \in \{1, ..., K\}$. Each class has a prototype $w_k \in \mathbb{R}^d$ and a nonnegative distance $d(x, w) \geq 0$. Given a decreasing link $\kappa : \mathbb{R}_+ \to \mathbb{R}_+$ we define

$$p_k(x_i) \;=\; \frac{\kappa\big(d(x_i, w_k)\big)}{\sum_{j=1}^K \kappa\big(d(x, w_j)\big)}, \qquad \mathcal{L}(\{w_k\}) \;=\; -\sum_{i=1}^n \log p_{y_i}(x_i).$$

harmonic $\kappa(r) = r^{-\omega}$ with $\omega > 0$; and Typical distances include Euclidean/Mahalanobis, ℓ_p, Bregman divergences, and cosine/angle on the sphere.

A.1 SCALE INVARIANCE AND FINITE MINIMIZERS

We begin by generalizing the finite-minimizer result of the harmonic loss (cf. Thm. 1, Sec. G in Baek et al. (2025)).

Definition 1 (Metric separability and homogeneity)**.** A dataset is *metric-separable* if for each i there exists $\{w_k\}$ s.t. $d(x_i, w_{y_i}) < \min_{j \neq y_i} d(x_i, w_j)$. A distance d is *1-homogeneous* if $d(cx, cw) = |c|\, d(x, w)$ for all $c > 0$.

Theorem 1 (Finite minimizer and scale invariance for harmonic link)**.** *Assume d is 1-homogeneous and the training set is metric-separable. For $\kappa(r) = r^{-\omega}$, the empirical loss $\mathcal{L}$ is invariant to the joint rescaling $(x, w) \mapsto (cx, cw)$ and attains a global minimum at* finite $\{w_k\}$. *In particular, increasing $\|w_k\|$ further does not reduce $\mathcal{L}$.*

Proof. Following the proof of Sec. G Thm. 1 in Baek et al. (2025), the probabilities remain unchanged under uniform scaling for any 1-homogeneous distance d. For the probabilities, if we replace x_i by $c_x i$ and w_j with cw_j, then $d(cx_i, cw_j) = c\, d(x_i, w_j)$, so the scaling factors cancel when using a harmonic link κ. Therefore, once the correct classification is achieved, no further reduction in loss is obtained by increasing $\|w_k\|$ and the loss achieves a global minimum at a finite $\{w_k\}$.

□

A.2 MARGIN-STYLE GENERALIZATION (PAC-BAYES VIEW)

Sec. G gives a PAC-Bayes margin bound that is finite because the harmonic solution has finite norm (Thm. 2) in Baek et al. (2025).

Definition 2 (Distance margin)**.** Given prototypes $W = \{w_k\}$, define $\gamma(W) \;=\; \min_i \big[d(x_i, w_{y_i}) - \min_{j \neq y_i} d(x_i, w_j)\big]$.

Theorem 2 (Generalization with metric margin)**.** *Assume all x_i lie in a ball of radius R (in the native norm of d or its inducing space). Let $\|W\|_\star$ denote a capacity measure compatible with d. With probability at least $1 - \delta$, the generalization error of the classifier satisfies*

$$\Pr_{(x,y)}\big[h_W(x) \neq y\big] \;=\; \mathcal{O}\left(\frac{R\,\|W\|_\star}{\gamma(W)\sqrt{n}} + \sqrt{\frac{\log(1/\delta)}{n}}\right),$$

where $h_W(x) = \arg\max_k p_k(x)$ denotes the predicted class and n is the number of training samples. For the harmonic link, $\|W\|_\star$ is finite by Thm. 1, yielding a finite bound (cf. Sec. G Thm. 2) in Baek et al. (2025).

Proof. Mirroring the proof for Sec. G Thm. 2 in Baek et al. (2025), applying the standard PAC-Bayes margin bounds, one obtains that with at least probability $1 - \delta$,

$$\Pr_{(x,y)}\left[h_W(x) \neq y\right] \;=\; \mathcal{O}\left(\frac{R\,\|W\|_\star}{\gamma(W)\sqrt{n}} + \sqrt{\frac{\log(1/\delta)}{n}}\right).$$

Since $\|W\|_\star$ is finite by Thm. 1, the bound is finite. □

B Integration into Deep Learning Pipelines

The `DistLayer` abstraction highlights that distance-based harmonic loss functions are highly modular and can be seamlessly integrated into existing deep learning pipelines. The `forward` method requires only three operations: (i) computing pairwise distances between sample embeddings and class prototype weights, (ii) clamping values for numerical stability, and (iii) applying a softmin via `log_softmax` to obtain normalized class probabilities. This makes the substitution of Euclidean distance with alternative metrics essentially a one-line change in the distance registry, with no modifications required in the broader training loop.

Several design choices make the implementation robust. First, all distance functions are implemented in a vectorized form, ensuring GPU efficiency and avoiding explicit loops. Second, numerical safeguards (e.g., ε-offsets, clamping before roots and divisions, regularization of covariance matrices) prevent instability across diverse datasets and architectures. Third, the registry-based design allows new distance functions to be added without disrupting the existing workflow, reinforcing the flexibility of harmonic loss as a general framework.

From a methodological perspective, this implementation highlights one of the key contributions of this work: *the ease of replacing cross-entropy with distance-based harmonic loss*. Unlike cross-entropy, which relies on unbounded logit growth, the harmonic formulation treats classification as a problem of minimizing distances to interpretable class prototypes. The plug-and-play nature of the `DistLayer` demonstrates that alternative geometries (e.g., cosine, Mahalanobis, Bray–Curtis) can be explored at negligible engineering cost, paving the way for systematic evaluation of accuracy, sustainability, and interpretability across diverse tasks.

```python
class DistLayer(nn.Module):
    """Final classification head using harmonic loss: logits = -distance."""
    def __init__(self, in_features, n_classes, dist_name="euclidean", **dist_kwargs):
        super().__init__()
        self.W = nn.Parameter(torch.empty(n_classes, in_features))
        nn.init.kaiming_uniform_(self.W, a=5**0.5)
        self.dist_name  = dist_name
        self.dist_fn    = DIST_REGISTRY[dist_name]
        self.dist_kwargs = dist_kwargs    # e.g., p for minkowski, cov_inv for mahalanobis

    def forward(self, h):
        """
        h: (B, D) features from backbone.
        Returns log-probs for harmonic loss:  log_softmax(-distance).
        """
        d = self.dist_fn(h, self.W, **self.dist_kwargs)    # (B, C)
        d = torch.clamp(d, min=1e-6, max=1e6)             # general safety clamp
        logits = -d                                        # softmin over distances
        return F.log_softmax(logits, dim=-1)
```

```python
import torch
import torch.nn as nn
import torch.nn.functional as F

def _pairwise(fn):
    """Lift a vector distance fn(h, w) -> scalar into a batched pairwise form."""
    def lifted(h, W):
        # h: (B, D), W: (C, D) -> (B, C)
        h_exp = h.unsqueeze(1)      # (B, 1, D)
        W_exp = W.unsqueeze(0)      # (1, C, D)
        return fn(h_exp, W_exp)
    return lifted

def euclidean(h, W, eps=1e-4):
    diff = h - W
    return torch.sqrt(torch.clamp((diff * diff).sum(-1) + eps, min=eps))
```

```python
def manhattan(h, W, eps=1e-4):
    return (h - W).abs().sum(-1) + eps

def cosine(h, W, eps=1e-6, stable=True):
    if stable:
        h_n = F.normalize(h, p=2, dim=-1)
        W_n = F.normalize(W, p=2, dim=-1)
        cos = (h_n * W_n).sum(-1)
    else:
        num = (h * W).sum(-1)
        den = torch.clamp(h.norm(dim=-1) * W.norm(dim=-1) + eps,
        ↪  min=eps)
        cos = num / den
    return 1.0 - cos + eps

def minkowski(h, W, p=1.5, eps=1e-6):
    diff = torch.clamp((h - W).abs() + eps, min=eps)
    dist_p = torch.clamp(diff.pow(p).sum(-1) + eps, min=eps)
    return dist_p.pow(1.0 / p)

def chebyshev(h, W, eps=1e-6, smooth=False, alpha=10.0):
    diff = (h - W).abs()
    if smooth:
        # soft-max norm
        return torch.logsumexp(alpha * diff, dim=-1) / alpha + eps
    return diff.max(dim=-1).values + eps

def canberra(h, W, eps=1e-4, variant="standard", min_denom=1e-3,
             weight_power=1.0, normalize_weights=True):
    num = (h - W).abs()
    den = h.abs() + W.abs() + eps
    if variant == "robust":
        den = torch.clamp(den, min=min_denom)
    if variant == "weighted":
        w = (den.pow(weight_power))
        if normalize_weights:
            w = w / (w.sum(-1, keepdim=True) + eps)
        return (w * (num / den)).sum(-1) + eps
    return (num / den).sum(-1) + eps

def bray_curtis(h, W, eps=1e-3, variant="standard", min_sum=1e-3):
    num = (h - W).abs().sum(-1)
    if variant == "abs":
        den = (h.abs() + W.abs()).sum(-1)
    else:  # standard/normalized
        den = (h + W).sum(-1).abs()
    den = torch.clamp(den + eps, min=10 * eps, max=1e6)
    return torch.clamp(num / den + eps, min=eps, max=1e6)
```

```python
def mahalanobis(h, W, eps=1e-6, cov_inv=None, reg_lambda=1e-2):
    # h: (B, 1, D), W: (1, C, D) expected (use _pairwise wrapper)
    diff = h - W  # (B, C, D)
    try:
        if cov_inv is None:
            # Identity with mild regularization
            return torch.sqrt(torch.clamp((diff * diff).sum(-1) + eps,
            ↪  min=eps))
        cov_inv_reg = cov_inv + torch.eye(cov_inv.size(0),
        ↪  device=cov_inv.device) * reg_lambda
        diff_M = torch.einsum('bcd,dd->bcd', diff, cov_inv_reg)
        dist2  = (diff_M * diff).sum(-1)
        return torch.sqrt(torch.clamp(dist2 + eps, min=eps))
    except Exception:
        # Safe fallback: Euclidean
        return torch.sqrt(torch.clamp((diff * diff).sum(-1) + eps,
        ↪  min=eps))
```

```python
# Lift to pairwise batch form
EUCLIDEAN     = _pairwise(euclidean)
MANHATTAN     = _pairwise(manhattan)
COSINE        = _pairwise(cosine)
MINKOWSKI     = _pairwise(minkowski)
CHEBYSHEV     = _pairwise(chebyshev)
CANBERRA      = _pairwise(canberra)
BRAY_CURTIS   = _pairwise(bray_curtis)
MAHALANOBIS   = _pairwise(mahalanobis)

DIST_REGISTRY = {
    "euclidean":    lambda h, W, **kw: EUCLIDEAN(h, W, **kw),
    "manhattan":    lambda h, W, **kw: MANHATTAN(h, W, **kw),
    "cosine":       lambda h, W, **kw: COSINE(h, W, **kw),
    "minkowski":    lambda h, W, **kw: MINKOWSKI(h, W, **kw),
    "chebyshev":    lambda h, W, **kw: CHEBYSHEV(h, W, **kw),
    "canberra":     lambda h, W, **kw: CANBERRA(h, W, **kw),
    "bray-curtis":  lambda h, W, **kw: BRAY_CURTIS(h, W, **kw),
    "mahalanobis":  lambda h, W, **kw: MAHALANOBIS(h, W, **kw),
}
```

C Model Architectures

C.1 Vision

We detail the architectures of the vision models used in our experiments – including a simple MLP, a small CNN, ResNet-50, and PVTv2-B0 – specifying their layers and neuron counts for reproducibility. All models were implemented in PyTorch, and for distance-based variants, the final fully-connected layer is replaced by a specialized distance layer as noted below.

MLP: **Input Layer:** Accepts the flattened image input (e.g., $28 \times 28 = 784$ features for MNIST, $32 \times 32 \times 3 = 3072$ for CIFAR). **Hidden Layer 1:** Fully-connected layer with 512 neurons, followed by ReLU. **Hidden Layer 2:** Fully-connected layer with 256 neurons, followed by ReLU. **Output Layer:** Linear mapping from 256 units to the number of classes (10 for MNIST/CIFAR-10, 100 for CIFAR-100). In `_DIST` variants, this layer is replaced with a distance-based classification head (e.g. Euclidean, cosine) that computes distances between the embedding and class prototypes, outputting *negative distances* as logits.

CNN: **Conv Layer 1:** 2D convolution, 32 filters, kernel size 3×3, padding 1, followed by ReLU, then 2×2 max pooling. **Conv Layer 2:** 2D convolution, 64 filters, kernel size 3×3, padding 1,

followed by ReLU, then 2×2 max pooling. **Fully Connected Layer:** Flattened output fed into a 128-unit linear layer with ReLU. **Output Layer:** Linear layer mapping the 128-D representation to the number of classes. In `_DIST` variants, this is replaced by a distance metric layer.

ResNet-50: **Stem:** Standard 7×7 convolution with 64 filters and stride 2, batch norm, ReLU, then 3×3 max pooling. For CIFAR/MNIST, we use a 3×3 conv with stride 1 and remove max pooling. **Stage 1:** 3 bottleneck blocks, output 256 channels. **Stage 2:** 4 bottleneck blocks, output 512 channels. **Stage 3:** 6 bottleneck blocks, output 1024 channels. **Stage 4:** 3 bottleneck blocks, output 2048 channels. **Global Pooling and Output:** Global average pooling yields a 2048-D vector. In the baseline, a linear FC layer maps to logits. In `_DIST` variants, the FC is replaced by a distance layer (e.g. cosine similarity) that outputs similarity-based logits.

Pyramid Vision Transformer (PVTv2-B0): **Stage 1:** Overlapping patch embedding with a 7×7 conv (stride 4), output 32 channels, followed by 2 Transformer encoder layers (1 attention head). **Stage 2:** 3×3 conv (stride 2), output 64 channels, followed by 2 encoder layers (2 heads). **Stage 3:** 3×3 conv (stride 2), output 160 channels, followed by 2 encoder layers (5 heads). **Stage 4:** 3×3 conv (stride 2), output 256 channels, followed by 2 encoder layers (8 heads). **Global Pooling and Output:** Global average pooling yields a 256-D vector. A linear classifier maps to the number of classes in the baseline, while in `_DIST` variants this is replaced with a distance layer producing log-similarity or negative distance scores.

Preprocessing Pipelines: **MNIST:** For MLP/CNN, grayscale input normalized to mean 0.5, std 0.5. For ResNet/VGG, normalization uses dataset statistics (mean 0.1307, std 0.3081). For ViT/PVT, grayscale converted to 3 channels, resized to 224 (ViT) or 32 (PVT), normalized to mean/std 0.5. **CIFAR-10:** Normalization with mean (0.4914, 0.4822, 0.4465) and std (0.2023, 0.1994, 0.2010). ResNet/VGG use data augmentation (random flips, crops, small rotations). ViT inputs are resized to 224×224. **CIFAR-100:** Normalization with mean (0.5071, 0.4867, 0.4408) and std (0.2675, 0.2565, 0.2761). Stronger augmentation (random flips, crops, rotations, color jitter). ViT inputs resized to 224×224. PVT models use 32×32 resized inputs with normalization.

C.2 LLMs

This section documents the **LLM** configurations used in our experiments for reproducibility. We report data preprocessing, architectural details for **GPT**, **BERT**, and **Qwen2**-style models, how **distance-based heads** are integrated in place of the standard linear classifier, and the training/evaluation/emissions-logging pipeline. All models are implemented in PyTorch and trained with mixed precision when available.

Data and Preprocessing Corpus and Storage. We pre-process a text corpus into contiguous token ID arrays and store them as memory-mapped files:

- `train.bin` and `val.bin`: `np.memmap` arrays of type `uint16` containing token IDs.
- `meta.pkl`: contains metadata including `vocab_size` (used to configure model embeddings).

Let V denote the discovered vocabulary size from `meta.pkl` (fallback $V{=}50304$ if not found).

Batching. For a given `block_size` L, batches are sampled by picking random starting indices and slicing L tokens:

`X = data[i : i+L], Y = data[i+1 : i+1+L]` (causal LM)

All batching is performed on-device with pinned memory. We denote `batch_size` by B.

Masking for MLM (BERT). For BERT runs, we construct masked language modeling (MLM) batches with the standard 15% corruption:

- Select $\approx 15\%$ token positions per sequence to form mask indices $\mathcal{M}$.
- For each $i \in \mathcal{M}$: with 80% probability replace x_i with `[MASK]` (id ≤ 103 or capped by $V-1$), with 10% replace by a random token in $[0, V)$, with 10% keep x_i unchanged.
- Labels use the original token at masked positions and -100 (ignore index) elsewhere.

This yields `input_ids`, `attention_mask` (all-ones here), and `labels` containing ground-truth only at masked positions.

Architectures Across models below, the principal hyperparameters are:

layers (n_ℓ), heads (n_h), embedding dim (d), context length (L=`block_size`), vocab size (V).

Unless otherwise specified, positional encodings follow each model's default (e.g., learned or rotary).

GPT2 (CAUSAL LM)

Backbone. A standard decoder-only Transformer with n_ℓ blocks. Each block has:

- Multi-Head Causal Self-Attention with n_h heads, hidden size d, and causal mask.
- Position-wise MLP of width typically $\approx 4d$ with nonlinearity (e.g., GELU).
- Pre/post LayerNorm and residual connections as in GPT-style decoders.

Token Embeddings. Learnable token and (implicit) position embeddings of sizes $V \times d$ and $L \times d$ (or rotary embeddings if enabled). **Projection Head (baseline).** A linear layer $W_{\text{lm}} \in \mathbb{R}^{d \times V}$ producing logits over the vocabulary at each position. **Distance Head (_DIST).** The linear projection is replaced by a *distance-based layer* that treats the vocabulary columns as *prototypes* $\{w_v \in \mathbb{R}^d\}_{v=1}^{V}$. Given a hidden state $h_t \in \mathbb{R}^d$, the head returns per-token logits $z_{t,v} = -D(h_t, w_v; \Theta)$ (or $\log S(h_t, w_v)$ for similarity-type layers), where $D(\cdot, \cdot; \Theta)$ is one of the distances defined in the main text (Euclidean, cosine, Manhattan, Minkowski, Canberra, Bray–Curtis, Chebyshev, Mahalanobis, Hamming). This integrates seamlessly with the causal LM objective (next-token prediction via softmax over V).

BERT (MASKED LM)

Backbone. An encoder-only Transformer with n_ℓ layers, each with:

- Multi-Head Self-Attention (bidirectional) with n_h heads.
- Position-wise MLP, LayerNorm, residual connections.

Embeddings. Token embeddings $V \times d$, segment/type embeddings (size 2), and positional embeddings of length L. **Head (baseline).** The standard MLM classifier projects $d \to V$ (optionally via an intermediate nonlinearity tied to the embedding matrix). **Distance Head (_DIST).** We replace the MLM classifier with the same prototype-based distance layer used for GPT, but applied *only at masked positions*. For each masked token representation h_i, logits are $z_{i,v} = -D(h_i, w_v; \Theta)$ (or log-similarity), and cross-entropy is computed against the ground-truth token at i.

QWEN2-STYLE DECODER (CAUSAL LM)

Backbone. A decoder-only Transformer similar to GPT, with model-specific details:

- Rotary Position Embeddings (RoPE) with θ (e.g., $\theta = 10^6$).
- RMSNorm with ϵ (e.g., 10^{-6}) in place of LayerNorm.
- Grouped key/value heads: `num_key_value_heads` may be $< n_h$.
- Intermediate MLP width (`intermediate_size`) configurable.

Vocabulary. By default, we use Qwen's native vocabulary (`vocab_size`=151,936); alternatively, one can adapt to the dataset vocab. **Head (baseline vs. _DIST).** As with GPT, the final projection is either a linear layer to V or a distance-based head over V prototype vectors.

DISTANCE-BASED OUTPUT LAYER

For all three families (GPT, BERT/MLM, Qwen2), the baseline $d \to V$ classifier is replaced in `_DIST` runs by a distance head:

$$z_v(h) \;=\; \begin{cases} -\,\|h - w_v\|_2 & \text{(Euclidean)} \\ -\,\|h - w_v\|_1 & \text{(Manhattan)} \\ -\,\|h - w_v\|_p & \text{(Minkowski, } p \text{ specified)} \\ -\left(1 - \frac{h^\top w_v}{\|h\|_2 \|w_v\|_2}\right) & \text{(Cosine)} \\ -\,D_{\text{Canberra}}(h, w_v) \;\text{ or }\; -\,D_{\text{Bray–Curtis}}(h, w_v) & \text{(variants as defined)} \\ -\,\|h - w_v\|_\infty & \text{(Chebyshev)} \\ -\sqrt{(h - w_v)^\top \Sigma^{-1} (h - w_v)} & \text{(Mahalanobis, variants)} \\ -\,D_{\text{Hamming}}(h, w_v) & \text{(soft/gumbel/hard)} \end{cases}$$

where w_v are learned prototype vectors (analogous to classifier weights). We adopt the numerically robust implementations given in the main text (e.g., small ε, clamping, optional normalization of h and/or w_v where appropriate). For cosine, we may output log-similarities for stability. Loss is standard cross-entropy over the V logits per position (causal) or per masked position (MLM).

TRAINING SETUP AND OPTIMIZATION

Device and Precision. We use `bfloat16/float16/float32` (configurable) with automatic mixed precision:

```
torch.autocast(device_type='cuda', dtype=ptdtype).
```

Training can run in single-GPU or **DDP** (`torch.distributed`) multi-GPU mode. In DDP, `LOCAL_RANK` selects the device, and gradients are synchronized across ranks.

Initialization and Checkpointing. Models are initialized *from scratch* using the specified architecture config (layers, heads, width, L, V). For GPT-only runs we optionally support `init_from='gpt2*'`, and for BERT we support `init_from='bert*'` (when provided), with appropriate overrides. Checkpoints store model/optimizer state, `iter_num`, `best_val_loss`, and the configuration.

Optimizer and LR Schedule. We use the model's `configure_optimizers` helper to instantiate an Adam/AdamW-style optimizer with weight decay and (β_1, β_2). Learning rate follows cosine decay with warmup:

$$\text{lr}(t) = \begin{cases} \text{lr}_{\max} \cdot t/\text{warmup} & t < \text{warmup}, \\ \text{lr}_{\min} + \frac{1}{2}\left(1 + \cos\frac{\pi(t - \text{warmup})}{T - \text{warmup}}\right)\left(\text{lr}_{\max} - \text{lr}_{\min}\right) & t \le T, \end{cases}$$

where T is `lr_decay_iters`. We apply gradient accumulation (`gradient_accumulation_steps`), optional gradient clipping (`grad_clip`), and AMP scaling (`GradScaler`).

Objectives.

- **GPT/Qwen2 (causal LM):** next-token cross-entropy over V at each position.
- **BERT (MLM):** cross-entropy computed only at masked positions; non-masked labels set to -100 (ignored).

Accuracy reporting: we compute token-level accuracy for monitoring (on next-token for causal LM, on masked tokens for MLM).

Evaluation and Early Signals. At fixed `eval_interval`, we run `estimate_loss()` over `eval_iters` batches on train/val splits (model in `eval()`), then resume training. Best validation loss checkpoints are saved; optional compile (`torch.compile`) can be enabled.

SUSTAINABILITY TRACKING

We integrate *CodeCarbon* to measure energy and emissions. At each evaluation interval:

1. Stop the tracker and record interval-level metrics: emissions (kg CO_2), duration, estimated CPU/GPU/RAM power and energy.
2. Log cumulative emissions and training metrics (loss, lr) to W&B (if enabled).
3. Restart the tracker for the next interval to avoid long-running file locks and to attribute emissions to training phases cleanly.

At the end of training, we stop the tracker one final time and persist all accumulated records to a CSV (`emissions_*.csv`) alongside model checkpoints.

Key Configuration Knobs (Reproducibility) The following knobs are saved in run configs/checkpoints and should be reported alongside results:

$$(n_\ell,\, n_h,\, d,\, L,\, V),\quad \text{distance head type and parameters } (\Theta),\quad \text{batch size } B,\ \text{precision},$$
$$\text{optimizer \& betas},\ \text{lr schedule } (\text{warmup},\, T,\, \text{lr}_{\max},\, \text{lr}_{\min}),\ \text{grad accumulation},$$
$$\text{grad clip},\ \text{DDP world size}.$$

When using `_DIST` variants, we additionally report which distance (Euclidean, cosine, Manhattan, Minkowski(p), Canberra, Bray–Curtis, Chebyshev, Mahalanobis, Hamming), any normalization/scaling flags, and regularization choices (e.g., Mahalanobis covariance learning/regularization).

Summary. In all models, the sole architectural change introduced by harmonic loss is confined to the **output head**: a drop-in replacement of the linear classifier with a **distance-based prototype head** over the vocabulary. This isolates the effect of the loss geometry while keeping the Transformer backbone (and training recipe) unchanged, enabling controlled comparisons across distances in terms of *accuracy*, *interpretability* (e.g., PCA-based analyses), and *sustainability* (emissions and runtime).

D HYPERPARAMETER CONFIGURATIONS

The hyperparameter settings in Tables 1–8 were chosen to balance *comparability*, *training stability*, and *sustainability*. Below we highlight several important considerations.

D.1 LANGUAGE MODELS (OPENWEBTEXT)

Table 1 specifies the core training parameters for GPT, BERT, and Qwen on OpenWebText. The main goal was to maintain a fair comparison across models of varying scale by using effective batch sizes of similar order (76–128). This ensures that any differences observed in performance or emissions are attributable to the *loss formulation*, not simply to batch scaling. The use of AdamW with default β values (0.9, 0.999) follows current best practices for stability.

Table 2 details architecture-specific modifications. BERT includes type embeddings and a masked language modeling (MLM) setup, while GPT and Qwen use causal language modeling (CLM). Qwen, being substantially larger, incorporates more advanced design elements such as grouped query attention (GQA) and rotary position embeddings (RoPE). Table 3 summarizes these differences: GPT and Qwen follow causal objectives, while BERT relies on bidirectional context, which may affect the degree to which distance-based losses interact with their representations.

D.2 VISION MODELS (MNIST, CIFAR-10, CIFAR-100)

Tables 4–7 provide the vision settings across MNIST, CIFAR-10, and CIFAR-100. As shown in Table 4, optimizer and learning-rate schedules are backbone-specific: Adam for MLPs and CNNs, AdamW for transformers (PVT), and SGD with momentum for ResNet50. This reflects both convention and empirical stability in preliminary experiments. Longer schedules are used for CIFAR-100 due to its greater difficulty, with patience for early stopping (Table 6) scaled accordingly.

Table 1: Core configuration for GPT, BERT, and Qwen on OpenWebText.

Configuration	GPT	BERT	Qwen
n_{layer}	12	12	24
n_{head}	12	12	14
n_{embd}	768	768	896
Vocab size	50304	50304	151936
Dropout	0.1	0.1	0.0
Bias	True	True	True
Batch size	16	38	6
Grad. accum. steps	8	2	10
Effective batch size	128	76	60
Learning rate	2e-4	1e-4	1e-4
Warmup iters	500	1000	1000
Weight decay	0.01	0.01	0.01
Grad clip	1.0	1.0	1.0
Min LR	2e-6	1e-6	1e-6
Decay LR	True	True	True
LR decay iters	10000	10000	10000
Max iters	10000	10000	10000
Dataset	OpenWebText	OpenWebText	OpenWebText
dtype	bfloat16	bfloat16	bfloat16
Optimizer	AdamW	AdamW	AdamW
β_1, β_2	0.9, 0.999	0.9, 0.999	0.9, 0.999
Eval interval	1000	1000	1000
Eval iters	100	100	100
Log interval	50	50	50
Scale attn by inverse layer idx	False	False	False

Table 2: Architecture-specific settings for GPT, BERT, and Qwen.

Configuration	GPT	BERT	Qwen
Block size / Seq length	1024	512	1024
Type vocab size	–	2	–
Pad token id	–	0	–
MLM probability	–	0.15	–
Mask token id	–	103	–
Intermediate size	–	–	4864
# key–value heads	–	–	2
RMSNorm ϵ	–	–	1e-6
RoPE θ	–	–	1,000,000.0

Batch size selection (Table 5) reflects hardware utilization on H100 GPUs. Notably, lightweight backbones (e.g., CNNs) leverage very large batches (up to 8192 for MNIST), while transformer-based models (ViT, PVT) are limited to much smaller batches (128–256) to fit memory constraints. These design choices affect emissions profiles: large-batch training can reduce wall-clock time but at the cost of GPU memory overhead.

Learning-rate schedulers differ across models. For example, PVT employs cosine annealing, which smooths convergence and interacts well with distance-based loss formulations. ResNet50 relies on multi-step decay, ensuring stability across the long 200-epoch training horizon on CIFAR-100.

Distance Layer Parameters. Table 8 summarizes the shared hyperparameters across all distance functions. The exponent n is fixed to 1.0 and $\varepsilon = 10^{-4}$ provides numerical stability. Importantly, distances are not scaled post hoc, ensuring that differences in results are directly attributable to the geometric properties of the chosen distance (Euclidean, Manhattan, Mahalanobis, etc.), rather than to auxiliary tuning.

Table 3: Key differences summary (task and position encoding).

Aspect	GPT	BERT	Qwen
Model size (approx.)	∼124M	∼110M	∼494M
Attention	Causal	Bidirectional	Causal (GQA)
Training task	CLM	MLM	CLM
Position encoding	Learned	Learned	RoPE

CLM = Causal Language Modeling; MLM = Masked Language Modeling; GQA = Grouped Query Attention.

Table 4: Core training configuration by backbone and dataset.

Configuration	MLP	CNN	PVT	ResNet50
LR (MNIST)	3e-4	3e-4	1e-4	0.1
LR (CIFAR-10)	3e-4	3e-4	1e-4	0.1
LR (CIFAR-100)	3e-4	3e-4	5e-5	0.1
Epochs (MNIST)	40	40	80	100
Epochs (CIFAR-10)	40	40	80	100
Epochs (CIFAR-100)	150	150	150	200
Optimizer	Adam	Adam	AdamW	SGD
Weight decay	0	0	0.05	1e-4
Momentum	–	–	–	0.9

D.3 Discussion

Language models. GPT and BERT use comparable depth/width with learned positional encodings, while Qwen is larger, adopts RoPE, and GQA. Effective batch sizes (via gradient accumulation) normalize throughput across models for fair comparison on OpenWebText.

Vision models. Optimizer and scheduler choices follow common practice: Adam/AdamW for MLP/CNN/PVT, SGD with momentum for ResNet50; deeper/longer CIFAR-100 runs employ stepped or cosine schedules. Early-stopping patience scales with dataset difficulty.

DistLayer defaults. A unified setting (n=1.0, ε=10^{-4}, no scaling) ensures distance variants differ only in geometry, not in auxiliary hyperparameters. These settings match the configuration used in our main experiments and figures.

Table 5: Batch size configuration on H100 GPU.

Model	MNIST	CIFAR-10	CIFAR-100
MLP	2048	1024	1024
CNN	8192	4096	512
PVT	256	256	256
ViT	256	128	128
VGG16	4096	2048	512
ResNet50	512	512	256

Table 6: Learning-rate schedulers by backbone and dataset.

Model	MNIST	CIFAR-10	CIFAR-100
MLP	None	None	StepLR (step=50, γ=0.5)
CNN	None	None	StepLR (step=50, γ=0.5)
VGG16	StepLR (30, 0.1)	StepLR (30, 0.1)	MultiStepLR ([75,125], 0.1)
ResNet50	StepLR (30, 0.1)	StepLR (30, 0.1)	MultiStepLR ([60,100,140], 0.2)
ViT	None	None	CosineAnnealingLR ($T_{\max}$=150)
PVT	None	None	CosineAnnealingLR ($T_{\max}$=150)

E Additional Results

E.1 Vision: Aggregated Emissions

Figure 3 reports cumulative emission *differences* (gCO_2eq) by distance across all 228 vision experiments (Total Baseline = 181.2 gCO_2eq). **Lower-than-baseline emissions**: *Mahalanobis (Standard)* shows the largest *positive* delta, indicating consistently lower emissions; *Bray–Curtis (Standard)* and *Cosine (Unstable)* also sit on the positive side, with *Canberra (Standard)* and *Cosine (Stable)* slightly above zero. Euclidean and Manhattan are close to baseline. Other distances are characterized by higher emissions, as shown by the red cluster. Results reinforce that non-Euclidean harmonic losses can be more sustainable than their Euclidean counterpart, and that the choice of distance materially affects the carbon footprint of model training.

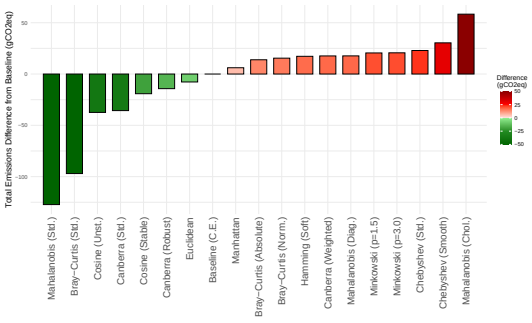

Figure 3: Vision: Aggregated Emissions.

E.2 Vision: Top-Performers across Configurations

The analysis of top-performers across the 12 configurations (4 models, 3 datasets) highlights consistent patterns in how non-Euclidean harmonic losses compare with Euclidean harmonic loss and cross-entropy.

Performance (Accuracy and F1). Bray–Curtis (normalized) leads in performance with 3 wins in *Acc* and 3 wins in *F1*, coupled with the strongest breadth of competitiveness (10 top–5s in both Acc and F1). **Cosine (stable)** remains a close, consistent performer (2 Acc wins, 2 F1 wins; 9 top–5s in both). **Cosine (unstable)** secures no outright wins in Acc/F1 but is frequently competitive (9 Acc top–5; 8 F1 top–5), indicating high upside with occasional instability. **Mahalanobis (cholesky)** contributes a handful of wins/top–5 placements.

Interpretability (Explained Variance and Prototype Coverage). For *PC2 (EV)*, **Bray–Curtis (normalized)** is the primary winner (5 wins; 6 top–5), followed by **Chebyshev (standard)** (4 wins;

Table 7: Dataset meta and early-stopping settings (vision).

Parameter	MNIST	CIFAR-10	CIFAR-100
Num classes	10	10	100
Early stopping patience	15	15	25
Min improvement (%)	0.01	0.01	0.01
Image size	$28 \times 28 \times 1$	$32 \times 32 \times 3$	$32 \times 32 \times 3$

Table 8: Distance-layer shared parameters (all backbones).

Parameter	Value
n	1.0
ε	1e-4
Scale distances	False

6 top–5). **Minkowski (p=3.0)** also appears among leaders (2 wins; 3 top–5). For *PC90*, **Chebyshev (standard)** is the strongest (6 wins; 6 top–5), followed by **Bray–Curtis (normalized)** (4 wins; 6 top–5). Several families (**Canberra robust/standard**, **Manhattan**, **Minkowski**, **Hamming**) secure wins or numerous top–5 placements in PC90, showing that prototype coverage is highly sensitive to the chosen geometry. Overall, non–Euclidean distances reshape the representation space more aggressively than Euclidean/baseline, with Bray–Curtis (normalized) and Chebyshev (standard) the most consistent for interpretability.

Sustainability (Emissions) **Mahalanobis (standard)** is the clear sustainability leader on emissions with 7 wins and 10 top–5 placements, echoing its repeated aggregate advantage in total gCO_2 savings. A few other methods notch isolated emissions wins (e.g., **Chebyshev (smooth)** with 2 wins; **Bray–Curtis (abs)**, **Canberra (weighted)**, **Cosine (unstable)** with 1 each), but none rival Mahalanobis (standard) in consistency.

E.3 Vision: Curve plots

Accuracy dynamics across backbones (incl. PVT). Figures E.3–4 plot test accuracy over epochs for MNIST, CIFAR-10, and CIFAR-100 using MLP, CNN, ResNet50, and the transformer-style PVT backbone. The curves reveal how distance choices in the harmonic loss affect *speed*, *stability*, and *asymptotic* performance, and how these effects vary with model capacity.

MNIST. All backbones reach near-saturation rapidly. Cosine (stable) yields the smoothest ascent for MLP/CNN and remains competitive on ResNet50/PVT; Euclidean and Bray–Curtis closely track it. Mahalanobis variants occasionally show noisier plateaus but do not materially change the final ceiling. PVT converges as quickly as CNN/ResNet50, indicating that distance choice mainly affects early-epoch smoothness on this easy task.

CIFAR-10. Differences become more visible. For CNN and ResNet50, cosine (stable) consistently accelerates early learning and attains higher final accuracy than Euclidean; Bray–Curtis is generally competitive but architecture-sensitive. Mahalanobis can learn quickly at first yet tends to plateau below cosine on deeper backbones. MLP remains the bottleneck regardless of distance. PVT reflects the convolutional trends: cosine (stable) produces the most monotone, high-accuracy trajectory; Euclidean lags; Bray–Curtis sits in between.

CIFAR-100. The hardest setting amplifies gaps. On CNN/ResNet50, cosine (stable) again dominates in both convergence speed and final accuracy; Euclidean is a reliable baseline but systematically lower. Mahalanobis exhibits the most irregular dynamics—occasionally promising starts that flatten early. PVT benefits markedly from cosine (stable) and, to a lesser extent, Bray–Curtis; Euclidean trails, and Mahalanobis remains volatile. MLP saturates at a lower level across distances, highlighting the limits of shallow models.

Cross-architecture takeaways. i) *Cosine (stable)* is the most robust choice across datasets and backbones—including PVT—delivering smoother training and higher endpoints. ii) *Bray–Curtis*

is a viable alternative with moderate gains but exhibits architecture-dependent variability. iii) *Mahalanobis* provides occasional early wins but is the least stable overall, often converging below cosine. iv) *Euclidean* remains a solid baseline yet is consistently outperformed by cosine on the more complex settings. Overall, adding PVT corroborates the main trend observed with CNN/ResNet50: geometry matters, and cosine-based harmonic losses provide the most reliable improvements in both optimization dynamics and final accuracy.

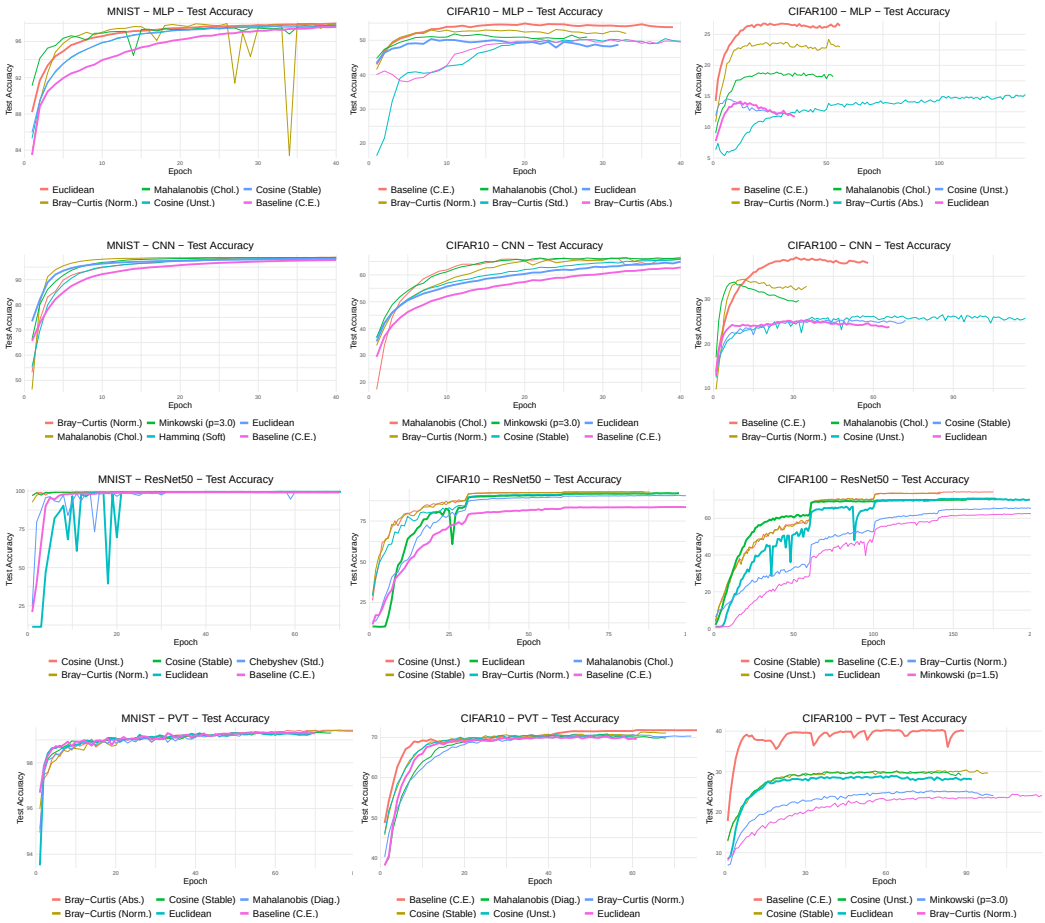

Figure 4: Vision: Curve Plots (MLP, CNN, ResNet50, PVT).

E.4 Vision: Tables

The empirical evaluation of non-Euclidean harmonic losses across MNIST, CIFAR-10, and CIFAR-100 with MLP, CNN, and ResNet50 backbones reveals several consistent patterns.

Model Performance. Cosine distance emerges as the most reliable performer across architectures and datasets. In both stable and unstable variants, cosine harmonic loss consistently improves test accuracy and F1 relative to Euclidean, with gains most pronounced in deeper models (CNNs and ResNets) and in medium-complexity datasets such as CIFAR-10. Bray–Curtis offers modest gains in certain contexts but is less consistent, while Mahalanobis can improve accuracy on simple datasets (e.g., MNIST) but often lags behind cosine in more challenging regimes. Euclidean harmonic loss, while better than cross-entropy in terms of stability, is consistently outperformed by cosine-based alternatives.

Interpretability. Distances strongly reshape the geometry of the learned representations. Cosine and Bray–Curtis often yield large improvements in explained variance (EV), indicating more compact feature spaces aligned with class prototypes. Mahalanobis produces the most dramatic gains in EV, frequently approaching full variance explanation, but this comes at the cost of stability and

Table 9: Results for CIFAR100 CNN. Parentheses: % changes w.r.t. Baseline (Cross-Entropy).

Method	Acc	F1	gCO_2eq	EV	PC90%
Baseline	0.3795	0.3795	1.18	0.459295	49.3333
Bray-Curtis (Norm.)	0.3229 (-14.91%)	0.3182 (-16.16%)	0.8132 (30.94%)	0.9094 (98%)	2.6667 (94.59%)
Mahalanobis (Chol.)	0.2927 (-22.86%)	0.2921 (-23.04%)	0.727 (38.25%)	0.341 (-25.75%)	50 (-1.35%)
Cosine (Unst.)	0.2602 (-31.44%)	0.2667 (-29.73%)	2.1156 (-79.68%)	0.5306 (15.52%)	45 (8.78%)
Cosine (Stable)	0.2501 (-34.09%)	0.2516 (-33.71%)	1.4263 (-21.14%)	0.5216 (13.57%)	45 (8.78%)
Euclidean	0.2413 (-36.4%)	0.2431 (-35.95%)	1.2866 (-9.28%)	0.4362 (-5.02%)	50 (-1.35%)

Table 10: Results for CIFAR100 MLP. Parentheses: % changes w.r.t. Baseline (Cross-Entropy).

Method	Acc	F1	gCO_2eq	EV	PC90%
Baseline	0.2617	0.2582	0.85	0.285203	50.0
Bray-Curtis (Norm.)	0.226 (-13.64%)	0.2191 (-15.15%)	0.8938 (-5.35%)	0.9843 (245.11%)	1 (98%)
Mahalanobis (Chol.)	0.1833 (-29.96%)	0.1811 (-29.85%)	1.0889 (-28.34%)	0.0354 (-87.57%)	50 (-0%)
Bray-Curtis (Abs.)	0.1444 (-44.81%)	0.1392 (-46.07%)	2.1255 (-150.51%)	0.5317 (86.43%)	47.6667 (4.67%)
Cosine (Unst.)	0.1237 (-52.74%)	0.1186 (-54.06%)	0.5064 (40.31%)	0.3799 (33.21%)	40.6667 (18.67%)
Euclidean	0.119 (-54.53%)	0.1222 (-52.69%)	0.589 (30.58%)	0.2437 (-14.55%)	50 (-0%)

efficiency. Prototype coverage (PC90%) tends to shrink under cosine and Mahalanobis, highlighting sharper clustering effects: models assign fewer prototypes to cover 90% of variance, making the representation space more interpretable but less evenly distributed.

Sustainability. Sustainability outcomes mirror performance trends. Cosine distances typically reduce carbon emissions relative to Euclidean, in some cases by up to 40%, making them both effective and energy-efficient. Bray–Curtis shows mixed results, with occasional emission savings but less consistent behavior. Mahalanobis tends to incur higher emissions, reflecting the computational overhead of covariance estimation and matrix operations. Shallow architectures (MLPs) show less differentiation across distances in emissions, while deeper backbones amplify both the benefits (cosine) and costs (Mahalanobis).

Trade-offs. Taken together, the results confirm that distance choice is not neutral in harmonic loss. Cosine provides the most favorable balance across performance, interpretability, and sustainability, representing the strongest general-purpose alternative to Euclidean. Bray–Curtis occupies a middle ground, offering interpretability benefits without always delivering accuracy or efficiency gains. Mahalanobis maximizes interpretability at a clear sustainability cost, making it attractive primarily when prototype clarity outweighs computational expense. Euclidean serves as a stable but suboptimal baseline.

Conclusion. This systematic study establishes that non-Euclidean harmonic losses provide a flexible and effective design space. In particular, cosine distance offers a compelling replacement for cross-entropy and Euclidean harmonic loss in vision tasks, consistently improving accuracy, interpretability, and sustainability. These findings position distance-tailored harmonic losses as a promising avenue for advancing deep learning models that are not only accurate but also more transparent and energy-conscious.

E.5 Vision: Sustainability

E.5.1 MNIST

Figure 5 summarizes the *carbon deltas* (gCO_2eq relative to cross-entropy) when swapping the training objective for harmonic-loss variants on MNIST across four backbones.

MLP. Most distances reduce per–step emissions vs. cross-entropy (green bars), with the largest savings from heavier geometry that replaces the softmax/cross-entropy path (e.g., Mahalanobis/-standardized, Chebyshev). Euclidean and Bray–Curtis yield modest savings; only a few variants

Table 11: Results for CIFAR100 ResNet50. Parentheses: % changes w.r.t. Baseline (Cross-Entropy).

Method	Acc	F1	gCO_2eq	EV	PC90%
Baseline	0.6983	0.6969	87.77	0.107216	50.0
Cosine (Stable)	0.7357 (5.35%)	0.736 (5.61%)	72.9745 (16.85%)	0.5979 (457.66%)	8 (84%)
Cosine (Unst.)	0.7323 (4.87%)	0.7332 (5.21%)	71.7592 (18.24%)	0.5857 (446.27%)	8 (84%)
Bray-Curtis (Norm.)	0.655 (-6.19%)	0.6513 (-6.54%)	106.4049 (-21.24%)	0.7131 (565.08%)	6 (88%)
Mahalanobis (Chol.)	0.6274 (-10.15%)	0.6239 (-10.47%)	138.9317 (-58.3%)	0.7353 (585.81%)	17.5 (65%)
Euclidean	0.7055 (1.03%)	0.7062 (1.33%)	97.432 (-11.01%)	0.5679 (429.66%)	25.5 (49%)

Table 12: Results for CIFAR10 CNN. Parentheses: % changes w.r.t. Baseline (Cross-Entropy).

Method	Acc	F1	gCO_2eq	EV	PC90%
Baseline	0.6278	0.6269	1.12	0.688081	9.0
Mahalanobis (Chol.)	0.6644 (5.82%)	0.6642 (5.95%)	1.1139 (0.68%)	0.4752 (-30.93%)	50 (-455.56%)
Bray-Curtis (Norm.)	0.6597 (5.08%)	0.6551 (4.5%)	1.1489 (-2.45%)	0.8913 (29.54%)	4.3333 (51.85%)
Minkowski (p=3.0)	0.6589 (4.95%)	0.6593 (5.17%)	1.1598 (-3.42%)	0.5425 (-21.15%)	50 (-455.56%)
Cosine (Stable)	0.6584 (4.87%)	0.6566 (4.74%)	1.1663 (-3.99%)	0.647 (-5.97%)	18.6667 (-107.41%)
Euclidean	0.6495 (3.45%)	0.6476 (3.31%)	1.1228 (-0.12%)	0.6582 (-4.34%)	14.3333 (-59.26%)

Table 13: Results for CIFAR10 MLP. Parentheses: % changes w.r.t. Baseline (Cross-Entropy).

Method	Acc	F1	gCO_2eq	EV	PC90%
Baseline	0.5397	0.5385	0.53	0.346504	47.0
Bray-Curtis (Norm.)	0.5224 (-3.21%)	0.5201 (-3.41%)	0.5264 (0.81%)	0.967 (179.07%)	1 (97.87%)
Mahalanobis (Chol.)	0.5087 (-5.75%)	0.5088 (-5.51%)	0.458 (13.7%)	0.0522 (-84.94%)	50 (-6.38%)
Bray-Curtis (Abs.)	0.4934 (-8.59%)	0.4924 (-8.55%)	0.6313 (-18.96%)	0.2434 (-29.76%)	50 (-6.38%)
Bray-Curtis (Std.)	0.4931 (-8.64%)	0.4935 (-8.35%)	0.6435 (-21.25%)	0.2906 (-16.14%)	50 (-6.38%)
Euclidean	0.4871 (-9.74%)	0.4852 (-9.9%)	0.4303 (18.92%)	0.4303 (24.19%)	42.3333 (9.93%)

Table 14: Results for CIFAR10 ResNet50. Parentheses: % changes w.r.t. Baseline (Cross-Entropy).

Method	Acc	F1	gCO_2eq	EV	PC90%
Baseline	0.843	0.8431	48.65	0.257211	50.0
Cosine (Stable)	0.9262 (9.87%)	0.9262 (9.86%)	40.6776 (16.39%)	0.7559 (193.9%)	5 (90%)
Cosine (Unst.)	0.9234 (9.54%)	0.9234 (9.53%)	29.3968 (39.58%)	0.761 (195.86%)	5 (90%)
Bray-Curtis (Norm.)	0.9193 (9.05%)	0.9192 (9.02%)	45.6222 (6.23%)	0.7883 (206.49%)	5 (90%)
Chebyshev (Std.)	0.905 (7.36%)	0.905 (7.34%)	48.5505 (0.21%)	0.9995 (288.59%)	1 (98%)
Euclidean	0.9185 (8.96%)	0.9185 (8.94%)	45.8759 (5.71%)	0.683 (165.56%)	25.5 (49%)

Table 15: Results for MNIST CNN. Parentheses: % changes w.r.t. Baseline (Cross-Entropy).

Method	Acc	F1	gCO_2eq	EV	PC90%
Baseline	0.9782	0.9782	1.19	0.585633	10.6667
Bray-Curtis (Norm.)	0.9889 (1.09%)	0.9888 (1.09%)	1.1348 (4.42%)	0.7225 (23.38%)	13.6667 (-28.12%)
Mahalanobis (Chol.)	0.9879 (1%)	0.9879 (0.99%)	1.0639 (10.39%)	0.4673 (-20.2%)	36.3333 (-240.63%)
Minkowski (p=3.0)	0.9877 (0.97%)	0.9876 (0.96%)	1.1154 (6.06%)	0.4195 (-28.37%)	49.3333 (-362.5%)
Hamming (Soft)	0.9833 (0.52%)	0.9832 (0.51%)	1.1815 (0.49%)	0.3089 (-47.26%)	50 (-368.75%)
Euclidean	0.9831 (0.5%)	0.9831 (0.5%)	1.1543 (2.78%)	0.4413 (-24.65%)	20.3333 (-90.62%)

Table 16: Results for MNIST MLP. Parentheses: % changes w.r.t. Baseline (Cross-Entropy).

Method	Acc	F1	gCO_2eq	EV	PC90%
Baseline	0.976	0.9758	0.55	0.565723	10.3333
Cosine (Unst.)	0.978 (0.2%)	0.9778 (0.2%)	0.5264 (3.58%)	0.382 (-32.48%)	10 (3.23%)
Mahalanobis (Chol.)	0.9774 (0.14%)	0.9771 (0.14%)	0.5611 (-2.78%)	0.092 (-83.74%)	50 (-383.87%)
Cosine (Stable)	0.9766 (0.06%)	0.9764 (0.06%)	0.5266 (3.54%)	0.4033 (-28.71%)	9.3333 (9.68%)
Chebyshev (Std.)	0.9756 (-0.04%)	0.9754 (-0.04%)	0.5881 (-7.73%)	0.7865 (39.03%)	5.6667 (45.16%)
Euclidean	0.9799 (0.4%)	0.9798 (0.41%)	0.5221 (4.35%)	0.358 (-36.72%)	9 (12.9%)

Table 17: Results for MNIST ResNet50. Parentheses: % changes w.r.t. Baseline (Cross-Entropy).

Method	Acc	F1	gCO_2eq	EV	PC90%
Baseline	0.9909	0.9909	29.36	0.420353	50.0
Bray-Curtis (Norm.)	0.9962 (0.52%)	0.9961 (0.53%)	25.2889 (13.86%)	0.8453 (101.09%)	4 (92%)
Cosine (Unst.)	0.996 (0.51%)	0.996 (0.52%)	26.1851 (10.8%)	0.6888 (63.87%)	6 (88%)
Cosine (Stable)	0.9953 (0.44%)	0.9953 (0.45%)	26.4064 (10.05%)	0.6974 (65.91%)	6 (88%)
Mahalanobis (Chol.)	0.9938 (0.29%)	0.9938 (0.3%)	31.9246 (-8.75%)	0.9966 (137.09%)	1 (98%)
Euclidean	0.9934 (0.25%)	0.9934 (0.25%)	24.457 (16.69%)	0.9998 (137.84%)	1 (98%)

show small positive overheads. Given MNIST's simplicity and the near-saturation accuracies, these reductions likely translate into *net* greener runs because steps-to-target are comparable.

CNN. A broad set of distances are carbon-negative vs. baseline. Again, standardized Mahalanobis/Chebyshev rank among the lowest-emission options; Bray–Curtis and Euclidean remain consistently frugal. Variants that introduce extra normalization or temperature schedules can erode part of the gain but rarely flip the sign.

ResNet50. The deepest convolutional model shows the *largest* per–step savings: many distances deliver substantial negative deltas relative to cross-entropy, suggesting that replacing the softmax loss with metric-based objectives amortizes well at this scale. Only a handful of choices (e.g., certain Chebyshev/Canberra parameterizations) incur small positive overheads.

PVT (vision transformer). In contrast to the CNN family, most distances *increase* per–step emissions over the baseline. The transformer's attention and normalization stack appears less amenable to the heavier distance computations; only a couple of standardized/normalized variants produce small savings. On PVT, greener training favors the lightest geometries or retaining cross-entropy.

Takeaways. i) On MNIST, distance-based harmonic losses are often *carbon-favorable* for MLP/CNN/ResNet50, with the biggest gains on the deepest CNN; ii) these gains are not universal—PVT tends to pay a premium; iii) because test accuracy curves on MNIST converge similarly across losses, the per–step savings for CNN/ResNet50 likely convert into lower *end-to-end* energy. Practically, we recommend Euclidean/Bray–Curtis/standardized Mahalanobis for convolutional backbones, and cautious use (or kernel-fused, mixed-precision implementations) of heavier distances on transformer-style models. Reporting both per–step emissions and energy-to-target accuracy remains essential for fair sustainability claims.

E.5.2 CIFAR-10

Figure 6 reports carbon deltas in gCO_2eq relative to cross-entropy when training with harmonic-loss distances on CIFAR-10.

MLP. Most distances are *carbon–negative* versus baseline, yielding small–to–moderate per–step savings. A few choices incur mild overheads (rightmost bars), indicating that added normalization or temperature scheduling can offset the gains on shallow networks.

CNN. The pattern strengthens: a broad set of distances reduce per–step emissions relative to cross-entropy. Only a handful of variants sit near zero or slightly positive, suggesting that, for convolutional encoders on CIFAR-10, metric-based objectives are generally more frugal per step.

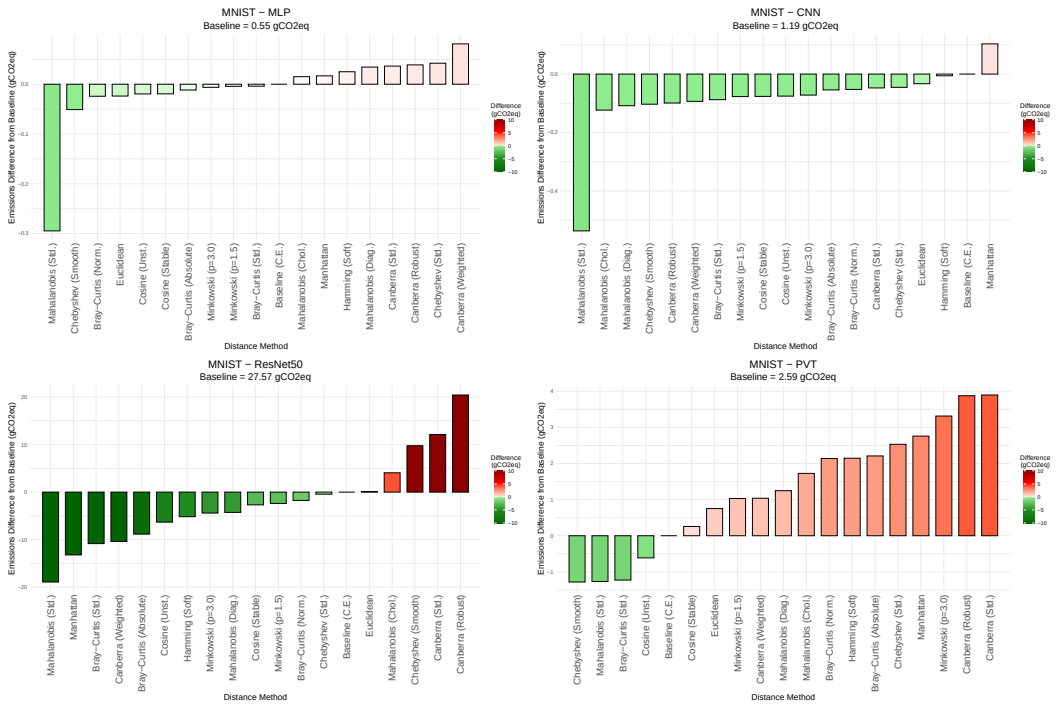

Figure 5: Carbon emission differences for MNIST across four model backbones (MLP, CNN, ResNet50, PVT) when replacing cross-entropy with harmonic loss variants. Bars show the emission difference in grams of CO_2eq relative to the baseline (cross-entropy). Values above zero indicate higher emissions than baseline, while negative values indicate greener, more sustainable outcomes.

ResNet50. Savings are *uniform and largest*: all distances fall below the baseline, with substantial negative deltas. This indicates that replacing the softmax loss amortizes particularly well at depth/width, likely due to better kernel utilization and reduced softmax/backprop overhead relative to the total compute.

PVT (vision transformer). Most distances are again carbon–negative, though the spread is narrower than ResNet50 and a couple of variants hover around parity or slightly positive. Transformers benefit, but less dramatically than deep CNNs.

Takeaways. i) On CIFAR-10, distance-based harmonic losses are typically *greener per step* for CNN/ResNet50/PVT, with the strongest effect on ResNet50; ii) MLP shows mixed but mostly favorable outcomes; iii) because our accuracy-vs-epoch curves on CIFAR-10 show similar or faster convergence for several distances, these per–step gains are likely to translate into lower *end-to-end* energy for deep backbones. Practically, we recommend adopting the more frugal distances for convolutional and transformer models and pairing per–step reports with *energy-to-target-accuracy* to substantiate sustainability claims.

E.5.3 CIFAR-100

Figure 7 shows the carbon *delta* (gCO_2eq vs. cross-entropy) when training with harmonic-loss distances on CIFAR-100.

MLP. Savings are modest and *geometry–dependent*. Light/standardized variants (e.g., cosine, Euclidean, some Minkowski/Canberra settings) are carbon–negative, while heavier norms and covariance–based Mahalanobis parameterizations flip to positive overheads. On shallow models, extra normalization steps can outweigh gains.

CNN. A broad swath of distances are carbon–negative relative to the 1.18 gCO_2eq baseline; several Mahalanobis and Bray–Curtis settings deliver the largest per–step reductions. A few choices (e.g.,

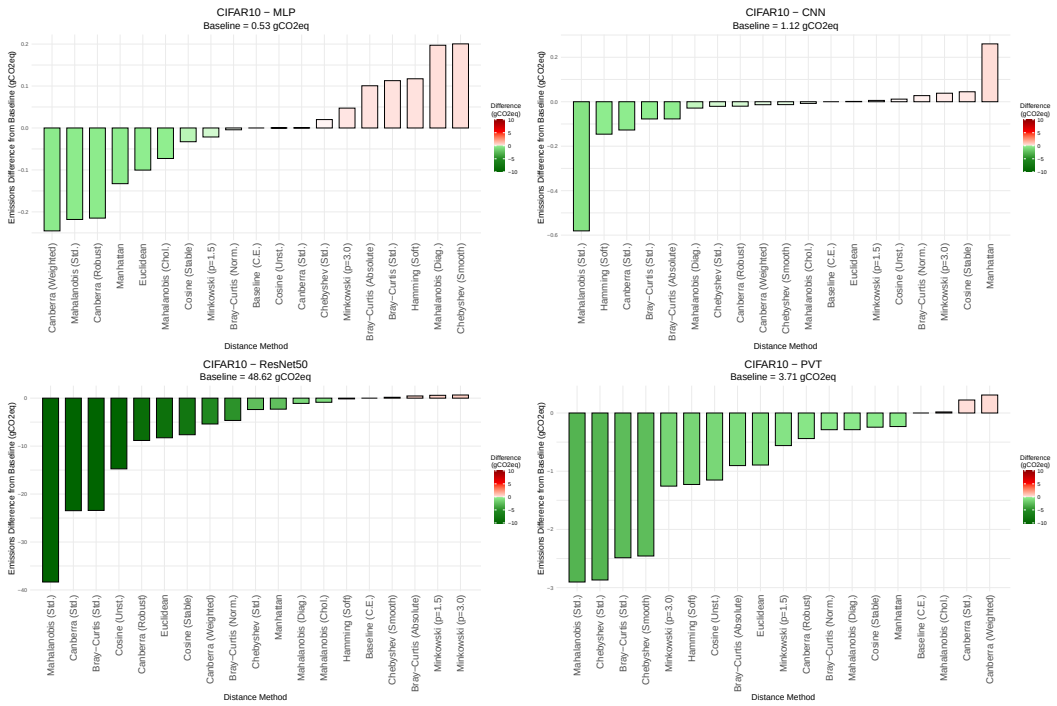

Figure 6: Carbon emission differences for CIFAR10 across four model backbones (MLP, CNN, ResNet50, PVT) when replacing cross-entropy with harmonic loss variants. Bars show the emission difference in grams of CO_2eq relative to the baseline (cross-entropy). Values above zero indicate higher emissions than baseline, while negative values indicate greener, more sustainable outcomes.

certain cosine/Canberra/Minkowski configurations) hover near parity or slightly positive, indicating mild architecture sensitivity.

ResNet50. The deepest convolutional model exhibits a *mixed but wide* spread: many distances achieve substantial savings (left cluster of dark-green bars), yet others incur clear premiums (right cluster). Thus, distance choice materially changes footprint at scale. Notably, cosine variants are among the frugal options here, whereas some Chebyshev/Minkowski/Bray–Curtis (absolute) settings are costlier.

PVT (vision transformer). Most distances are *carbon–positive* vs. the 3.67 gCO_2eq baseline, with only a couple of standardized/smoothed variants slightly negative. As on MNIST/CIFAR-10, the attention/normalization stack appears less amenable to heavier metric computations.

Takeaways. i) On CIFAR-100, harmonic distances can be *greener per step* for CNNs and selectively for ResNet50, but PVT generally pays a premium; ii) cosine tends to be frugal on deeper CNNs (and competitive on MLP), aligning with its strong accuracy dynamics, whereas several Mahalanobis/Minkowski/Chebyshev configurations increase emissions unless they deliver clear quality gains; iii) because CIFAR-100 accuracy converges differently across distances, claims of sustainability should couple per–step deltas with *energy-to-target-accuracy/perplexity*. Practically, prefer cosine/Euclidean/standardized Bray–Curtis (and selected Mahalanobis settings that are both stable and frugal) for CNN/ResNet50, and use kernel fusion + mixed precision if heavier geometries are needed on transformer backbones.

Insights across datasets: A clear trend emerges across datasets: **transformer models (PVT)** often incur higher emissions with distance-based harmonic losses, particularly on CIFAR-100 (see Figure 7), whereas **convolutional and residual networks** (CNN, ResNet50) frequently yield greener outcomes (see results in Figures 5 – 7). The sustainability benefit is especially pronounced when distances incorporate robustness (Hamming-gumbel, Canberra-robust) or covariance awareness

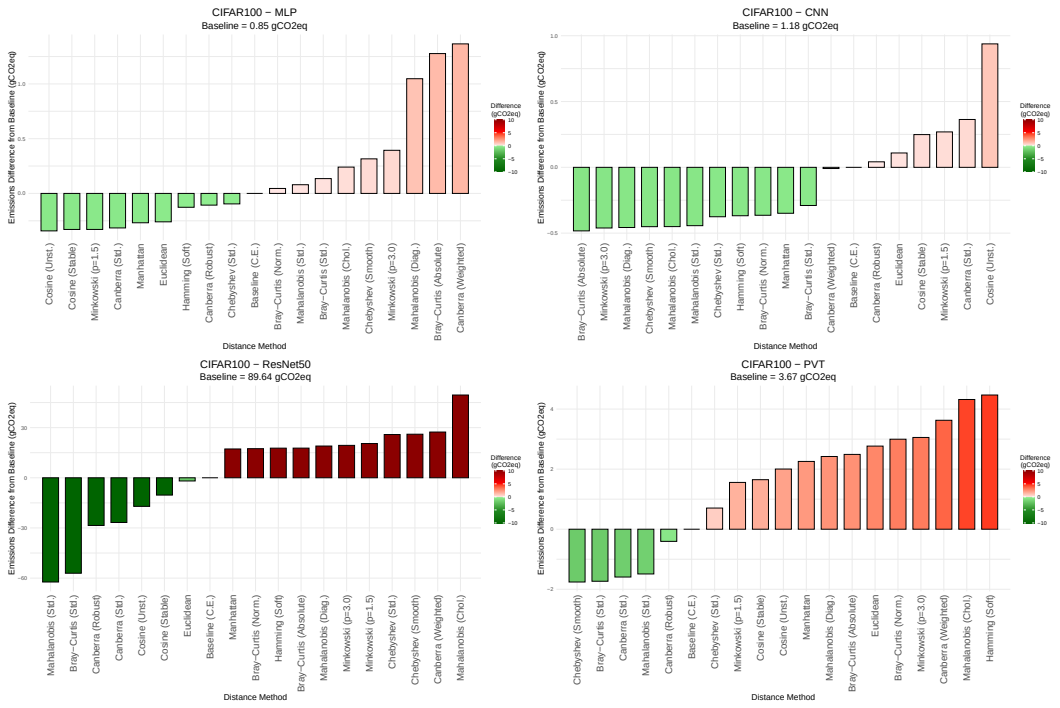

Figure 7: Carbon emission differences for CIFAR100 across four model backbones (MLP, CNN, ResNet50, PVT) when replacing cross-entropy with harmonic loss variants. Bars show the emission difference in grams of CO_2eq relative to the baseline (cross-entropy). Values above zero indicate higher emissions than baseline, while negative values indicate greener, more sustainable outcomes.

(Mahalanobis-diagonal). Simpler datasets like MNIST show limited differences, while CIFAR-10 and CIFAR-100 highlight the greater impact of distance choice on carbon footprint.

Cross-architecture insights: **MLPs** present a limited sustainability differences; emissions remain close to baseline across all distances. With **CNNs**, multiple distances (Hamming-gumbel, Mahalanobis-diagonal, Canberra-weighted) consistently reduce emissions, showing CNNs benefit most from harmonic loss efficiency. In **PVT**, harmonic losses generally increase emissions, especially on CIFAR-100, highlighting potential overhead in attention-based models. **ResNet50** demonstrates an effective integration with several distances (Hamming, Canberra, Bray–Curtis), which achieve significant reductions in emissions over baseline, indicating that deep CNNs can combine effectiveness with sustainability.

Overall, the sustainability analysis shows that harmonic losses can improve or degrade carbon efficiency depending on the backbone and dataset. The choice of distance measure therefore plays a critical role not only in accuracy but also in environmental impact, reinforcing the need for holistic evaluation across the accuracy–sustainability–interpretability triangle.

E.6 Language: Sustainability

Figure 8 reports *per–1k-step* carbon differences (gCO_2eq) when replacing cross-entropy with distance-based harmonic losses for BERT, GPT, and QWEN. Positive bars indicate higher emissions than the cross-entropy baseline (annotated atop each subplot).

Overall. Across all three backbones, distance-based losses tend to *increase* per–1k-step emissions relative to cross-entropy. The magnitude of overhead correlates with the computational complexity of the distance: lightweight cosine variants add the least overhead, while Mahalanobis and Minkowski incur the most.

BERT. Cosine (simple or temperature-scaled) yields small overheads (low single-digit gCO_2eq over a 7.87 gCO_2eq baseline), suggesting that the extra normalization and dot-product operations have modest cost. Euclidean and Bray–Curtis sit mid-pack, whereas Mahalanobis (Cholesky/standard/-diagonal) and Minkowski ($p > 2$) are consistently more carbon intensive per 1k steps.

GPT. All distances increase emissions over the 60.36 gCO_2eq baseline, with a clearer spread: cosine remains the most frugal among alternatives; Euclidean and Manhattan are mid-range; Mahalanobis (any parameterization) and Minkowski/L2 are the heaviest. This indicates that the per-step FLOPs and memory traffic of covariance-related computations (and higher-order norms) become more pronounced at GPT scale.

QWEN. For this larger model (baseline 75.29 gCO_2eq), the methods we evaluated (Minkowski/L2 and Euclidean) both raise per–1k-step emissions, with Minkowski/L2 showing a substantial increase. Although the set of distances is smaller here, the pattern mirrors GPT: heavier metrics cost more per step as model width/depth grows.

Implications. i) If *Green AI* considerations are primary, cosine-based harmonic losses are the most promising drop-in replacements, especially on encoder-style models (BERT). ii) Mahalanobis and Minkowski should be justified by clear accuracy or stability gains, as they carry the largest per-step carbon premiums. iii) Reported values are per–1k-step; end-to-end footprint also depends on *steps-to-target-quality*. Thus, a distance that reduces time-to-accuracy could still yield net carbon savings even with higher per-step cost.

Summary. Distance choice in harmonic loss is not carbon-neutral: cosine variants introduce minimal overhead; Euclidean/Bray–Curtis are moderate; Mahalanobis/Minkowski are expensive. Any claimed performance gains from richer geometries should be weighed against these systematic energy costs, preferably via *energy-normalized* quality metrics (e.g., accuracy per kWh).

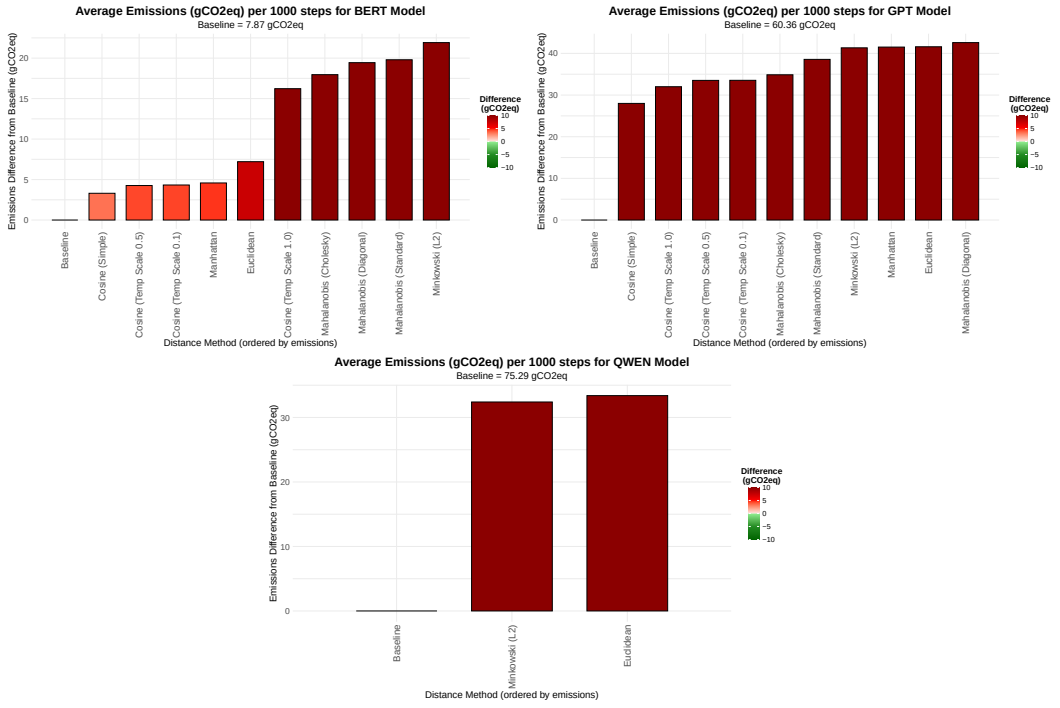

Figure 8: Carbon emission differences for LLM pretraining on OpenWebText (BERT, GPT2, QWEN) when replacing cross-entropy with harmonic loss variants. Bars show the emission difference in grams of CO_2eq relative to the baseline (cross-entropy). Values above zero indicate higher emissions than baseline, while negative values indicate greener, more sustainable outcomes.

E.7 LANGUAGE: INTERPRETABILITY

Mechanistic and representation-level interpretability of large language models (LLMs) increasingly leverages the hypothesis that internal activations admit *approximately linear* structure: many features behave like directions in an activation space, and linear operations can steer or probe them (Elhage et al., 2022; Huben et al., 2024; Turntrout, 2023). Within this paradigm, Principal Component Analysis (PCA) is a simple, well-understood lens for: i) summarizing dominant sources of variance in activations; ii) stabilizing analyses by denoising; and (iii) producing human-auditable axes that can be inspected, correlated with concepts, and tracked over time.

Given a layer ℓ with residual-stream activations $H_\ell \in \mathbb{R}^{N\times d}$ collected across N tokens (or prompts), PCA factorizes H_ℓ via SVD to yield orthogonal directions $\{u_k\}_{k=1}^{d}$ ordered by explained variance. In practice this supports:

1. **Concept probing and visualization.** Projections onto top PCs often align with semantically meaningful contrasts; e.g., the first PC of GPT-style embeddings correlated with human well-being judgments in zero-shot tests (FAR AI, 2023), and per-layer PCA can reconstruct or predict response modes in GPT-2 (Jorgensen, 2023).
2. **Diagnosing and localizing phenomena.** Layer-wise or head-wise PCA reveals where variance concentrates, helping localize depth at which concepts emerge or consolidate (complementary to linear probing) (Zhou et al., 2024). Tracking *subspace distance* across checkpoints detects representational drift during fine-tuning or domain shift.
3. **Sanity checks and baselines.** With growing interest in sparse autoencoders (SAEs) for monosemantic features (Huben et al., 2024), PCA serves as a transparent baseline decomposition: if SAEs meaningfully improve sparsity/faithfulness over PCA while matching reconstruction, that strengthens the interpretability claim (Templeton et al., 2023).

PCA is most compelling under: a) approximately linear feature superposition and b) high signal-to-noise in dominant directions. Toy and empirical studies argue that Transformers often encode many features as *directions* (superposition) (Elhage et al., 2022), and even simple linear additions to activations can steer model behavior (Turntrout, 2023). PCA then becomes an appropriate first-pass tool to:

- extract high-variance axes that frequently correlate with coherent features or tasks,
- reduce dimensionality before causal tests (e.g., ablate/project-out a PC and re-evaluate behavior),
- build compact surrogates (e.g., PCA embeddings for downstream analyses or compression) (Bengtsson et al., 2025; He et al., 2024).

Under widely observed linear-structure assumptions in Transformer activations, PCA offers an interpretable, testable starting point: it surfaces dominant directions, supports hypothesis generation, and provides quantitative targets for more advanced decompositions.

F Results on Toy Datasets: Modulo Addition

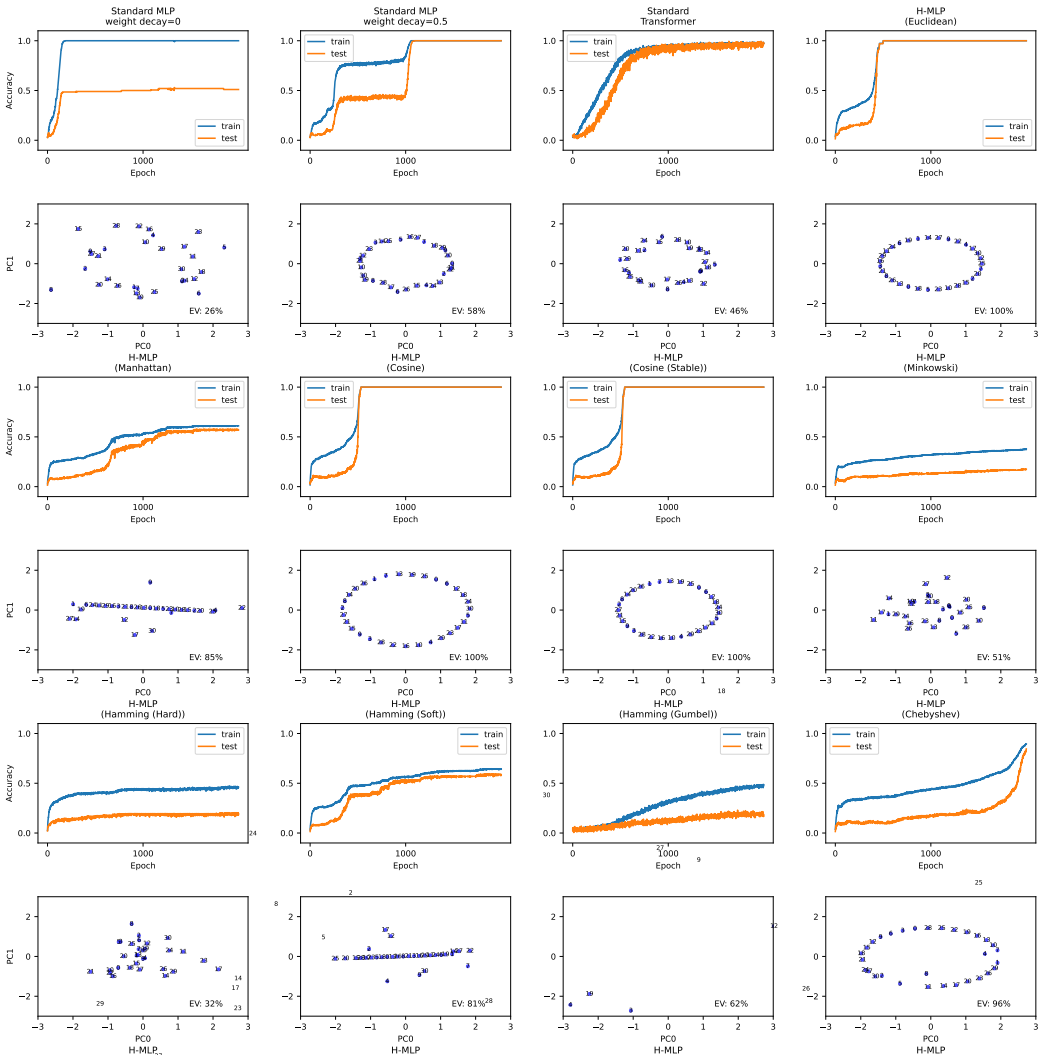

Figure 9: Results on standard MLP trained for modular addition. The harmonic model trained for modular addition generalizes quickly without grokking. Moreover, the embedding forms a perfect 2D circle. EV in the plot represents the explained variance by the first two principal components of the embedding.