MAGE: MATRIX-WEIGHTED ATTENTION GRAPH ENERGIES FOR LLM FAMILY FINGERPRINTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Limited transparency in large language model (LLM) development motivates provenance analysis to distinguish independently developed models from derivatives. We introduce Matrix-weighted Attention Graph Energies (MAGE), a framework for extracting attention metrics that can be used as model fingerprints. MAGE represents self-attention as a matrix-weighted graph, with tokens as nodes and attention-weighted value–output projection matrices as edge weights. Inspired by this representation and the Laplacian quadratic energy form, we define three attention energy metrics. Each metric's layerwise profiles show high within-model Pearson correlations across the evaluated prompts. This consistency despite input-dependent attention supports their use as model fingerprints. To evaluate the utility of these profiles, we construct a model family classifier that compares the three metrics' z-score-normalized layerwise profiles using Dynamic Time Warping (DTW). With a fixed bank of 23 reference models from six families, the classifier achieves 25/30 accuracy in a development evaluation on models outside the bank. Separately, predictions for three of four DeepSeek distilled models match their documented Qwen or Llama base lineages rather than their DeepSeek product identity. We also evaluate robustness under LoRA-based supervised fine-tuning: the fixed classifier retains correct family predictions for all 32 post-training checkpoints from 8 source models. These results suggest that MAGE captures consistent attention characteristics across prompts useful for model family identification and base-lineage analysis. Our code will be available at: https://github.com/piberius0421/mage_llm_fingerprint after acceptance.

1 INTRODUCTION

Limited disclosure of LLM development makes model origins and lineages difficult to verify. OpenAI raised concerns about DeepSeek's potential distillation in 2025, while Anthropic alleged unauthorized distillation by companies including Moonshot AI in 2026 (Nellis et al., 2025; Anthropic, 2026). Publicly released derivatives may also have product identities distinct from their base lineages, motivating fingerprints independent of names or developers' accounts.

Existing LLM fingerprinting approaches insert identifying signals into models or extract signatures from trained models (Xu et al., 2024; Jin et al., 2024). We examine internal forward-pass characteristics in models with accessible weights and attention, without inserting fingerprints. Our central question is whether input-dependent computations reveal recurring structures characteristic of a model.

Self-attention aggregates token values according to attention probabilities, admitting a graph interpretation with tokens as nodes and information transfer as weighted edges (Vaswani et al., 2017). Noël (2025) uses the Dirichlet energy of scalar-weighted attention graphs for hallucination diagnosis; a subsequent study analyzes model-specific spectral fingerprints through attention-graph spectra (Noël, 2026). These scalar edge weights reflect attention probabilities but do not explicitly incorporate the value projection W_V and output projection W_O that transform the transmitted messages.

We introduce Matrix-weighted Attention Graph Energies (MAGE), a framework for extracting attention metrics that can be used as model fingerprints. It represents self-attention as a matrix-weighted graph incorporating attention probabilities and value–output projections. Inspired by this representation and the Laplacian quadratic energy form, we define three layerwise attention energy metrics—

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

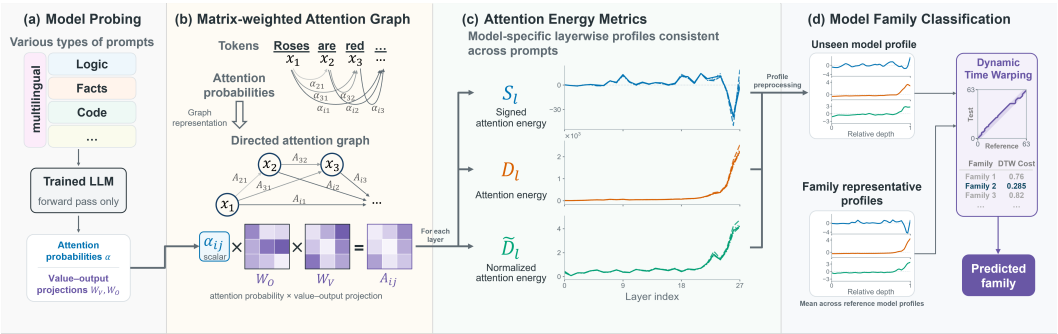


Figure 1: Overview of MAGE. (a) Diverse prompts probe a trained LLM via forward passes. (b) Attention probabilities and value–output projections define matrix-weighted edges between token nodes. (c) Three attention energy metrics reveal model-specific layerwise profiles consistent across prompts. (d) DTW matches an unseen model's profile to family representatives for classification.

Signed Attention Energy S_l, Attention Energy D_l, and Normalized Attention Energy $\widetilde{D}_l$—and combine them into a layerwise fingerprint vector. Across the evaluated prompts, each metric's layerwise profiles generally exhibit high Pearson correlations within the same model. This observation suggests recurring model-specific layerwise structure despite input-dependent attention, supporting its use for fingerprinting.

To evaluate the utility of these characteristics, we construct a model family classifier that compares z-score-normalized profiles of all three metrics using Dynamic Time Warping (DTW). We separately evaluate family classification of 30 models outside a fixed reference bank, and whether predicted families of four DeepSeek distilled models match their publicly documented base lineages (DeepSeek-AI, 2025).

Our contributions are as follows: (1) **Representation and metrics.** We define three metrics inspired by a matrix-weighted graph representation of self-attention and the Laplacian quadratic energy form. (2) **Recurring profile characteristics.** We analyze within-model profile consistency across prompts and profile relationships across models and families. (3) **Utility evaluation.** We assess the identifying information and scope of the proposed fingerprint through model family classification and base-lineage attribution for distilled models.

2 Related Work and Background

We review LLM fingerprinting and the background underlying MAGE: self-attention causality, matrix-weighted graphs, and the Laplacian quadratic energy form.

2.1 LLM Fingerprinting and Model Provenance

LLM fingerprinting constructs and verifies identifying signals to analyze model identity, ownership, or provenance (Xu et al., 2024; Jin et al., 2024). Instructional fingerprinting learns a backdoor associating private-key instructions with responses through lightweight instruction tuning for ownership verification (Xu et al., 2024). MergePrint learns fingerprints with pseudo-merged models to support black-box ownership verification after model merging (Yamabe et al., 2025).

Among approaches extracting signals from existing models, ProFLingo optimizes queries through white-box access without changing the original model's parameters, then tests whether suspect models are derived from it through black-box responses (Jin et al., 2024). HuRef exploits the relative stability of parameter-vector directions during subsequent training, converting quantities invariant to parameter rearrangements into human-readable image fingerprints for base model identification (Zeng et al., 2024).

2.2 Self-Attention and Causality

Let $x_i^{(l)} \in \mathbb{R}^C$ be the column vector for token i at layer l, and $X^{(l)}$ the block collecting these vectors, where C is the model's hidden dimension. Normalized inputs are denoted by $\bar{X}^{(l)} = \mathcal{N}_l(X^{(l)})$, with token representations $\bar{x}_i^{(l)}$ and a normalization operation $\mathcal{N}_l$ such as LayerNorm or RMSNorm (Ba et al., 2016; Zhang & Sennrich, 2019).

For head dimension d_h, $W_V^{(l,h)} \in \mathbb{R}^{d_h \times C}$ maps tokens into the head's value space, and the column block $W_O^{(l,h)} \in \mathbb{R}^{C \times d_h}$ of the output projection maps back into token space. Attention probabilities $\alpha_{ij}^{(l,h)} \geq 0$ sum to one over allowed key positions and satisfy $\alpha_{ij}^{(l,h)} = 0$ for $j > i$ under causal masking. Omitting biases, the attention output is (Vaswani et al., 2017)

$$\sum_h \sum_{j \leq i} \alpha_{ij}^{(l,h)} W_O^{(l,h)} W_V^{(l,h)} \bar{x}_j^{(l)}. \tag{1}$$

Here, $\alpha_{ij}^{(l,h)}$ determines aggregation weights between tokens, while $W_O^{(l,h)} W_V^{(l,h)}$ linearly transforms the transmitted vector information. Query, key, and value definitions and the causal softmax formulation are given in Appendix C.1.

2.3 Matrix-weighted Graph and Laplacian Quadratic Energy

A matrix-weighted graph $\mathcal{G} = (V, E, \{A_e\}_{e \in E})$ comprises nodes $V = \{1, \ldots, n\}$, edges E, and edge weights $A_e \in \mathbb{R}^{d \times d}$, with state $x_i \in \mathbb{R}^d$ at node i. Matrix weights express componentwise weighting and coupling between state components (Trinh et al., 2018; Trinh & Ahn, 2026, Section 1.1). A directed edge $e = (j, i)$ represents interaction $j \to i$, distinct from the reverse direction; an undirected edge $e = \{i, j\}$ has one weight without distinguishing direction (Trinh & Ahn, 2026, Section 1.2).

For an undirected graph without self-loops, assume symmetric positive semidefinite weights $A_e = A_e^\top \succeq 0$, meaning $z^\top A_e z \geq 0$ for all z. Counting each edge once in E, the matrix-weighted Laplacian $L_\mathcal{G} \in \mathbb{R}^{nd \times nd}$ has blocks (Trinh et al., 2018; Trinh & Ahn, 2026, Definition 2.1)

$$(L_\mathcal{G})_{ij} = \begin{cases} \sum_{e \in E:\, i \in e} A_e, & i = j, \\ -A_e, & i \neq j,\ e = \{i, j\} \in E, \\ 0_{d \times d}, & \text{otherwise.} \end{cases} \tag{2}$$

For the stacked state $x = [x_1^\top, \ldots, x_n^\top]^\top$, the Laplacian quadratic energy form is defined by (Trinh & Ahn, 2026, Corollary 2.1)

$$x^\top L_\mathcal{G} x = \sum_{e = \{i,j\} \in E} (x_i - x_j)^\top A_e (x_i - x_j) \geq 0. \tag{3}$$

Each term measures differences between connected node states with directional weighting specified by A_e; the energy is zero when all states are identical.

3 MAGE: Matrix-Weighted Attention Graph Energies

MAGE combines token representations and attention probabilities obtained from a model's forward pass with its value–output projections. With tokens as nodes and matrices combining attention probabilities and value–output projections as edge weights, a single matrix-weighted attention graph captures both the aggregation weights between tokens and the transformations of the transmitted vector information.

Inspired by this representation and the concept of Laplacian quadratic energy in matrix-weighted graph theory, we define three attention energy metrics. Their layerwise profiles describe how token

interactions vary across model layers, and we found that the three metrics exhibit recurring model-specific layerwise structure across diverse prompts. Building on this observation, we combine these metrics into a layerwise fingerprint vector and use it to construct a model family classifier.

3.1 Matrix-Weighted Graph Representation of Self-Attention

For the normalized attention input $\bar{X}^{(l)}$ in Section 2.2, each token position is a node, with $\bar{x}_i^{(l)} \in \mathbb{R}^C$ as the state of node i. Denoting the value–output projection of head h by $M^{(l,h)}$, we define the matrix-valued edge weight corresponding to information transfer from token j to token i as

$$\begin{aligned} M^{(l,h)} &= W_O^{(l,h)} W_V^{(l,h)} \in \mathbb{R}^{C\times C}, \\ A_{ij}^{(l,h)}\big(\bar{X}^{(l)}\big) &= \alpha_{ij}^{(l,h)}\big(\bar{X}^{(l)}\big) M^{(l,h)}. \end{aligned} \tag{4}$$

This weight jointly represents the aggregation weight determined by the attention probability and the transformation of vector information through the value–output projection. While $M^{(l,h)}$ is determined by the model's weights, $A_{ij}^{(l,h)}$ depends nonlinearly on the input through the attention probability. For grouped-query attention, let $g(h)$ denote the KV head corresponding to head h, and use $M^{(l,h)} = W_O^{(l,h)} W_V^{(l,g(h))}$. For standard multi-head attention, $g(h) = h$.

When token i attends to token j, information flows in the direction $j \to i$, and the causal mask imposes $\alpha_{ij}^{(l,h)} = 0$ for $j > i$. Thus, the representation at each layer and head is an input-dependent directed matrix-weighted attention graph. Hereafter, we omit the input argument $\bar{X}^{(l)}$ from $A_{ij}^{(l,h)}$ and $\alpha_{ij}^{(l,h)}$.

3.2 Attention Energy Metrics

The Laplacian quadratic energy form in Section 2.3 measures differences between connected node states using matrix weights (Trinh et al., 2018; Trinh & Ahn, 2026). Inspired by this edgewise quadratic form, we aggregate contributions from token state differences $d_{ij}^{(l)} = \bar{x}_i^{(l)} - \bar{x}_j^{(l)}$ along the directed edges of the attention graph. Its edge weights are generally non-symmetric and are not guaranteed to be positive semidefinite. We therefore define a signed metric using the original weights and a nonnegative metric using a positive semidefinite weight construction.

Let T be the number of tokens in the analyzed sequence and H the number of query heads in the layer. The sums below range over $h = 1, \dots, H$ and $i, j = 1, \dots, T$. The factor $1/(HT)$ averages the attention-probability-weighted terms over query tokens and heads. Terms excluded by the causal mask have zero weight, and self-loops contribute nothing because $d_{ii}^{(l)} = 0$.

Only the symmetric part of an edge weight contributes to the quadratic form $d^\top A d$. Specifically, with $\mathrm{sym}(A) = (A + A^\top)/2$, we have $d^\top A d = d^\top \mathrm{sym}(A) d$.

Definition 1 (Signed Attention Energy)**.** We define S_l by aggregating this quantity over token pairs and heads:

$$S_l = \frac{1}{HT} \sum_{h,i,j} \big(d_{ij}^{(l)}\big)^\top \mathrm{sym}\big(A_{ij}^{(l,h)}\big) d_{ij}^{(l)}. \tag{5}$$

Each term equals $\alpha_{ij}^{(l,h)} (d_{ij}^{(l)})^\top M^{(l,h)} d_{ij}^{(l)}$, so S_l aggregates signed inner products between token state differences and their value–output-projected counterparts. S_l can be positive, negative, or zero. The name Signed Attention Energy refers to this energy-like quadratic form that permits either sign.

To construct a nonnegative energy, we retain the attention probability and replace $M^{(l,h)}$ with $\big(M^{(l,h)}\big)^\top M^{(l,h)}$, defining a positive semidefinite weight $B_{ij}^{(l,h)}$. With this weight, the quadratic form of a token state difference becomes the squared magnitude of its value–output-projected counterpart, weighted by the attention probability.

Definition 2 (Attention Energy)**.** Aggregating these terms gives the Attention Energy D_l:

$$B_{ij}^{(l,h)} = \alpha_{ij}^{(l,h)} \big(M^{(l,h)}\big)^\top M^{(l,h)} \succeq 0,$$
$$D_l = \frac{1}{HT} \sum_{h,i,j} \big(d_{ij}^{(l)}\big)^\top B_{ij}^{(l,h)} d_{ij}^{(l)} = \frac{1}{HT} \sum_{h,i,j} \alpha_{ij}^{(l,h)} \big\| M^{(l,h)} d_{ij}^{(l)} \big\|_2^2. \tag{6}$$

Nonnegative attention probabilities ensure $D_l \geq 0$, and a larger difference between tokens after value–output projection increases the corresponding term. This quantity reflects the scale of $M^{(l,h)}$ as well as token representations and attention probabilities.

To reduce the influence of head-specific projection scale, we normalize $M^{(l,h)}$ by its Frobenius norm and apply the same aggregation.

Definition 3 (Normalized Attention Energy)**.** We define the Normalized Attention Energy $\widetilde{D}_l$ as

$$\widetilde{M}^{(l,h)} = \frac{M^{(l,h)}}{\|M^{(l,h)}\|_F + \epsilon},$$
$$\widetilde{D}_l = \frac{1}{HT} \sum_{h,i,j} \alpha_{ij}^{(l,h)} \big\| \widetilde{M}^{(l,h)} d_{ij}^{(l)} \big\|_2^2. \tag{7}$$

Here, $\epsilon > 0$ is a constant for numerical stability, set to 10^{-12} in our implementation. We have $\widetilde{D}_l \geq 0$, and normalization is performed separately for each head before summation. D_l reflects the magnitude of token state differences after value–output projection, including the projection scale. $\widetilde{D}_l$ uses the same token state differences and attention probabilities after normalizing each head's projection by its Frobenius norm.

3.3 Layerwise Fingerprint Construction

For each prompt's forward pass, we compute the three metrics at every layer and combine their values at layer l into a layerwise fingerprint vector:

$$V_l = \begin{bmatrix} S_l & D_l & \widetilde{D}_l \end{bmatrix}^\top, \qquad l = 1, \ldots, L, \tag{8}$$

where L is the number of analyzed layers in the model. Each metric's layerwise profile is the sequence of its values in layer order, while $V_1, \ldots, V_L$ retains all three metrics together. Sections 5.2 and 5.3 analyze profile consistency across prompts and profile relationships across models and families.

For the model family classifier, an application of this fingerprint, we interpolate each prompt's profile onto a common depth grid, apply z-score normalization separately to each metric, and average over prompts to obtain a representative model profile. Section 4 describes this preprocessing, the construction of family representatives, and comparison using DTW.

4 Model Family Identification with MAGE

We construct a model family classifier based on the layerwise fingerprint vectors of the three attention energy metrics defined in Section 3. We form family representatives from reference model profiles and identify the family of an unseen model outside the reference bank by comparing its layerwise patterns with each family representative using DTW.

4.1 Reference Bank and Family Representatives

The reference bank comprises 23 models: four Llama, five Qwen, four Gemma, four SmolLM2, four Pythia, and two DeepSeek models. The complete checkpoint list and model specifications are provided in the reference bank table in Appendix A.1.

To compare models with different numbers of layers on a common basis, we express each prompt's metric profiles in terms of relative depth in $[0, 1]$ and linearly interpolate them onto 64 uniformly spaced grid points. We then apply z-score normalization along the depth axis to each metric and average over prompts to obtain a model representative profile containing all three metrics. Family representative profiles are the equally weighted arithmetic means of the model representative profiles within each family. We apply the same preprocessing to unseen models and use a fixed reference bank and fixed family representative profiles for evaluation.

4.2 Dynamic Time Warping (DTW) Matching

Dynamic Time Warping (DTW) computes similarity between two sequences by nonlinearly aligning corresponding positions while preserving their order. Sakoe & Chiba (1978) developed a dynamic programming approach to time normalization for handling speaking-rate variation in speech recognition. This alignment is suited to comparing sequences in which similar patterns appear at different positions or span different widths along the time axis.

Because models have different native layer counts, similar peaks or regions of change may appear at neighboring grid points after their profiles are interpolated onto a common 64-point depth grid. These small positional differences may reflect both differences in the original profiles and the effects of differing layer resolutions and interpolation. To accommodate this possibility, we compare layerwise patterns using DTW, which allows local positional shifts while preserving layer order.

Let Q denote the representative profile of an unseen model and C_f that of family f. At each grid point, we combine the three metric values into a three-component vector. DTW aligns these vector sequences using one shared path, with squared Euclidean distance as the local cost. Along an order-preserving path connecting the start and end points, we limit the difference between matched grid indices to at most three to prevent excessive depth warping. Denoting the minimum accumulated local cost over admissible paths by $\mathcal{D}_{\mathrm{DTW}}(Q, C_f)$, we predict the family from the set of six families $\mathcal{F}$ as

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \mathcal{D}_{\mathrm{DTW}}(Q, C_f). \tag{9}$$

The DTW recurrence, path conditions, and computational procedure are described in Appendices A.4–A.5. Section 5 presents the evaluation protocol and results for the classifier.

5 Experiments and Analysis

We prepared a set of 1,000 prompts covering diverse languages and purposes (Section 5.1). Analysis of these prompts revealed recurring layerwise profile shapes for the three attention energy metrics within the same model (Section 5.2). Comparisons across models also showed a tendency toward within-family similarity and between-family differences (Section 5.3). We then evaluated unseen model family classification and base-lineage attribution for distilled models using a model family classifier constructed from the three metrics. The results support the use of these layerwise characteristics for model identification (Sections 5.5–5.6).

5.1 Experimental Setup and Evaluation Protocol

For profile analysis, we use the 23 reference models from six families described in Section 4.1. All models receive the same 1,000 prompts, organized into 12 categories including technical explanation, summarization, translation, coding, and reasoning. The set includes prompts in multiple languages and of varying lengths to ensure input diversity. We extract all three metrics at every layer during a forward pass under a prompt-only protocol, without applying chat templates or generating responses. The reference model list, prompt composition, and extraction settings are provided in Appendices A.1, D.1, and D.2, respectively.

We measure cross-prompt consistency using Pearson correlation on each model's native layer grid. For comparisons across models and families, we construct representative profiles and compare them using the preprocessing and DTW procedure in Section 4. Model family classification is evaluated

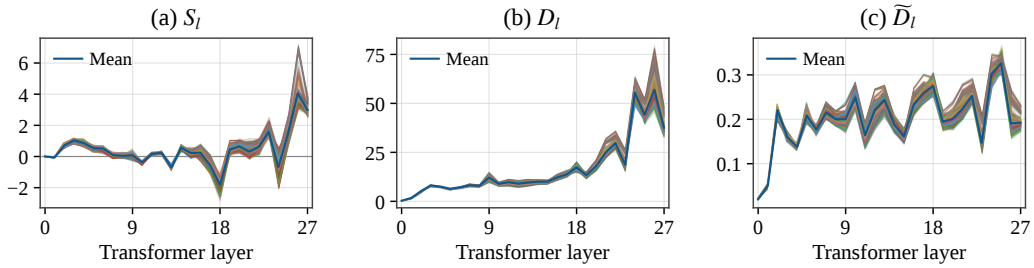


Figure 2: Layerwise profiles of attention energy metrics for Llama-3.2-3B across 1,000 prompts on 28 native layers. Thin curves show individual prompts; thick blue curves show their mean.

Table 1: Median prompt-pair Pearson correlations on native layer grids for one example model per family (499,500 pairs per metric).

Family	Model	S_l	D_l	$\widetilde{D}_l$
Llama	Llama-3.2-3B	0.9806	0.9968	0.9843
Qwen	Qwen2.5-1.5B	0.9814	0.9988	0.9955
Gemma	Gemma-2-2B	0.9627	0.9765	0.9739
SmolLM2	SmolLM2-1.7B	0.9596	0.9974	0.9876
Pythia	Pythia-2.8B	0.9492	0.9958	0.9916
DeepSeek	DeepSeek-LLM-7B-Base	0.9559	0.9976	0.9912

on 30 models outside the fixed reference bank (Table 8, Appendix D.3). Four DeepSeek distilled models are evaluated separately for attribution to their publicly documented base lineages. The results of these two evaluations are reported in Sections 5.5 and 5.6, respectively. Hyperparameter sensitivity and computational costs are reported in Appendices G and C.4.

5.2 Within-Model Profile Consistency across Prompts

To quantify the consistency of layerwise profile shapes across prompts within a model, we compute Pearson correlations between profiles from distinct prompts for each model and metric. Correlations are computed on the native layer grid without interpolation or DTW alignment. We summarize the 499,500 distinct prompt pairs formed by 1,000 prompts using the median of these Pearson correlations.

Table 1 summarizes the median prompt-pair Pearson correlations of the three metrics for example models from each family. Figure 2 shows the layerwise profiles obtained from 1,000 prompts for Llama-3.2-3B, one of the analyzed models. Its median prompt-pair Pearson correlations for S_l, D_l, and $\widetilde{D}_l$ were 0.9806, 0.9968, and 0.9843, respectively. Each panel uses values computed on the model's 28 native layers. All 23 evaluated models exhibited high median prompt-pair Pearson correlations for all three metrics. These results show that consistent layerwise profile shapes recur within each model across different prompts. Correlation distribution summaries for all models and profiles of example models from the remaining families are provided in Appendices E.1 and E.2, respectively.

5.3 Between-Model and Within-Family Profile Structure

To examine relationships between these recurring profiles across models, we use model representative profiles obtained by averaging the preprocessed prompt-level profiles of each of the 23 reference models. We compute pairwise DTW costs between these model profiles and compare each with the six family representatives used by the classifier. The median cost for the 35 within-family model pairs was 0.2181, lower than 1.5207 for the 218 between-family pairs. Although the cost ranges overlapped, models within the same family tended to have more similar layerwise profiles. With

each model included in the construction of its own family representative, all 23 models had the lowest cost for their own family. The two heatmaps illustrating these model relationships and family-level structure are provided in Appendix E.3. Section 5.5 evaluates identification performance on models outside the reference bank.

5.4 Comparison with a Scalar-Weighted Graph Baseline

Motivated by scalar attention-graph analyses and spectral fingerprinting (Noël, 2025; 2026), we construct a scalar-weighted Dirichlet energy baseline using the same preprocessing and DTW classification protocol to assess the contribution of learned value–output projections. We compare methods on 15 of the 30 ordinary unseen models, using the same 100 prompts for both the 23 reference models and the queries. The scalar baseline uses the same token representations and attention probabilities as MAGE but replaces the value–output transformation in D_l with the identity matrix I. The scalar baseline correctly classified 10/15 models, compared with 13/15 for MAGE. Among the two models misclassified by MAGE, Qwen2.5-3B-Instruct was assigned to SmolLM2 (0.570948), while its true family, Qwen (0.571429), had the second-lowest DTW cost, with a gap of 0.000481. The comparison table and experimental details are provided in Appendix H.

5.5 Unseen Model Family Classification

To assess whether the observed layerwise characteristics also support family identification beyond the reference bank, we classified 30 unseen models while keeping the 23 reference models and six family representative profiles from Section 4 fixed. We predicted one family for each model representative profile constructed from 1,000 prompts. The development evaluation correctly classified 25 of the 30 models. Among the five misclassified models, the correct family had the second-lowest DTW cost in all cases except Pythia-12B; for Gemma-2-27B, the predicted family Pythia (1.3524) and the correct family Gemma (1.3956) differed by a small cost margin of 0.0432. These results support the utility of the three metrics for family identification beyond the reference bank. The ground-truth and predicted families of all models are provided in Table 10 (Appendix F.1). Sensitivity analyses of the classification results with respect to prompt budget and composition, DTW window size, interpolation grid size, metric combination, and normalization are reported in Appendix G.

5.6 Robustness under LoRA-SFT and Base-Lineage Attribution of Distilled Models

LoRA-SFT robustness. We applied LoRA-based supervised fine-tuning to eight source models from the Llama and Qwen families using 4,000 instruction–response examples from No Robots and two seeds. LoRA targeted attention and MLP projections, including W_V and W_O. We merged adapters into the original weights at epochs 1 and 3 and extracted metrics using the same 100 evaluation prompts, which were not used for training. With the pre-training 23-model reference bank and $S_l + D_l + \widetilde{D}_l$ configuration fixed, correct family predictions were retained for all 32 post-training checkpoints (32/32). Excluding the source checkpoint wherever present in the bank and recomputing family representatives also yielded 32/32. These results demonstrate robustness of family identification under the evaluated LoRA-SFT conditions. Table 12 in Appendix F.2 reports the checkpoint-level predictions.

Distilled-model base-lineage attribution. According to the model card from DeepSeek-AI (2025), the evaluated DeepSeek distilled models were fine-tuned from Qwen or Llama checkpoints using data generated by DeepSeek-R1. Using the same classifier, we compared the predicted families of four models with their publicly documented base families. As shown in Table 2, the predictions matched the documented base families for three of the four models. DeepSeek-R1-Distill-Qwen-14B was classified as DeepSeek, which had the lowest DTW cost (0.4771), while its documented base family, Qwen (0.7909), had the second-lowest cost among the six families. These results suggest that MAGE can capture characteristics associated with a base family even in models released under a different product identity.

Table 2: Base-lineage attribution for DeepSeek distilled models, with public base families from DeepSeek-AI (2025).

Model	Public base family	MAGE prediction
DeepSeek-R1-Distill-Qwen-1.5B	Qwen	Qwen
DeepSeek-R1-Distill-Qwen-7B	Qwen	Qwen
DeepSeek-R1-Distill-Llama-8B	Llama	Llama
DeepSeek-R1-Distill-Qwen-14B	Qwen	DeepSeek

5.7 Analysis of the Results

The cross-prompt consistency of the three metrics and the similarity of profiles within families suggest that MAGE may capture recurring internal model characteristics despite input variation. We hypothesize that these layerwise profiles can serve as low-dimensional representations summarizing structural properties of model-specific latent spaces. Although the model's projection weights are fixed during evaluation, token states and attention probabilities vary across prompts. Nevertheless, recurring profiles may emerge if, under fixed learned weights, the directions and magnitudes of token state differences and their transformations have similar statistical properties at each layer. From this perspective, MAGE profiles may provide a compact summary of common properties of latent representations realized across diverse inputs and of how they are transformed.

One possible explanation for MAGE's ability to perform model family classification is that training-data composition and training procedures shared within a family shape learned weights and token representations in similar ways. Similar language and topic distributions in the training corpus, training objectives, and optimization procedures may produce family-related similarities in value–output projections and token state distributions despite differences in model scale or fine-tuning. Because MAGE explicitly incorporates learned projections, its layerwise profiles may capture how these shared training characteristics manifest in attention interactions. This interpretation links the observed within-family similarity and between-family differences and offers a possible explanation for family identification by comparing recurring profile shapes using z-score normalization and DTW. The agreement of some distilled models' predictions with their documented base families is also consistent with partial retention of characteristics formed in the base models after subsequent fine-tuning.

Testing these hypotheses requires distinguishing the contributions of training and model design. Transformer-based models differ in attention configurations and normalization designs (Gemma Team, 2024), while tokenizers and model scale may also affect the profiles. The current results show shape consistency and family-identifying information over the evaluated prompt set, while the specific correspondence between the profiles and latent spaces requires further analysis. In future work, we plan to test these hypotheses by varying training data and training procedures separately under a fixed architecture, while controlling conditions such as initialization. We also aim to track changes in projections and token state distributions alongside metric profiles across intermediate training checkpoints to investigate how cross-prompt consistency develops and where family-related commonalities originate.

6 Conclusion

We proposed MAGE, a framework for extracting attention energy metrics usable as model fingerprints, inspired by matrix-weighted self-attention graphs and the Laplacian quadratic energy form. Across 23 models evaluated with 1,000 diverse prompts, we found recurring within-model layerwise profile shapes despite input-dependent attention. A DTW-based model family classifier using these characteristics correctly classified 25 of 30 models outside the reference bank, while predictions for three of four separately evaluated DeepSeek distilled models matched their publicly documented base families. These results show that MAGE captures model-specific layerwise characteristics and that fingerprints derived from them yield meaningful results in model family identification and base-lineage attribution.

AI USE STATEMENT

In this work, we used generative AI tools for aiding or polishing writing, translating portions of the manuscript, providing feedback on research methodology, and generating the set of 1,000 prompts used in our experiments. We have not used generative AI tools for generating the research ideas or developing the mathematical formulations presented in this work.

We have reviewed all AI-assisted work. We manually reviewed AI-assisted writing and translations, and independently evaluated methodological feedback before incorporating it into the work. LLM-generated prompts used in our experiments were also verified by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 3 defines the attention energy metrics, with supporting derivations in Appendix B. Appendix A specifies the reference bank, profile preprocessing, family representative construction, and DTW classification procedure, including the numerical settings used in evaluation. Appendix C describes metric extraction, architecture-specific handling, computational precision, and attention-output reconstruction checks; Appendix C.4 reports the runtime and GPU memory measurements and their scope. Appendix D documents the prompt set, extraction settings, and evaluation protocol. Profile consistency and structure results are reported in Appendix E, and model-level classification, LoRA-SFT robustness, and base-lineage results in Appendix F. Appendix F.2 additionally specifies the LoRA-SFT training settings and source-checkpoint exclusion protocol. Appendix G reports sensitivity to prompt budget and composition, DTW window, interpolation grid, metric combination, and normalization, together with leave-one-model-out analysis. Appendix H provides the scalar-weighted baseline definition, matched evaluation protocol, and model-level results. Our code will be available at https://github.com/piberius0421/mage_llm_fingerprint after acceptance.

ACKNOWLEDGEMENTS

To be added after acceptance.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.298.

Anthropic. Detecting and preventing distillation attacks. Anthropic, February 23, 2026. URL https://www.anthropic.com/news/detecting-and-preventing-distillation-attacks. Accessed September 12, 2026.

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

DeepSeek-AI. DeepSeek-R1: Official model card. Hugging Face, 2025. URL https://huggingface.co/deepseek-ai/DeepSeek-R1. Accessed September 21, 2026.

Gemma Team. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Heng Jin, Chaoyu Zhang, Shanghao Shi, Wenjing Lou, and Y. Thomas Hou. ProFLingo: A fingerprinting-based intellectual property protection scheme for large language models. *arXiv preprint arXiv:2405.02466*, 2024. Version 3 consulted.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Stephen Nellis, Krystal Hu, Jeffrey Dastin, Anna Tong, and Katie Paul. Why blocking China's DeepSeek from using US AI may be difficult. Reuters, January 29. Republished by Investing.com, 2025. URL https://www.investing.com/news/stock-market-news/why-blocking-chinas-deepseek-from-using-us-ai-may-be-difficult-3837920. Accessed September 12, 2026.

Valentin Noël. A graph signal processing framework for hallucination detection in large language models. *arXiv preprint arXiv:2510.19117*, 2025.

Valentin Noël. Spectral archaeology: The causal topology of model evolution. *arXiv preprint arXiv:2601.03424*, 2026. Version 1 consulted.

Nazneen Rajani, Lewis Tunstall, Edward Beeching, Nathan Lambert, Alexander M. Rush, and Thomas Wolf. No Robots. Hugging Face dataset, 2023. URL https://huggingface.co/datasets/HuggingFaceH4/no_robots.

Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978. doi: 10.1109/TASSP.1978.1163055.

Minh Hoang Trinh and Hyo-Sung Ahn. *Matrix-Weighted Graphs: Theory and Applications*, volume 498 of *Lecture Notes in Control and Information Sciences*. Springer Nature Switzerland, Cham, 2026. doi: 10.1007/978-3-032-03079-5.

Minh Hoang Trinh, Chuong Van Nguyen, Young-Hun Lim, and Hyo-Sung Ahn. Matrix-weighted consensus and its applications. *Automatica*, 89:415–419, 2018. doi: 10.1016/j.automatica.2017.12.024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Jiashu Xu, Fei Wang, Mingyu Ma, Pang Wei Koh, Chaowei Xiao, and Muhao Chen. Instructional fingerprinting of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3277–3306. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.naacl-long.180.

Shojiro Yamabe, Futa Kai Waseda, Tsubasa Takahashi, and Koki Wataoka. MergePrint: Merge-resistant fingerprints for robust black-box ownership verification of large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6894–6916. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.342.

Boyi Zeng, Lizheng Wang, Yuncong Hu, Yi Xu, Chenghu Zhou, Xinbing Wang, Yu Yu, and Zhouhan Lin. HuRef: HUman-REadable fingerprint for large language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

A CONSTRUCTION OF THE MAGE MODEL FAMILY CLASSIFIER

This appendix describes the construction of the model family classifier in Section 4. We preprocess the prompt-level layerwise profiles of the three attention energy metrics defined in Section 3 on a common depth grid. We then average over prompts and over models within each family to construct family representative profiles. The same preprocessing is applied to an unseen model outside the reference bank, and its family is predicted from the Dynamic Time Warping (DTW) costs to the family representatives.

A.1 REFERENCE BANK AND INPUT PROFILES

The reference bank comprises 23 models: four Llama, five Qwen, four Gemma, four SmolLM2, four Pythia, and two DeepSeek models. Family labels follow the model series listed in Table 3, with base, instruction-tuned, and chat models from the same series assigned to the same family. Qwen2.5-Math-1.5B belongs to the Qwen family. The DeepSeek references are deepseek-llm-7b-base and deepseek-llm-7b-chat; the separately evaluated DeepSeek distilled models are excluded from the reference bank.

Table 3: The 23 reference bank checkpoints and their model specifications. Parameter counts recorded in the model metadata at collection time are expressed in units of 10^9 and rounded to two decimal places. Native layers denotes the number of transformer layers from which metrics were extracted.

Family	**Checkpoint ID**	**Parameters** (10^9)	**Model type**	**Native layers**
Llama	meta-llama/Llama-3.2-1B	1.24	Base	16
Llama	meta-llama/Llama-3.2-1B-Instruct	1.24	Instruct	16
Llama	meta-llama/Llama-3.2-3B	3.21	Base	28
Llama	meta-llama/Llama-3.2-3B-Instruct	3.21	Instruct	28
Qwen	Qwen/Qwen2.5-0.5B	0.49	Base	24
Qwen	Qwen/Qwen2.5-0.5B-Instruct	0.49	Instruct	24
Qwen	Qwen/Qwen2.5-1.5B	1.54	Base	28
Qwen	Qwen/Qwen2.5-1.5B-Instruct	1.54	Instruct	28
Qwen	Qwen/Qwen2.5-Math-1.5B	1.54	Base (Math)	28
Gemma	google/gemma-3-1b-pt	1.00	Base	26
Gemma	google/gemma-3-1b-it	1.00	Instruct	26
Gemma	google/gemma-2-2b	2.61	Base	26
Gemma	google/gemma-2-2b-it	2.61	Instruct	26
SmolLM2	HuggingFaceTB/SmolLM2-360M	0.36	Base	32
SmolLM2	HuggingFaceTB/SmolLM2-360M-Instruct	0.36	Instruct	32
SmolLM2	HuggingFaceTB/SmolLM2-1.7B	1.71	Base	24
SmolLM2	HuggingFaceTB/SmolLM2-1.7B-Instruct	1.71	Instruct	24
Pythia	EleutherAI/pythia-410m	0.41	Base	24
Pythia	EleutherAI/pythia-1b	1.01	Base	16
Pythia	EleutherAI/pythia-1.4b	1.41	Base	24
Pythia	EleutherAI/pythia-2.8b	2.78	Base	32
DeepSeek	deepseek-ai/deepseek-llm-7b-base	6.91	Base	30
DeepSeek	deepseek-ai/deepseek-llm-7b-chat	6.91	Chat	30

Let L_m be the number of native layers in model m and P_m the number of analyzed prompts. For each prompt p, we obtain the layerwise fingerprint vector V_l from Section 3.3 and denote its three components by $v_{m,p,k}(l)$, $k = 1, 2, 3$. These components correspond to Signed Attention Energy S_l, Attention Energy D_l, and Normalized Attention Energy $\widetilde{D}_l$, respectively, with $l = 1, \ldots, L_m$. For example, $v_{m,10,1}(5)$ is the Signed Attention Energy at the fifth layer of model m for the tenth prompt. The classifier takes three $P_m \times L_m$ matrices as input, one per metric. Signed Attention Energy is used with its extracted sign preserved.

We use the same 1,000 prompts for every reference and unseen model. The three input matrices for each model share the same prompt and layer ordering. Appendix D describes the prompt set and extraction settings, and Appendix C details metric extraction. Subsequent classifier computations use the stored metric profiles.

A.2 Profile Preprocessing

To compare models with different native layer counts, we map the first and last layers to relative depths 0 and 1. The relative depth of native layer l and the coordinate of common grid point r are defined as follows.

$$\rho_{m,l} = \frac{l-1}{L_m - 1}, \qquad t_r = \frac{r-1}{G-1}, \qquad G = 64, \quad r = 1, \ldots, G.$$

For each prompt and metric, we construct the piecewise-linear interpolant through $(\rho_{m,l}, v_{m,p,k}(l))$ and denote its value at t_r by $u_{m,p,k}(r)$. The implementation uses numpy.interp. This represents every model's profiles on the same 64-point grid, with the first- and last-layer values at the grid endpoints. Each common grid index denotes a sample position along relative depth.

We then compute the mean and standard deviation along depth for each prompt and metric and apply z-score normalization.

$$\mu_{m,p,k} = \frac{1}{G}\sum_{r=1}^{G} u_{m,p,k}(r),$$

$$\sigma_{m,p,k} = \sqrt{\frac{1}{G}\sum_{r=1}^{G}\big(u_{m,p,k}(r) - \mu_{m,p,k}\big)^2},$$

$$z_{m,p,k}(r) = \frac{u_{m,p,k}(r) - \mu_{m,p,k}}{\sigma_{m,p,k} + \epsilon_z}, \qquad \epsilon_z = 10^{-12}.$$

The standard deviation uses G as the denominator, corresponding to ddof=0 in numpy.std. The constant ϵ_z stabilizes division when the standard deviation is very small. A profile that is constant across depth becomes a zero vector under this definition. Normalization is applied independently to each prompt and metric, placing the metrics on comparable scales while retaining their relative variation along depth. For unseen models, normalization likewise uses the mean and standard deviation of each prompt-level profile from that model.

This preprocessing supports profile comparisons across models in the classifier. The Pearson correlations used to measure cross-prompt consistency in Section 5.2 are computed on each model's native layer grid, as described in Appendix E.1.

A.3 Model and Family Representative Profiles

For each model and metric, we average the normalized profiles over prompts to obtain a model representative profile.

$$b_{m,k}(r) = \frac{1}{P_m}\sum_{p=1}^{P_m} z_{m,p,k}(r).$$

Thus, z-score normalization is performed before averaging, and every prompt receives equal weight. We combine the mean profiles of the three metrics at each depth and apply a common scaling factor to represent model m as follows.

$$Q_m(r) = \frac{1}{\sqrt{3G}}\begin{bmatrix} b_{m,1}(r) \\ b_{m,2}(r) \\ b_{m,3}(r) \end{bmatrix}, \qquad r = 1, \ldots, G.$$

Each $Q_m(r)$ is a three-component vector, and arranging these vectors in depth order gives a 64×3 sequence Q_m. Let $\mathcal{M}_f$ be the set of reference models in family f. The family representative profile is their arithmetic mean:

$$C_f(r) = \frac{1}{|\mathcal{M}_f|} \sum_{m \in \mathcal{M}_f} Q_m(r).$$

Each reference model within a family receives equal weight regardless of model size. Denoting the representative profile of unseen model q by $Q = Q_q$, the sequences Q and C_f are the two DTW inputs used in Section 4.2. Evaluation uses fixed family representatives C_f constructed from the 23 reference models.

The implementation applies a common factor of $1/\sqrt{3G} = 1/\sqrt{192}$ to each component, based on the total of $3G = 192$ components across three metrics and 64 grid points. This multiplies the squared Euclidean local costs and final DTW costs by $1/192$, preserving the ordering of family costs and the predicted family. The factor must be included to reproduce the DTW costs reported in the main text and appendices. In the implementation, it is applied before prompt averaging; linearity of the arithmetic mean yields the same representative profiles as the equations above.

For storage, the 64 values of each metric are concatenated in the order S_l, D_l, $\widetilde{D}_l$ to form a vector of length 192. For DTW, this vector is converted back into a 64×3 sequence so that all three metrics are compared jointly at each depth.

A.4 Dynamic Time Warping Matching

As discussed in Section 4.2, differences in native layer counts and interpolation onto a common grid may contribute to similar layerwise patterns appearing at neighboring grid points. We use DTW to accommodate these local positional differences (Sakoe & Chiba, 1978). The alignment preserves order along relative depth and accumulates profile differences along the path.

The local cost between grid point i of an unseen model and grid point j of family f is the squared Euclidean distance between their three metric components:

$$\delta_f(i,j) = \left\| Q(i) - C_f(j) \right\|_2^2, \qquad i,j = 1,\ldots,G.$$

The three metrics share a single alignment path for each query–family pair. For example, if the query's twentieth grid point is aligned with the family representative's twenty-second grid point, all three metrics contribute to the local cost at these same positions. A separate path is computed for each family compared with the query.

An admissible path π connects $(1,1)$ to (G,G), with consecutive positions separated by a step of $(1,1)$, $(1,0)$, or $(0,1)$. Every position on the path must satisfy the window condition

$$|i-j| \le w, \qquad w = 3.$$

Corresponding positions can therefore differ by at most three common grid points, or $3/63$ in relative depth. This condition limits the permitted depth deformation while preserving layer order. Let Π_3 denote the set of paths satisfying these conditions. The final DTW cost is

$$\mathcal{D}_{\mathrm{DTW}}(Q, C_f) = \min_{\pi \in \Pi_3} \sum_{(i,j) \in \pi} \delta_f(i,j).$$

Local costs are summed with equal weight over all path positions, including the start and end points. The accumulated cost is not divided by path length.

We compute this minimum accumulated cost using dynamic programming. Let R_f be the accumulated-cost array, initialized with $R_f(0,0) = 0$ and all other entries set to $+\infty$. For $i,j = 1,\ldots,G$ with $|i-j| \le 3$, entries are updated using the recurrence

$$R_f(i,j) = \delta_f(i,j) + \min\left\{R_f(i-1,j-1),\ R_f(i-1,j),\ R_f(i,j-1)\right\}.$$

The boundary entries $R_f(0,j)$ and $R_f(i,0)$, except the origin, and all entries outside the window remain $+\infty$. The final cost is $R_f(G,G)$. Recording the predecessor selected at each update allows the alignment path to be recovered by backtracking from the final position. The equations use one-based grid indices; the implementation stores the same path with indices from $(0,0)$ to $(63,63)$.

Throughout the main text and appendices, DTW cost refers to the minimum accumulated cost $\mathcal{D}_{\mathrm{DTW}}$ defined above. The implementation also stores its square root, $\sqrt{\mathcal{D}_{\mathrm{DTW}}}$, as a separate distance output. Both quantities give the same ordering of families; the paper reports the cost before taking the square root.

A.5 Classification Rule and Implementation Details

With the family set $\mathcal{F} = \{\text{Llama}, \text{Qwen}, \text{Gemma}, \text{SmolLM2}, \text{Pythia}, \text{DeepSeek}\}$, the classification rule in Section 4.2 is

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \mathcal{D}_{\mathrm{DTW}}(Q, C_f).$$

For each unseen model, we compute costs to all six families and predict the family with the lowest cost. If multiple families attain the minimum in the argmin above, all of them are reported. The query's ground-truth family label is not used in this procedure. The classifier performs closed-set classification over the six reference bank families, with no separate threshold for identifying an unknown family. DTW costs serve as comparative measures of profile differences across families.

Algorithm A1. Construction and prediction procedure for the MAGE model family classifier.

Input: Prompt-level profiles of the three metrics for 23 reference models with family labels, and profiles of the same three metrics for an unseen model.

Settings: Metric order $(S_l, D_l, \widetilde{D}_l)$, $G = 64$, $\epsilon_z = 10^{-12}$, and DTW window $w = 3$.

Output: The predicted family $\hat{f}$ of the unseen model (all minimum-cost families in a tie) and DTW costs to all six families.

1. Linearly interpolate each prompt-level metric profile of each reference model onto the common 64-point relative-depth grid.
2. Apply z-score normalization along depth to each interpolated profile, then construct Q_m by averaging over prompts and applying the common scaling factor.
3. Average the reference model profiles Q_m within each family with equal weights and store C_f.
4. Apply the preprocessing in steps 1–2 to the unseen model to construct Q.
5. For each $f \in \mathcal{F}$, compute $\mathcal{D}_{\mathrm{DTW}}(Q, C_f)$ using the recurrence in Appendix A.4.
6. Return the minimum-cost family $\hat{f}$ and the family costs. If the minimum is tied, report all tied families.

Preprocessing, representative profile construction, and DTW computations use float64. Input validation checks that the three metric matrices have matching prompt and layer counts, contain at least one prompt and two layers, and have finite entries. Finiteness is also checked during normalization and DTW computation, and the calculation is stopped if NaN or infinity occurs. For unseen model evaluation, we additionally verify that query and reference model IDs do not overlap.

When predecessor positions have exactly equal accumulated costs in dynamic programming, path recovery prioritizes the diagonal step $(1,1)$, followed by the query-only step $(1,0)$ and the family-representative-only step $(0,1)$. If multiple families have exactly the same minimum final cost, all are reported. Ties are determined from accumulated costs before rounding.

The stored classifier contains the preprocessing settings, DTW window, reference model IDs and family labels, model representative profiles, and family representative profiles. These settings and family representatives are retained when evaluating additional unseen models. Family costs and alignment paths are stored with each prediction to make the computation traceable. The evaluation protocol and model-level results are described in Appendix D.3 and Appendix F, respectively.

B Mathematical Details of Attention Energy Metrics

This appendix provides the algebraic derivations of the three attention energy metrics defined in Section 3.2, inspired by the Laplacian quadratic energy form in Section 2.3. Following the main text, let $d_{ij}^{(l)} = \bar{x}_i^{(l)} - \bar{x}_j^{(l)} \in \mathbb{R}^C$ be the token state difference and $M^{(l,h)} \in \mathbb{R}^{C\times C}$ the value–output projection for head h. The edge weight for information transfer $j \to i$ is $A_{ij}^{(l,h)} = \alpha_{ij}^{(l,h)} M^{(l,h)}$, where the attention probability satisfies $\alpha_{ij}^{(l,h)} \geq 0$.

The sums below range over $h = 1, \ldots, H$ and $i, j = 1, \ldots, T$, where H is the number of query heads in the layer and T is the number of tokens in the analyzed sequence. All three metrics use the factor $1/(HT)$, as in the main text. Terms excluded by the causal mask have $\alpha_{ij}^{(l,h)} = 0$, and self-loops have $d_{ii}^{(l)} = 0$.

B.1 Signed Attention Energy and the Symmetric Part

For a real matrix $A \in \mathbb{R}^{C\times C}$, define its symmetric part as $\mathrm{sym}(A) = (A + A^\top)/2$. For any vector $d \in \mathbb{R}^C$, the quantity $d^\top A d$ is a scalar and is therefore unchanged by transposition:

$$d^\top A d = (d^\top A d)^\top = d^\top A^\top d.$$

Averaging these two expressions gives

$$\begin{aligned} d^\top A d &= \frac{1}{2}\big(d^\top A d + d^\top A^\top d\big) \\ &= d^\top \frac{A + A^\top}{2} d = d^\top \mathrm{sym}(A) d. \end{aligned}$$

Here, $\mathrm{sym}\big(A_{ij}^{(l,h)}\big)$ is the symmetric part of the individual $C \times C$ matrix weight for a fixed token pair (i, j) and head h. Transposition acts on the feature coordinates of this matrix, leaving the edge indices (i, j) and the direction of information transfer $j \to i$ unchanged.

The identity holds for the real matrix determined by each input, so it also applies to $A_{ij}^{(l,h)}$, whose input dependence arises through the attention probability.

Substituting $A = A_{ij}^{(l,h)} = \alpha_{ij}^{(l,h)} M^{(l,h)}$ and $d = d_{ij}^{(l)}$ yields the following expansion of Signed Attention Energy:

$$\begin{aligned} S_l &= \frac{1}{HT} \sum_{h,i,j} \big(d_{ij}^{(l)}\big)^\top \mathrm{sym}\big(A_{ij}^{(l,h)}\big) d_{ij}^{(l)} \\ &= \frac{1}{HT} \sum_{h,i,j} \alpha_{ij}^{(l,h)} \big(d_{ij}^{(l)}\big)^\top M^{(l,h)} d_{ij}^{(l)}. \end{aligned}$$

Each term is an inner product between a token state difference and its value–output-projected counterpart. Depending on the relative directions of these vectors, the inner product can be positive, negative, or zero, so S_l can also have either sign. Taking the symmetric part preserves the quadratic form's value without restricting its sign.

B.2 Attention Energy and Its Squared-Norm Form

Attention Energy aggregates the squared magnitudes of value–output-projected token state differences, retaining attention probabilities as weights. The corresponding matrix weight is

$$B_{ij}^{(l,h)} = \alpha_{ij}^{(l,h)} \big(M^{(l,h)}\big)^\top M^{(l,h)}.$$

The matrix $\big(M^{(l,h)}\big)^\top M^{(l,h)}$ is symmetric, and for any $z \in \mathbb{R}^C$,

$$\begin{aligned} z^\top B_{ij}^{(l,h)} z &= \alpha_{ij}^{(l,h)} z^\top \big(M^{(l,h)}\big)^\top M^{(l,h)} z \\ &= \alpha_{ij}^{(l,h)} \big(M^{(l,h)} z\big)^\top \big(M^{(l,h)} z\big) \\ &= \alpha_{ij}^{(l,h)} \big\|M^{(l,h)} z\big\|_2^2 \geq 0 \end{aligned}$$

Hence, $B_{ij}^{(l,h)}$ is symmetric positive semidefinite. Substituting $z = d_{ij}^{(l)}$ and summing over token pairs and heads establishes the equivalence of the two expressions in Definition 2:

$$\begin{aligned} D_l &= \frac{1}{HT} \sum_{h,i,j} \big(d_{ij}^{(l)}\big)^\top B_{ij}^{(l,h)} d_{ij}^{(l)} \\ &= \frac{1}{HT} \sum_{h,i,j} \alpha_{ij}^{(l,h)} \big\|M^{(l,h)} d_{ij}^{(l)}\big\|_2^2 \geq 0. \end{aligned}$$

In this expansion, $\alpha_{ij}^{(l,h)}$ appears once outside the squared norm, retaining the attention weight assigned to each token pair. The transformation defined by $M^{(l,h)}$ appears inside the squared norm. Thus, D_l is the attention-probability-weighted squared magnitude of projected token state differences.

B.3 Normalized Attention Energy

Normalized Attention Energy applies Frobenius normalization to each head's value–output projection. With $\|M\|_F = \sqrt{\mathrm{tr}(M^\top M)}$, the normalization in Section 3.2 is

$$\widetilde{M}^{(l,h)} = \frac{M^{(l,h)}}{\|M^{(l,h)}\|_F + \epsilon}, \qquad \epsilon > 0.$$

The implementation uses $\epsilon = 10^{-12}$. Because the denominator is a positive scalar, the matrix in the resulting quadratic form can be written as

$$\big(\widetilde{M}^{(l,h)}\big)^\top \widetilde{M}^{(l,h)} = \frac{\big(M^{(l,h)}\big)^\top M^{(l,h)}}{\big(\|M^{(l,h)}\|_F + \epsilon\big)^2}$$

Definition 3 therefore becomes

$$\begin{aligned} \widetilde{D}_l &= \frac{1}{HT} \sum_{h,i,j} \alpha_{ij}^{(l,h)} \big(d_{ij}^{(l)}\big)^\top \big(\widetilde{M}^{(l,h)}\big)^\top \widetilde{M}^{(l,h)} d_{ij}^{(l)} \\ &= \frac{1}{HT} \sum_{h,i,j} \alpha_{ij}^{(l,h)} \frac{\big\|M^{(l,h)} d_{ij}^{(l)}\big\|_2^2}{\big(\|M^{(l,h)}\|_F + \epsilon\big)^2} \geq 0. \end{aligned}$$

Each head's contribution is divided by $\big(\|M^{(l,h)}\|_F + \epsilon\big)^2$ before summation. The token state differences and attention probabilities are the same as those used for D_l. This normalization reduces

the effect of the projection's overall magnitude on each head's contribution while reflecting how the normalized projection transforms token state differences along different directions.

The numerical stability term ϵ ensures that the projection's normalization denominator satisfies $\|M^{(l,h)}\|_F + \epsilon \geq \epsilon > 0$, so division remains defined even when the Frobenius norm is zero. If $M^{(l,h)} = 0$, both the normalized projection and the head's contribution are zero. When $\|M^{(l,h)}\|_F$ is sufficiently larger than ϵ, the relative effect of ϵ on the squared denominator $\left(\|M^{(l,h)}\|_F + \epsilon\right)^2$ becomes small. Both the equations above and the implementation retain ϵ.

C Metric Extraction and Implementation Details

This appendix describes how the three attention energy metrics in Sections 3.1–3.2 are extracted from a forward pass. Architecture-specific adapters map normalized token representations, attention probabilities, and headwise value–output projections to the common notation, and metric computation uses the factorization of the projections. The experiments in Section 5 share the same prompt-only protocol; model-specific computational settings and the experimental environment are described in Appendix D.2. Our code will be available at: https://github.com/piberius0421/mage_llm_fingerprint after acceptance.

C.1 Self-attention, Token Representations and Attention Probabilities

Standard attention notation. For standard multi-head attention, the bias-free query, key, and value projections of the normalized input are (Vaswani et al., 2017)

$$q_i^{(l,h)} = W_Q^{(l,h)} \bar{x}_i^{(l)}, \qquad k_j^{(l,h)} = W_K^{(l,h)} \bar{x}_j^{(l)}, \qquad v_j^{(l,h)} = W_V^{(l,h)} \bar{x}_j^{(l)}. \tag{10}$$

Queries and keys determine attention scores, while values carry the information to aggregate. In the standard scaled dot-product formulation with a causal mask, scores for future positions $j > i$ are set to $-\infty$ before softmax, giving

$$\alpha_{ij}^{(l,h)} = \begin{cases} \mathrm{softmax}_{j \leq i}\left(\dfrac{(q_i^{(l,h)})^\top k_j^{(l,h)}}{\sqrt{d_h}}\right), & j \leq i, \\ 0, & j > i. \end{cases} \tag{11}$$

These equations describe the standard formulation before architecture-specific operations such as positional encoding. In our implementation, the attention probabilities are captured from the model's actual forward pass, including its biases, positional encoding, and additional masks, as described below.

We extract attention information by passing each prompt to the model without adding a chat template or a generated response. The model is placed in evaluation mode, and a full-sequence forward pass is performed with gradient computation disabled. We use eager attention with output_attentions=True to obtain attention probabilities and do not use a key–value cache. Under this prompt-only protocol, the replay function processes the input prompt's token sequence directly.

We obtain the normalized token representation $\bar{x}_i^{(l)}$ using a forward pre-hook to capture the hidden state entering the attention module of each layer. This capture point follows the normalization preceding attention and precedes the query, key, and value projections. The metric therefore uses the state that the model actually supplies to attention. The captured tensor is stored as a $T \times C$ array with tokens as rows.

For the attention probability $\alpha_{ij}^{(l,h)}$, we use the post-softmax tensor returned by the same forward pass. Removing the batch dimension gives an $H \times T \times T$ array, whose last two dimensions index query token i and key token j, respectively. Attention dropout is disabled in evaluation mode, so these probabilities are used in value aggregation.

The returned probabilities already reflect the model's causal mask and any layer-specific local or sliding-window mask. Metric computation additionally multiplies entries with $j > i$ by zero to preserve causality and uses the remaining probabilities without further normalization. Entries excluded by a local mask retain their returned values. Model-specific operations such as rotary positional

encoding, query–key normalization, and attention logit soft-capping are applied during the forward pass, so their effects are reflected in the captured probabilities.

We process one prompt at a time and add no padding. Metric computation uses all input tokens, including special tokens added by the tokenizer's default settings, and their number is denoted by T. Causal and local masks are applied within the model's attention computation.

We retain the biases in the model's projections during the forward pass. Query and key biases therefore affect the extracted attention probabilities. Within a head, the same value bias b_V is added to every token, so it cancels when taking a token difference: $(W_V\bar{x}_i + b_V) - (W_V\bar{x}_j + b_V) = W_V(\bar{x}_i - \bar{x}_j)$. The metric's value–output transformation applies the linear component $W_O W_V$ defined in Section 3.1 to this difference. The model's output bias is added after headwise value aggregation and output projection. The reconstruction check in Appendix C.3 includes both value and output biases when reconstructing the actual attention output.

C.2 Value–Output Projections and Head Mapping

Let H denote the number of query heads, H_{KV} the number of key–value heads, and d_h the head dimension. For models that store value and output projections as separate weights, their dimensions are

$$W_V^{(l)} \in \mathbb{R}^{H_{\mathrm{KV}} d_h \times C}, \qquad W_O^{(l)} \in \mathbb{R}^{C \times H d_h}.$$

To construct the headwise projections, we extract the d_h consecutive columns of the output projection corresponding to query head h, obtaining $W_O^{(l,h)} \in \mathbb{R}^{C \times d_h}$. From the value projection, we extract the d_h consecutive rows for the corresponding key–value head, obtaining $W_V^{(l,g(h))} \in \mathbb{R}^{d_h \times C}$. The head dimension is taken from the model configuration when specified and otherwise computed as C/H. The adapter verifies that the full output and value projection dimensions match $C \times H d_h$ and $H_{\mathrm{KV}} d_h \times C$, respectively.

In grouped-query attention, multiple query heads share a key–value head (Ainslie et al., 2023). In the analyzed models, the query heads associated with each key–value head form equally sized contiguous groups. The adapter checks that H is divisible by H_{KV} and sets the number of query heads per group to $r = H/H_{\mathrm{KV}}$. Using one-based head indices as in the main text, the mapping is

$$g(h) = 1 + \left\lfloor \frac{h-1}{r} \right\rfloor, \qquad M^{(l,h)} = W_O^{(l,h)} W_V^{(l,g(h))}$$

For standard multi-head attention, $H_{\mathrm{KV}} = H$ and hence $g(h) = h$. Each query head's contribution uses its output block and attention probabilities together with the corresponding value block. Aggregation and averaging also use the query head count H.

Table 4 summarizes the projection structure and capture points for each architecture. The adapter is selected according to the checkpoint's architecture configuration.

Table 4: Architecture-specific capture points and projection handling for the analyzed models. Normalized states are captured at the input to the attention module listed in each row.

Architecture and applicable models	**Attention module**	**Value and output block construction and handling**
Llama-based architectures: Llama, SmolLM2, native DeepSeek-LLM, DeepSeek-R1-Distill-Llama	self_attn	Extract row blocks from v_proj and column blocks from o_proj, with key–value head mapping determined by the configuration.

Continued on next page

Table 4 (continued)

Architecture and applicable models	Attention module	Value and output block construction and handling
Qwen2-based architectures: Qwen2.5, DeepSeek-R1-Distill-Qwen	self_attn	Extract blocks from v_proj and o_proj. Include value bias in reconstruction when present.
Gemma 2	self_attn	Use v_proj and o_proj, capturing probabilities after layer-specific local/global attention masking and logit soft-capping.
Gemma 3	self_attn	Use v_proj and o_proj while retaining query–key normalization and local/global attention settings. For multimodal checkpoints, analyze only decoder layers in the text backbone.
GPT-NeoX architecture: Pythia	attention	Extract value blocks from the headwise combined query_key_value array and use column blocks from dense as the output projection.

Pythia stores query, key, and value weights jointly within each head. We therefore interpret the combined weight as an $H \times 3 \times d_h \times C$ array and select the value component for each head. Biases are extracted in the same order.

For Gemma 3 multimodal checkpoints, we provide text prompts only. Metric computation uses the text configuration and language-model backbone, excluding vision tower parameters.

C.3 Efficient Computation and Numerical Validation

Factorized metric computation. We compute the three metrics by applying the value and output projections in sequence. For a single layer l, write $x_i = \bar{x}_i^{(l)}$, $O_h = W_O^{(l,h)}$, $V_h = W_V^{(l,g(h))}$, and $\alpha_{ij}^h = \alpha_{ij}^{(l,h)}$. The token representation after both projections is

$$y_i^h = O_h(V_h x_i) = M^{(l,h)} x_i.$$

The intermediate vector $V_h x_i$ has dimension d_h. Applying the projections in sequence gives the required values without explicitly constructing the $C \times C$ matrix $M^{(l,h)}$ or the individual edge matrices $A_{ij}^{(l,h)}, B_{ij}^{(l,h)}$. The implementation stacks token representations as rows in $X_{\mathrm{row}} = [x_1, \ldots, x_T]^\top \in \mathbb{R}^{T \times C}$. The successive matrix products $X_{\mathrm{row}} V_h^\top$ and $(X_{\mathrm{row}} V_h^\top) O_h^\top$ compute the projections for all tokens.

Since $y_i^h - y_j^h = M^{(l,h)} d_{ij}^{(l)}$, the signed term and squared-distance term for each token pair can be written as

$$\begin{aligned} s_{ij}^h &= (x_i - x_j)^\top (y_i^h - y_j^h) \\ &= x_i^\top y_i^h + x_j^\top y_j^h - x_i^\top y_j^h - x_j^\top y_i^h, \\ q_{ij}^h &= \|y_i^h - y_j^h\|_2^2 \\ &= \|y_i^h\|_2^2 + \|y_j^h\|_2^2 - 2(y_i^h)^\top y_j^h. \end{aligned}$$

The implementation computes the inner products in these expansions using matrix multiplication and combines the required diagonal entries to obtain the signed and squared-distance terms as $T \times T$ arrays. This avoids storing a C-dimensional difference vector for every token pair. The projection's Frobenius norm is obtained from two Gram matrices:

$$\|O_h V_h\|_F^2 = \mathrm{tr}\big((O_h^\top O_h)(V_h V_h^\top)\big).$$

Both Gram matrices are symmetric and have dimension $d_h \times d_h$. The trace can therefore be computed by summing their elementwise products, and its square root gives the projection's Frobenius norm. Writing $f_h = \|O_h V_h\|_F$, the three metrics are computed as

$$S_l = \frac{1}{HT} \sum_{h,i,j} \alpha_{ij}^h s_{ij}^h,$$
$$D_l = \frac{1}{HT} \sum_{h,i,j} \alpha_{ij}^h q_{ij}^h,$$
$$\widetilde{D}_l = \frac{1}{HT} \sum_{h,i,j} \frac{\alpha_{ij}^h q_{ij}^h}{(f_h + \epsilon)^2}, \qquad \epsilon = 10^{-12}.$$

These expressions evaluate the definitions in Appendix B in factorized form. For each query token, contributions are weighted once by the corresponding attention probabilities and summed over key tokens, then averaged over the H query heads and T query tokens. Causal and local masks are reflected in the attention probabilities, and the averaging denominator is HT.

To handle floating-point roundoff, we apply $\max(\cdot, 0)$ to the computed squared distances and squared Frobenius norms. Signed terms retain their computed signs. The token-pair quantities stored in this calculation are $T \times T$ arrays of attention probabilities, signed terms, and squared-distance terms.

Numerical precision. Metric computation uses the captured representations and attention probabilities and the extracted projection blocks converted to float32. Headwise accumulation also uses float32, while classifier preprocessing and DTW in Appendix A use float64.

During the forward pass, all evaluated models retain float16 weights, while attention scores, softmax probabilities, and value aggregation are computed in float32. Under the default setting, normalized sublayer inputs and the returned attention aggregation values use float16, while the residual stream remains in float32.

For Gemma-3-27B-PT and Gemma-3-27B-IT, we use a full-block stabilization setting that retains normalization outputs, linear projection outputs, and intermediate feed-forward activations in float32. If the maximum absolute value of a tensor entering a linear operation exceeds 32768, the input is rescaled by a power of two before conversion to float16. The result is converted to float32 and restored to its original scale. Model-specific settings and the execution environment are described in Appendix D.2.

Attention-output reconstruction. To verify that the capture points and headwise projection decomposition are consistent with the actual attention computation, we record the attention module's output using a forward hook and compare it with a reconstruction. The hook captures the attention output after output projection and before residual addition and any subsequent normalization. With value bias $b_V^{(l,g(h))}$ and output bias $b_O^{(l)}$, the attention output is reconstructed as follows. Bias terms are set to zero when absent.

$$\widehat{o}_i^{(l)} = \sum_{h=1}^{H} W_O^{(l,h)} \left[\sum_{j=1}^{T} \alpha_{ij}^{(l,h)} \left(W_V^{(l,g(h))} \bar{x}_j^{(l)} + b_V^{(l,g(h))} \right) \right] + b_O^{(l)}.$$

Let $Z^{(l)}$ and $\widehat{Z}^{(l)}$ be the arrays collecting the actual and reconstructed outputs over tokens, respectively. We measure their difference using the layerwise relative Frobenius error:

$$e_l = \frac{\|Z^{(l)} - \widehat{Z}^{(l)}\|_F}{\|Z^{(l)}\|_F + 10^{-12}}.$$

Reconstruction uses the captured states and probabilities together with the headwise projection blocks, with computations performed in float32. We use a tolerance of 0.02 to accommodate dif-

ferences in computational precision and accumulation order between the forward pass and reconstruction, checking that e_l is finite and does not exceed this threshold. Maximum and mean absolute errors are also recorded, and the default execution stops the analysis if validation fails. The current validation function computes differences between float32 outputs and accumulates the Frobenius norms in float64 to avoid overflow during norm calculation. This check verifies that the value–output computation using the normalized states, attention probabilities, head mapping, and biases reconstructs the actual attention output.

We reviewed the existing validation records for the 23 reference models, 30 ordinary unseen models, and four separately evaluated distilled models used in the paper. All 1,874,000 prompt–layer pairs passed the criterion above. The maximum relative Frobenius error was approximately 1.338×10^{-3}, observed for Pythia-12B. These values measure discrepancies between actual and reconstructed attention outputs.

C.4 Computational Cost and Memory Usage

Measurement scope. We measured MAGE extraction for Llama-3.2-3B, a 28-layer reference model, on the same 1,000 prompts used in the main experiments. Each prompt was processed individually on a single NVIDIA A100-SXM4-80GB GPU, with float16 weights and float32 attention and metric computations. Input lengths, including tokenizer-added special tokens, ranged from 14 to 160 tokens, with a median of 31. The prompt-only protocol used neither chat templates nor response generation. Three warm-up passes were excluded, after which all 1,000 prompts were measured in a single session.

We timed input tokenization and preparation, the full-sequence forward pass with capture, computation of all three MAGE metrics across all layers, and attention-output reconstruction separately, synchronizing CUDA before and after each stage. The forward stage includes capture hooks and transfer of captured tensors to CPU; the metric stage includes their transfer back to GPU, projection-block preparation, and metric computation. Thus, these are implementation-level stage times rather than isolated GPU-kernel times. No scalar baseline metric was computed. Model loading (3.57 s), warm-up (4.32 s), and result serialization are excluded from Table 5.

Table 5: Stage-wise runtime and GPU memory for Llama-3.2-3B over 1,000 prompts. Total is the sum of measured stage times across prompts; median and 95th percentile are per-prompt statistics. Combined-row statistics are computed from per-prompt sums, not sums of percentiles. Peak memory is the maximum PyTorch allocated GPU memory during the listed stage(s), including resident model weights but excluding allocator-reserved memory and other processes. Stage peaks are not additive. $1\,\mathrm{GiB} = 2^{30}$ bytes.

Stage	Total (s)	Median (s)	95th pct. (s)	Peak (GiB)
Input preparation	1.03	0.001	0.001	5.992
Forward pass and capture	62.24	0.056	0.079	6.172
Three-metric computation	359.61	0.359	0.369	6.047
Reconstruction check	188.61	0.180	0.254	6.051
Extraction without reconstruction	422.88	0.416	0.446	6.172
Extraction with reconstruction	611.49	0.597	0.700	6.172

The extraction stages excluding reconstruction took 422.88 s (7.05 min) in total, or a median of 0.416 s per prompt. Including reconstruction increased the measured total to 611.49 s (10.19 min). These totals sum the synchronized stages rather than the entire job's wall-clock duration. All 28,000 prompt–layer reconstruction checks passed the 0.02 tolerance in Appendix C.3, with a maximum relative Frobenius error of 4.990×10^{-4}.

Interpretation and limitations. These measurements characterize one reference model over the evaluated 14–160-token inputs, not a controlled input-length scaling study or a long-context benchmark. The reported memory is GPU allocation, not total host and device memory.

DTW classification runs on the CPU after extraction and is excluded from these measurements. It compares a model representative on a 64-point grid with six family representatives, rather than operating on the input token sequence. Its computational cost is therefore distinct from the token-dependent extraction cost. A separate DTW wall-clock time was not measured.

D Prompts and Experimental Protocol

This appendix describes the prompt set, metric extraction environment, and evaluation procedure used in Sections 4.1 and 5.1. Profile analysis and both classification experiments use the same prompt set. We describe the reference models and evaluation models separately.

D.1 Prompt Set Composition

We constructed 1,000 distinct prompts by combining task-specific templates with topics, sentences, or situational conditions. The set covers 12 categories, including technical explanation, summarization, translation, coding, and reasoning. Table 6 summarizes the prompt count and task types in each category. For example, technical explanation prompts request explanations of scientific or engineering concepts, while long-context reasoning prompts ask for judgments under multiple constraints and incomplete evidence.

Table 6: Category composition and task types of the 1,000-prompt set.

Category	**Prompts**	**Tasks**
Technical explanation	100	Explain scientific, mathematical, and engineering concepts and mechanisms
Humanities and society	100	Explain causes, effects, and perspectives on historical and social phenomena
Factual and everyday knowledge	80	Answer questions about everyday phenomena and basic knowledge
Summarization	80	Summarize the main claims and supporting evidence in a passage
Text transformation	70	Rewrite text or change its style while preserving meaning
Translation	100	Translate sentences between languages
Coding	100	Explain implementation approaches, errors, and boundary conditions
Logic and mathematical reasoning	100	Reason about arithmetic problems and logical conditions
Creative writing	80	Write short passages subject to specified topics and constraints
Safety and refusal	80	Respond safely to harmful requests and suggest alternatives
Noisy and ambiguous input	60	Interpret questions containing typographical errors or ambiguous wording
Long-context reasoning	50	Make judgments and decisions under multiple situational constraints
Total	1,000	

Table 7 reports the language and length composition using the metadata labels assigned during prompt construction. For translation prompts, a label may indicate the language of the sentence to be translated, while the surrounding instruction is in English. For example, a prompt labeled ja contains a Japanese sentence and an English translation instruction. The labels mixed and noisy denote separately assigned input types in the metadata.

Table 7: Counts by language and length_class labels in the prompt metadata. Each classification covers the same 1,000 prompts.

Metadata field	Label	Prompts
language	en (English)	785
language	ko (Korean)	115
language	es (Spanish)	10
language	fr (French)	10
language	ja (Japanese)	10
language	zh (Chinese)	10
language	mixed	6
language	noisy	54
length_class	short	100
length_class	medium	850
length_class	long	50

The labels short, medium, and long were assigned during prompt construction. Actual character counts, including spaces, range from 59 to 523, with a median of 145.5. Across all experiment records, input lengths after applying each model's tokenizer and default special tokens range from 13 to 574 tokens. Thus, the token count T can vary across models for the same prompt.

For example, prompt 1 is "Explain photosynthesis to a first-year university student and give one concrete example. Answer in one paragraph." Prompt 54 is "Translate this ja sentence accurately into English: "会議の目的を最初に確認してください。" Answer in one paragraph." and requests a Japanese-to-English translation. Each prompt includes an instruction to answer in one paragraph. The experiments use a prompt-only protocol that takes the entire prompt, including this instruction, as input (Appendix D.2).

D.2 Extraction Settings and Computational Environment

All experiments follow the prompt-only protocol in Appendix C.1. Prompts are processed individually without an added chat template or separate system instruction, and metrics are extracted from the full sequence, including special tokens added by the tokenizer's default settings. The model is placed in evaluation mode, and a full-sequence forward pass uses eager attention with output_attentions=True and gradient computation disabled. No response is generated, and no key–value cache is used.

All model weights are retained in float16, while attention scores, softmax probabilities, value aggregation, and metric computations use float32. The default fp16_weights_fp32_attention setting is used for all 23 reference models, 28 ordinary unseen models, and four distilled models. Gemma-3-27B-PT and Gemma-3-27B-IT use fp16_weights_fp32_full_block, which also retains normalization outputs, linear projection outputs, and intermediate feed-forward activations in float32. Their weights remain in float16. Appendix C.3 describes the rescaling used for full-block stabilization and the precision of the individual operations. Profile preprocessing and DTW computations follow Appendix A and use float64.

D.3 Evaluation Protocol

Profile analysis uses the 23 reference models in Appendix A.1. For each model and metric, cross-prompt consistency is evaluated using Pearson correlations over the 499,500 distinct pairs formed by 1,000 prompts, computed on the native layer grid before interpolation. For comparisons across models and families, each prompt's profiles are interpolated onto a 64-point relative-depth grid, z-score-normalized along depth, and averaged over prompts. Each family representative is the equally weighted average of its reference model representative profiles. In the analysis of reference bank structure, each model is included in the construction of its own family representative. Appendix E presents the full correlation statistics, additional profiles, and heatmaps.

Family classification is evaluated on 30 ordinary unseen models outside the reference bank: four Llama, ten Qwen, ten Gemma, two SmolLM2, and four Pythia models. Table 8 lists all evaluated models.

Table 8: The 30 ordinary unseen models used for model family classification.

Family	**Model checkpoint**
Llama	meta-llama/Meta-Llama-3-8B
Llama	meta-llama/Meta-Llama-3-8B-Instruct
Llama	meta-llama/Llama-3.1-8B
Llama	meta-llama/Llama-3.1-8B-Instruct
Qwen	Qwen/Qwen2.5-3B
Qwen	Qwen/Qwen2.5-3B-Instruct
Qwen	Qwen/Qwen2.5-7B
Qwen	Qwen/Qwen2.5-7B-Instruct
Qwen	Qwen/Qwen2.5-Math-7B
Qwen	Qwen/Qwen2.5-Math-7B-Instruct
Qwen	Qwen/Qwen2.5-14B
Qwen	Qwen/Qwen2.5-14B-Instruct
Qwen	Qwen/Qwen2.5-32B
Qwen	Qwen/Qwen2.5-32B-Instruct
Gemma	google/gemma-2-9b
Gemma	google/gemma-2-9b-it
Gemma	google/gemma-2-27b
Gemma	google/gemma-2-27b-it
Gemma	google/gemma-3-4b-pt
Gemma	google/gemma-3-4b-it
Gemma	google/gemma-3-12b-pt
Gemma	google/gemma-3-12b-it
Gemma	google/gemma-3-27b-pt
Gemma	google/gemma-3-27b-it
SmolLM2	HuggingFaceTB/SmolLM2-135M
SmolLM2	HuggingFaceTB/SmolLM2-135M-Instruct
Pythia	EleutherAI/pythia-70m
Pythia	EleutherAI/pythia-160m
Pythia	EleutherAI/pythia-6.9b
Pythia	EleutherAI/pythia-12b

During evaluation, the six family representatives constructed from the 23 reference models remain fixed. The same preprocessing is applied to each unseen model to obtain a single model representative profile. Costs to all families are computed using the three metrics, a 64-point grid, and a DTW window of 3, and the minimum-cost family is predicted. There were no ties at the minimum cost in this evaluation, so each model received a single family prediction. The general tie-handling rule is given in Appendix A.5.

The four DeepSeek distilled models are evaluated separately for base-lineage attribution using the same classifier. Table 2 in Section 5.6 lists these models and their publicly documented base families, against which the predictions are compared. The unit of evaluation in both experiments is a model checkpoint, and accuracy is reported as the number of correct models divided by the number of evaluated models. The denominators are 30 ordinary unseen models for family classification and four distilled models for base-lineage attribution. The reference models are listed in Table 3 of Appendix A.1, while ground-truth labels, predictions, and sources for the publicly documented base lineages are provided in Appendices F.1–F.3.

E Additional Profile Consistency and Structure Results

E.1 Native-Layer Pearson Correlations

Table 9 extends the analysis in Section 5.2 to all 23 reference models. For each model and each of the three attention energy metrics, we compute Pearson correlations between profiles from distinct prompts on the native layer grid, without interpolation or DTW alignment. All 499,500 pairs formed by the 1,000 prompts have valid correlations for every model and metric.

Table 9: Native-layer prompt-pair Pearson correlations for all reference models. Each metric column reports the median [5th percentile, 95th percentile], and the valid pair count is per metric. Values are rounded to four decimal places.

Model (native layers)	S_l	D_l	$\widetilde{D}_l$	**Valid pairs / metric**
Llama-3.2-1B (16)	0.9836 [0.9528, 0.9953]	0.9990 [0.9952, 0.9998]	0.9764 [0.8568, 0.9947]	499,500
Llama-3.2-1B-Instruct (16)	0.9821 [0.9373, 0.9950]	0.9991 [0.9918, 0.9999]	0.9778 [0.8912, 0.9951]	499,500
Llama-3.2-3B (28)	0.9806 [0.8983, 0.9951]	0.9968 [0.9864, 0.9994]	0.9843 [0.9386, 0.9964]	499,500
Llama-3.2-3B-Instruct (28)	0.9804 [0.8476, 0.9945]	0.9976 [0.9863, 0.9994]	0.9856 [0.9138, 0.9960]	499,500
Qwen2.5-0.5B (24)	0.9559 [0.5375, 0.9880]	0.9960 [0.9810, 0.9994]	0.9877 [0.9560, 0.9972]	499,500
Qwen2.5-0.5B-Instruct (24)	0.9591 [0.6148, 0.9888]	0.9970 [0.9871, 0.9995]	0.9908 [0.9704, 0.9978]	499,500
Qwen2.5-1.5B (28)	0.9814 [0.9097, 0.9950]	0.9988 [0.9864, 0.9998]	0.9955 [0.9723, 0.9991]	499,500
Qwen2.5-1.5B-Instruct (28)	0.9812 [0.9174, 0.9947]	0.9989 [0.9928, 0.9998]	0.9959 [0.9818, 0.9992]	499,500
Qwen2.5-Math-1.5B (28)	0.9657 [0.1543, 0.9940]	0.9982 [0.9883, 0.9997]	0.9957 [0.9826, 0.9990]	499,500
gemma-3-1b-pt (26)	0.9997 [0.9967, 0.9999]	0.9814 [0.8670, 0.9992]	0.9964 [0.9682, 0.9999]	499,500
gemma-3-1b-it (26)	0.9997 [0.9978, 0.9999]	0.9946 [0.9741, 0.9995]	0.9957 [0.9747, 0.9996]	499,500
gemma-2-2b (26)	0.9627 [0.8419, 0.9901]	0.9765 [0.9012, 0.9946]	0.9739 [0.8905, 0.9941]	499,500
gemma-2-2b-it (26)	0.9707 [0.9000, 0.9915]	0.9852 [0.9126, 0.9962]	0.9839 [0.9009, 0.9959]	499,500
SmolLM2-360M (32)	0.9781 [0.8622, 0.9946]	0.9970 [0.9773, 0.9994]	0.9860 [0.8252, 0.9960]	499,500
SmolLM2-360M-Instruct (32)	0.9826 [0.8841, 0.9956]	0.9973 [0.9831, 0.9994]	0.9877 [0.8654, 0.9963]	499,500
SmolLM2-1.7B (24)	0.9596 [0.7791, 0.9950]	0.9974 [0.9781, 0.9996]	0.9876 [0.8963, 0.9972]	499,500
SmolLM2-1.7B-Instruct (24)	0.9728 [0.8678, 0.9960]	0.9975 [0.9814, 0.9997]	0.9894 [0.9157, 0.9978]	499,500
pythia-410m (24)	0.9429 [0.4140, 0.9930]	0.9645 [0.6429, 0.9968]	0.9327 [0.1521, 0.9908]	499,500
pythia-1b (16)	0.9858 [0.8265, 0.9971]	0.9951 [0.9467, 0.9988]	0.9645 [0.4245, 0.9913]	499,500
pythia-1.4b (24)	0.9820 [0.8595, 0.9937]	0.9947 [0.8510, 0.9985]	0.9908 [0.7887, 0.9972]	499,500
pythia-2.8b (32)	0.9492 [0.8354, 0.9826]	0.9958 [0.8990, 0.9987]	0.9916 [0.7955, 0.9970]	499,500

Continued on next page

Table 9 (continued)

Model (native layers)	S_l	D_l	$\widetilde{D}_l$	**Valid pairs / metric**
deepseek-llm-7b-base (30)	0.9559 [0.7167, 0.9908]	0.9976 [0.9823, 0.9995]	0.9912 [0.9283, 0.9979]	499,500
deepseek-llm-7b-chat (30)	0.9637 [0.7703, 0.9920]	0.9973 [0.9829, 0.9995]	0.9901 [0.9304, 0.9978]	499,500

E.2 Example Profiles across Families

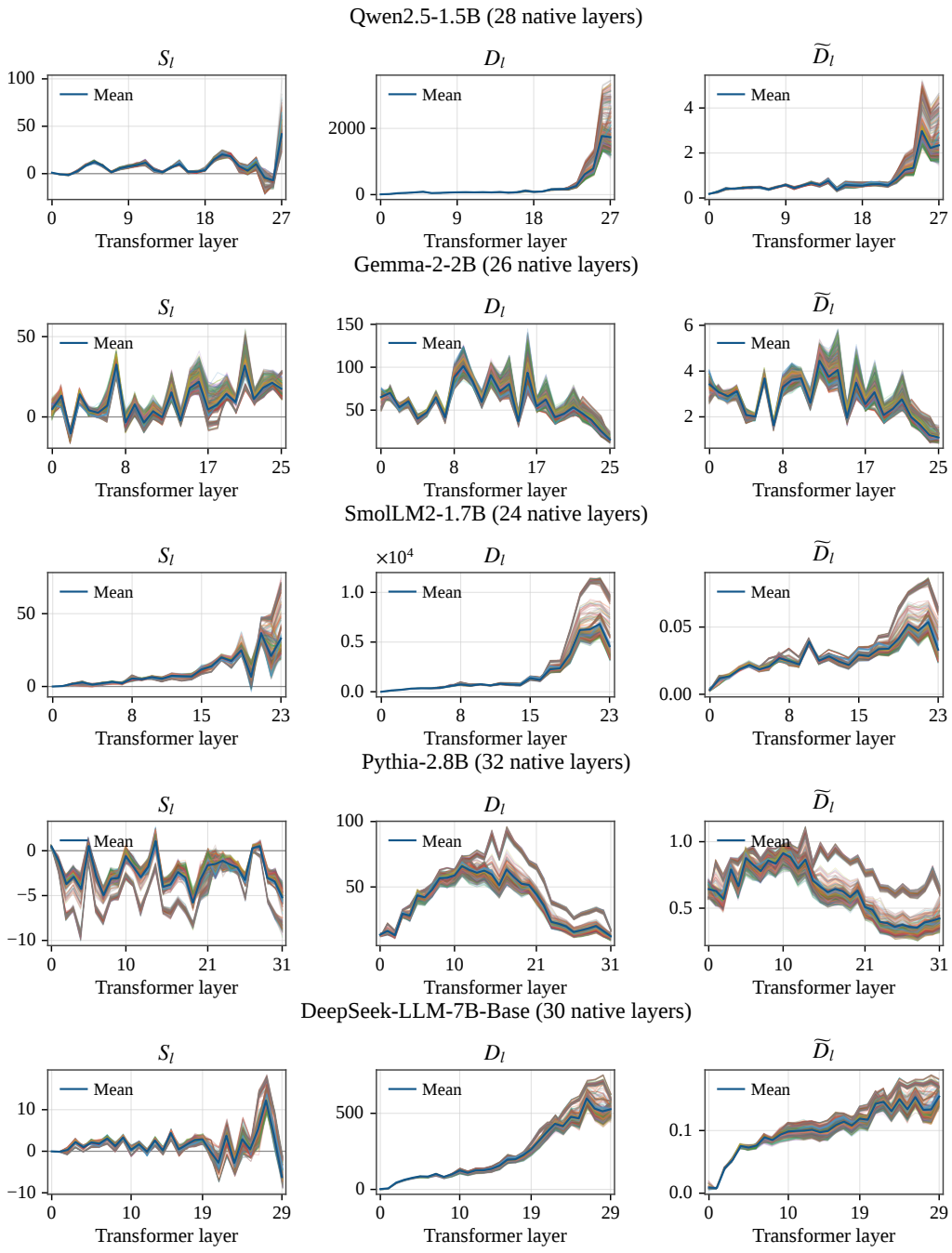


Figure 3: Layerwise profiles of the attention energy metrics across 1,000 prompts for the five example models from Table 1. From top to bottom: Qwen2.5-1.5B, Gemma-2-2B, SmolLM2-1.7B, Pythia-2.8B, and DeepSeek-LLM-7B-Base, with 28, 26, 24, 32, and 30 native layers, respectively. Columns show S_l, D_l, and $\widetilde{D}_l$ from left to right. Thin curves show individual prompts; thick blue curves show their mean at each layer.

E.3 Between-Model and Within-Family Structure

Figures 4–5 show the between-model comparisons and family-level structure examined in Section 5.3. Both use costs computed with the three metrics, a 64-point grid, prompt-level z-score normalization followed by averaging, and a DTW window of 3, as described in Appendix A.

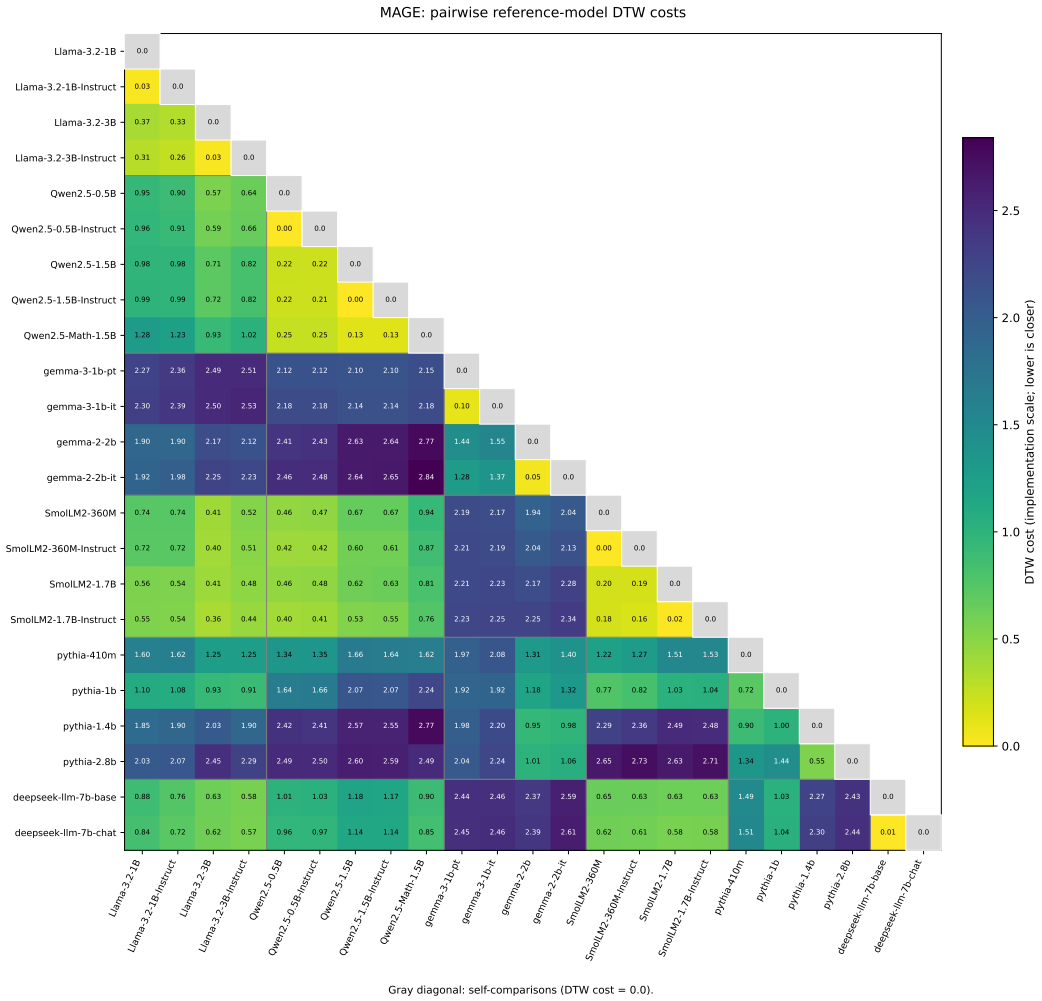


Figure 4: Pairwise DTW costs between the model representative profiles of 23 reference models. Models on both axes follow the same family ordering, and the lower triangle of the symmetric cost matrix is shown. The gray diagonal entries of 0.0 indicate self-comparisons. Lower costs indicate greater similarity between preprocessed profiles. Off-diagonal values are rounded to two decimal places.

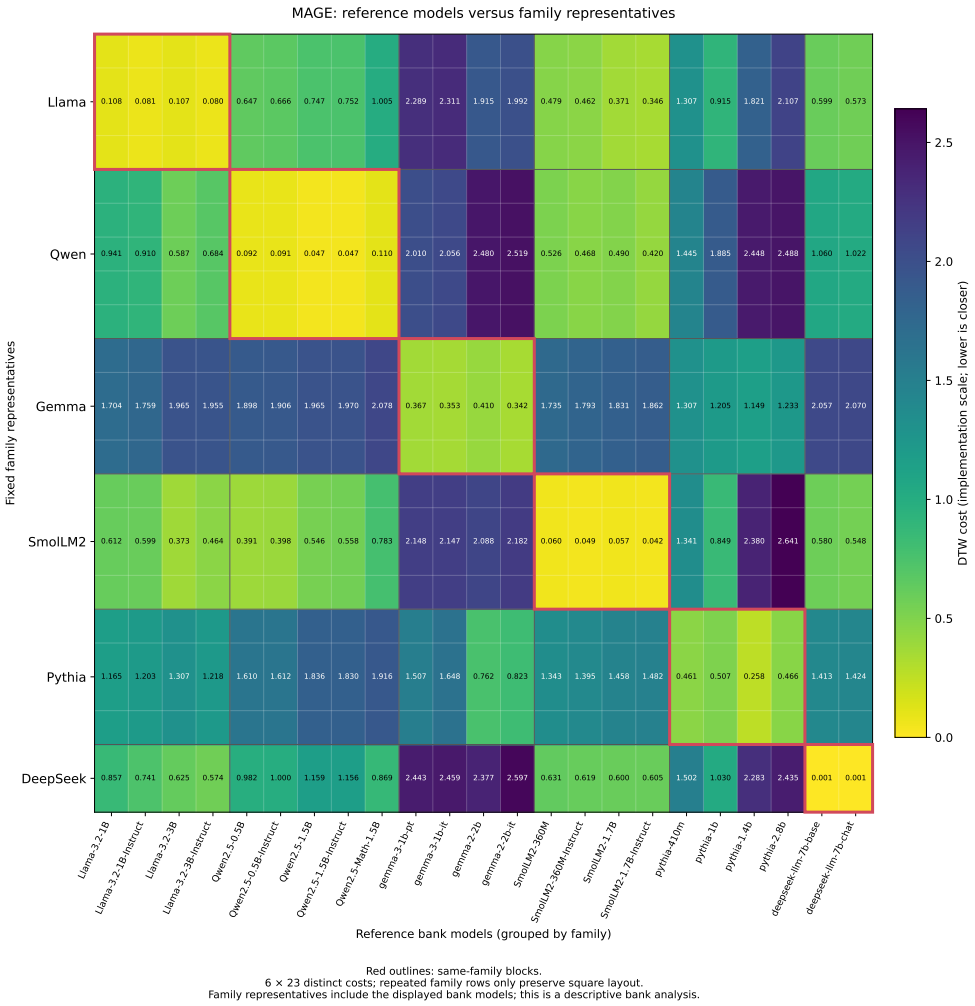


Figure 5: DTW costs between six family representatives and the model representative profiles of 23 reference models. The horizontal axis lists models grouped by family, and the vertical axis lists family representatives. Red outlines highlight the six same-family blocks. Lower costs indicate greater profile similarity. Values are rounded to three decimal places.

F Model Family Classification, LoRA-SFT Robustness, and Base-Lineage Attribution Results

F.1 Unseen Model Family Classification

Table 10 summarizes the ground-truth families and MAGE predictions for the 30 ordinary unseen models evaluated in Section 5.5. We use the same reference bank and classifier described in Appendix A, predicting one family per model representative profile constructed from 1,000 prompts.

Table 10: Model family classification results for the 30 ordinary unseen models. Model names retain the original checkpoint names with the repository organization omitted, and misclassifications are explicitly marked in the outcome column. Overall accuracy is 25/30.

Model	**Ground-truth family**	**MAGE prediction**	**Outcome**
Llama-3.1-8B	Llama	Llama	Correct
Llama-3.1-8B-Instruct	Llama	Llama	Correct
Meta-Llama-3-8B	Llama	Llama	Correct
Meta-Llama-3-8B-Instruct	Llama	Llama	Correct
Qwen2.5-3B	Qwen	Qwen	Correct
Qwen2.5-3B-Instruct	Qwen	Qwen	Correct
Qwen2.5-7B	Qwen	Qwen	Correct
Qwen2.5-7B-Instruct	Qwen	Qwen	Correct
Qwen2.5-Math-7B	Qwen	Qwen	Correct
Qwen2.5-Math-7B-Instruct	Qwen	Qwen	Correct
Qwen2.5-14B	Qwen	DeepSeek	Incorrect
Qwen2.5-14B-Instruct	Qwen	DeepSeek	Incorrect
Qwen2.5-32B	Qwen	Qwen	Correct
Qwen2.5-32B-Instruct	Qwen	Qwen	Correct
gemma-2-9b	Gemma	Gemma	Correct
gemma-2-9b-it	Gemma	Gemma	Correct
gemma-3-4b-it	Gemma	Gemma	Correct
gemma-3-4b-pt	Gemma	Gemma	Correct
gemma-2-27b	Gemma	Pythia	Incorrect
gemma-2-27b-it	Gemma	Gemma	Correct
gemma-3-12b-it	Gemma	Gemma	Correct
gemma-3-12b-pt	Gemma	Gemma	Correct
gemma-3-27b-it	Gemma	Gemma	Correct
gemma-3-27b-pt	Gemma	Gemma	Correct
SmolLM2-135M	SmolLM2	SmolLM2	Correct
SmolLM2-135M-Instruct	SmolLM2	SmolLM2	Correct
pythia-160m	Pythia	Pythia	Correct
pythia-6.9b	Pythia	Pythia	Correct
pythia-70m	Pythia	Gemma	Incorrect
pythia-12b	Pythia	Qwen	Incorrect

Table 11: DTW costs and ranks of the predicted and ground-truth families for misclassified ordinary unseen models. Each comparison column gives the family, cost, and rank. Ranks are determined by ascending unrounded cost among the six families. The cost gap is the ground-truth family cost minus the predicted family cost. Costs and gaps are rounded to four decimal places.

Model	**Predicted family: cost (rank)**	**Ground-truth family: cost (rank)**	**Cost gap**
Qwen2.5-14B	DeepSeek: 0.4952 (1)	Qwen: 0.7767 (2)	0.2815
Qwen2.5-14B-Instruct	DeepSeek: 0.4829 (1)	Qwen: 0.7301 (2)	0.2471
gemma-2-27b	Pythia: 1.3524 (1)	Gemma: 1.3956 (2)	0.0432
pythia-70m	Gemma: 0.6692 (1)	Pythia: 1.0696 (2)	0.4004
pythia-12b	Qwen: 0.7796 (1)	Pythia: 1.5698 (5)	0.7903

F.2 Robustness under LoRA-SFT

For the LoRA-SFT evaluation in Section 5.6, we used eight source models from the Llama and Qwen families. Each model was trained on 4,000 instruction–response examples from No Robots (Rajani et al., 2023) with seeds 42 and 123, and evaluated at epochs 1 and 3. LoRA (Hu et al., 2022) used rank 16 and alpha 32 and targeted q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, and down_proj. Before evaluation, adapters were merged into the original weights so that metric computation incorporated the updated value–output projections. All checkpoints were evaluated using the same 100 prompts, which were not used for training, and the classifier settings in Appendix A.

Table 12: MAGE family predictions for post-LoRA-SFT checkpoints. With the pre-training 23-model reference bank fixed, correct family predictions were retained for all 32 checkpoints from eight source models, two seeds, and two epochs (32/32). Removing the source checkpoint wherever present in the bank and recomputing family representatives produced identical predictions (32/32). This condition uses 22 reference models for seven source models and retains all 23 for Qwen2.5-3B, which was not in the original bank. Other checkpoints from the same family remain in the bank.

Source model	True family	Seed 42 Epoch 1	Seed 42 Epoch 3	Seed 123 Epoch 1	Seed 123 Epoch 3
Llama-3.2-1B	Llama	Llama	Llama	Llama	Llama
Llama-3.2-1B-Instruct	Llama	Llama	Llama	Llama	Llama
Llama-3.2-3B	Llama	Llama	Llama	Llama	Llama
Llama-3.2-3B-Instruct	Llama	Llama	Llama	Llama	Llama
Qwen2.5-0.5B	Qwen	Qwen	Qwen	Qwen	Qwen
Qwen2.5-1.5B	Qwen	Qwen	Qwen	Qwen	Qwen
Qwen2.5-1.5B-Instruct	Qwen	Qwen	Qwen	Qwen	Qwen
Qwen2.5-3B	Qwen	Qwen	Qwen	Qwen	Qwen
Correct / Total	—	8/8	8/8	8/8	8/8

Training settings. Table 13 summarizes the training configuration. Instructions and responses use the same text format across models, with instruction and padding tokens excluded from the loss. Examples exceeding 512 tokens after tokenization were excluded rather than truncated.

Table 13: LoRA-SFT training settings. Each seed starts independently from the source checkpoint, and one epoch comprises 250 optimizer steps. Validation examples are not used for parameter updates.

Setting	Value
Training / validation examples	4,000 / 200 (No Robots)
Seeds	42, 123
Epochs / optimizer steps	3 / 750; evaluation at epochs 1 and 3

Table 13 (continued)

Setting	Value
Maximum sequence length	512 tokens
Micro-batch / gradient accumulation	2 examples / 8 steps
Effective batch size	16 examples
Optimizer / learning rate	AdamW (Loshchilov & Hutter, 2019) / 2×10^{-4}
Weight decay	0
Learning-rate schedule / warmup	Cosine / 0.03 of optimizer steps
LoRA rank / alpha / dropout	16 / 32 / 0.05
Target projections	q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj
Training precision / merged export precision	BF16 / FP16
Training loss	Assistant-response and EOS tokens only

Projection changes and merge validation. All layers showed nonzero LoRA updates to both W_V and W_O. Across layerwise projections in the 32 post-training checkpoints, the relative Frobenius change $\|\Delta W\|_F/\|W\|_F$ ranged from 0.005921 to 0.052721 for W_V and from 0.009076 to 0.045493 for W_O. Adapters were merged in FP32, and pre- and post-merge logits were compared on the first 32 tokens of a validation example for each export. The maximum relative Frobenius error was 6.74×10^{-6}, below the validation threshold of 10^{-4}. Metrics were then extracted from FP16 exports, with the pre-training checkpoints evaluated under the same FP16 conditions. This check establishes numerical agreement on the selected input.

Profile preservation. We computed Pearson correlations between pre- and post-training native-layer profiles for each matched prompt. For each post-training checkpoint, we took the median correlation over 100 prompts. Across the 32 checkpoints, these medians ranged from 0.992394 to 0.999512 for S_l, from 0.999665 to 0.999981 for D_l, and from 0.994906 to 0.999831 for $\widetilde{D}_l$. All 100 correlations were valid for every checkpoint and metric. These results support preservation of layerwise profile shape, in addition to family predictions, under the evaluated LoRA-SFT conditions; they do not imply unchanged metric magnitudes.

F.3 Base-Lineage Attribution of Distilled Models

Table 14 lists the publicly documented base checkpoints and MAGE predictions for the four DeepSeek distilled models in Section 5.6. The predicted family matches the documented base family for three of the four models. Table 15 reports the family-level DTW costs and ranks for the mismatched case, DeepSeek-R1-Distill-Qwen-14B.

Table 14: Publicly documented base checkpoints and MAGE predictions for the DeepSeek distilled models. Base checkpoints follow the official model card from DeepSeek-AI (2025). A match indicates agreement between the predicted and documented base families; the number of matches out of evaluated models is 3/4.

Model	**Public base checkpoint**	**Base family**	**MAGE prediction**	**Match**
DeepSeek-R1-Distill-Qwen-1.5B	Qwen2.5-Math-1.5B	Qwen	Qwen	Yes
DeepSeek-R1-Distill-Qwen-7B	Qwen2.5-Math-7B	Qwen	Qwen	Yes
DeepSeek-R1-Distill-Llama-8B	Llama-3.1-8B	Llama	Llama	Yes
DeepSeek-R1-Distill-Qwen-14B	Qwen2.5-14B	Qwen	DeepSeek	No

Table 15: DTW costs and ranks for DeepSeek-R1-Distill-Qwen-14B across the six families. Ranks are based on ascending unrounded costs, and values are rounded to four decimal places. The MAGE prediction, DeepSeek, and the publicly documented base family, Qwen, are indicated separately.

Family	**DTW cost**	**Rank**	**Note**
DeepSeek	0.4771	1	MAGE prediction
Qwen	0.7909	2	Public base family
SmolLM2	0.8656	3	—
Llama	0.9805	4	—
Pythia	1.8303	5	—
Gemma	2.1135	6	—

G Hyperparameter Sensitivity and Additional Analyses

We examine how the classification results depend on the number and composition of prompts, the DTW window, the interpolation grid, and the combination of attention energy metrics. All analyses reuse the collected profiles of the same 23 reference models, 30 ordinary unseen models, and four separately evaluated distilled models. No additional model inference is required. Unless varied explicitly, we retain 1,000 prompts per model, a 64-point grid, prompt-wise and metric-wise z-score normalization, all three metrics, and a DTW window of three grid points. The default setting reproduces the 25/30 family classification result and 3/4 base-family agreement reported in Section 5.

These are post-hoc sensitivity analyses on the existing development panel, not evaluations on newly held-out models. We retain the original setting rather than selecting a replacement from the results below. Repeated prompt subsets quantify variability within the collected prompt pool; they do not provide confidence intervals over independent models. The evaluation unit remains a model checkpoint, and related Base/Instruct checkpoints may have correlated outcomes. Distilled-model results measure agreement with documented base families, not identification of a distillation teacher.

G.1 Prompt Budget

For each of 20 seeds (0–19), we randomly permute the 1,000 prompt IDs without replacement and take nested subsets of size $n \in \{1, 5, 10, 25, 50, 100, 250, 500, 1000\}$. We use the same selected IDs for all models and metrics. In the *matched* condition, both reference and query model representatives are reconstructed from the selected prompts. In the *fixed-bank* condition, reference representatives retain all 1,000 prompts and only query representatives use the subset. The complete 1,000-prompt condition is evaluated once because it contains the same prompts for every seed. Table 16 reports the mean and sample standard deviation of the number of correct predictions.

Table 16: Prompt-budget sensitivity. Entries are mean ± standard deviation of correct predictions, out of 30 ordinary unseen models or four distilled models, across 20 prompt subsets. The full 1,000-prompt condition is deterministic.

	Ordinary unseen models (out of 30)		Distilled models (out of 4)	
Prompts	Matched	Fixed bank	Matched	Fixed bank
1	23.35 ± 1.39	24.25 ± 1.12	2.60 ± 0.50	3.00 ± 0.00
5	23.70 ± 1.13	24.05 ± 0.89	2.65 ± 0.49	3.00 ± 0.00
10	24.00 ± 0.86	24.35 ± 0.67	2.70 ± 0.47	3.00 ± 0.00
25	24.50 ± 0.76	24.55 ± 0.51	2.85 ± 0.37	3.00 ± 0.00
50	24.75 ± 0.55	24.80 ± 0.41	2.95 ± 0.22	3.00 ± 0.00
100	24.65 ± 0.49	24.75 ± 0.44	3.00 ± 0.00	3.00 ± 0.00
250	24.90 ± 0.31	25.00 ± 0.00	3.00 ± 0.00	3.00 ± 0.00
500	25.00 ± 0.00	25.00 ± 0.00	3.00 ± 0.00	3.00 ± 0.00
1000	25	25	3	3

With matched budgets, 10 and 100 prompts yield means of 24.00/30 and 24.65/30, respectively, compared with 25/30 using all prompts. At 100 prompts, the correct count ranges from 24 to 25; the five default errors persist in every repeat, while Qwen2.5-32B-Instruct is correct in 13 of 20 repeats. At 500 prompts, all 20 repeats reproduce every default ordinary-model prediction. With a fixed bank, this agreement is already observed at 250 query prompts. These results suggest that fewer prompts can retain much of the identifying information in this pool, while the smallest budgets introduce additional variability. The 1,000-prompt analysis in Section 5.2 serves the separate purpose of characterizing within-model profile consistency across a broad set of inputs.

G.2 Prompt Composition

To examine composition at a fixed budget, we sample 50 prompts without replacement from each category and reconstruct both reference and query representatives, using the same 20 seeds. We also consider English-only, Korean-only, and a pooled set of prompts labeled Korean, Japanese, Chinese, French, or Spanish. The latter excludes the `mixed` and `noisy` labels. Table 17 includes uniformly sampled 50-prompt subsets as a reference. The long-context category contains exactly 50 prompts, so its repeats all use the same set.

Table 17: Prompt-composition sensitivity with 50 prompts per reference and query model. Means, standard deviations, and observed ranges refer to 20 subsets; the long-context row repeatedly uses the same complete category.

Prompt pool	Ordinary: mean ± SD / 30	Range	Distilled: mean / 4
Uniform random	24.75 ± 0.55	23–25	2.95
Coding	22.00 ± 0.00	22–22	2.00
Creative writing	23.25 ± 0.55	22–24	2.00
Factual / everyday	23.00 ± 0.00	23–23	3.00
Humanities / society	23.85 ± 0.75	22–25	2.00
Logic / math reasoning	23.00 ± 0.00	23–23	3.00
Long-context reasoning	27.00 ± 0.00	27–27	3.00
Noisy / ambiguous	23.00 ± 0.00	23–23	2.00
Safety / refusal	23.00 ± 0.00	23–23	2.00
Summarization	25.00 ± 0.00	25–25	3.00
Technical explanation	23.05 ± 0.22	23–24	3.00
Text transformation	25.00 ± 0.00	25–25	3.00
Translation	25.00 ± 0.00	25–25	3.00
English	23.30 ± 0.47	23–24	2.15
Korean	23.00 ± 0.00	23–23	3.00
Pooled named non-English	24.85 ± 0.37	24–25	3.00

Composition affects classification even though within-model profiles show high median cross-prompt correlations. Coding-only subsets yield 22/30, whereas the complete long-context category yields 27/30; in the latter, Qwen2.5-14B and Qwen2.5-14B-Instruct become correct relative to the default setting. This is an exploratory observation, not a basis for replacing the mixed prompt set with the highest-scoring category. Category and language subsets also differ in content and length, so these comparisons do not isolate causal language effects or establish robustness to unseen prompt distributions.

G.3 DTW Window and Interpolation Grid

At $G = 64$, we vary the DTW window from zero to the full grid width while retaining all 1,000 prompts and all three normalized metrics (Table 18). With $w = 0$, only the diagonal path is available, and the cost equals the squared Euclidean distance between the preprocessed profiles. This yields 19/30, compared with 25/30 at the default $w = 3$, supporting the utility of local depth alignment in this evaluation. Larger windows do not consistently improve the results. Although $w = 8$ also yields 25/30, only 28 of its 30 ordinary-model predictions agree with the default; identical correct counts need not imply identical decisions.

Table 18: DTW-window sensitivity at 64 grid points and 1,000 prompts. Agreement counts ordinary-model predictions identical to the default $w = 3$ predictions, whether correct or incorrect.

w	$w/(G-1)$	Ordinary correct	Distilled matches	Default agreement
0	0.0000	19/30	3/4	24/30
1	0.0159	23/30	3/4	28/30
2	0.0317	23/30	3/4	28/30
3	0.0476	25/30	3/4	30/30
4	0.0635	24/30	3/4	29/30
6	0.0952	24/30	3/4	29/30
8	0.1270	25/30	3/4	28/30
12	0.1905	22/30	2/4	27/30
16	0.2540	23/30	2/4	26/30
32	0.5079	23/30	2/4	26/30
63	1.0000	23/30	2/4	26/30

Changing the grid size while holding $w = 3$ also changes the allowed displacement in relative depth. We therefore compare a fixed three-point window with $w_G = \lfloor 3(G-1)/63 + 0.5 \rfloor$, which approximately preserves the default relative-depth width. Each profile is interpolated again from the original native-layer values and normalized on the new grid; scaling becomes $1/\sqrt{3G}$. Table 19 shows that increasing grid resolution does not monotonically improve classification. Differences remain after matching the relative-depth window, indicating sensitivity to discretization and alignment rather than only to window units.

Table 19: Interpolation-grid sensitivity using 1,000 prompts. The matched window w_G approximately preserves the default relative-depth width $3/63$. Each result gives ordinary correct predictions / 30 and, after the semicolon, distilled base-family matches / 4.

G	Fixed $w = 3$	Matched w_G	$w_G/(G-1)$	Matched-window result
32	22/30; 2/4	1	0.0323	21/30; 2/4
48	21/30; 2/4	2	0.0426	22/30; 3/4
64	25/30; 3/4	3	0.0476	25/30; 3/4
80	23/30; 3/4	4	0.0506	23/30; 3/4
100	23/30; 3/4	5	0.0505	23/30; 3/4
128	22/30; 3/4	6	0.0472	23/30; 3/4
192	21/30; 3/4	9	0.0471	24/30; 3/4

G.4 Metric Combination and Normalization

Table 20 compares all seven nonempty combinations of the three metrics, with 1,000 prompts, $G = 64$, and $w \in \{0, 3\}$. Model and family representatives are reconstructed for each combination. For K active metrics, the common scaling is $1/\sqrt{64K}$, and a single DTW path is shared across the active channels. Raw-profile variants omit prompt-wise z-score normalization.

At $w = 3$, the normalized three-metric combination yields 25/30, compared with 20/30, 24/30, and 22/30 for S_l, D_l, and $\widetilde{D}_l$ alone. However, D_l alone, $\widetilde{D}_l$ alone, and their combination each match all four documented base families, whereas the three-metric combination matches three. Thus, the joint representation is useful for ordinary family identification in this panel, but is not uniformly superior across tasks. Removing z-score normalization reduces the three-metric ordinary-model result to 21/30, while base-family agreement remains 3/4. These comparisons concern the downstream classifier and do not negate the observed cross-prompt consistency of the individual metrics.

G.5 Excluding a Model from Its Family Representative

The descriptive bank comparison in Section 5.3 includes each model in its own family representative. To assess this self-inclusion effect, we repeat the comparison after excluding the evaluated

Table 20: Metric-combination and normalization ablation. Each entry gives ordinary family classification correct / 30, followed by distilled base-family matches / 4. All conditions use 1,000 prompts and 64 grid points.

Metrics	z-score, $w = 0$	z-score, $w = 3$	Raw, $w = 0$	Raw, $w = 3$
S_l	19/30; 2/4	20/30; 2/4	16/30; 2/4	19/30; 2/4
D_l	22/30; 4/4	24/30; 4/4	21/30; 3/4	21/30; 3/4
$\widetilde{D}_l$	21/30; 4/4	22/30; 4/4	18/30; 2/4	20/30; 2/4
$S_l + D_l$	21/30; 2/4	22/30; 2/4	21/30; 3/4	21/30; 3/4
$S_l + \widetilde{D}_l$	20/30; 3/4	22/30; 2/4	16/30; 2/4	19/30; 2/4
$D_l + \widetilde{D}_l$	23/30; 4/4	23/30; 4/4	21/30; 3/4	21/30; 3/4
All three	19/30; 3/4	25/30; 3/4	21/30; 3/4	21/30; 3/4

model from its family's average while retaining all other references and the default preprocessing. Correct family matches decrease from 23/23 to 22/23. The only changed prediction is Pythia-1B: SmolLM2 has cost 0.8493, while Pythia has cost 0.8725. The remaining families retain 4/4 Llama, 5/5 Qwen, 4/4 Gemma, 4/4 SmolLM2, and 2/2 DeepSeek matches, while Pythia retains 3/4. This check separates self-inclusion from the observed within-bank structure; it does not replace the evaluation on models outside the reference bank or remove dependencies among related checkpoints.

H Scalar-Weighted Graph Baseline Comparison

H.1 Baseline Definition and Matched Evaluation Protocol

The scalar baseline in Section 5.4 is motivated by the scalar attention-graph analyses and spectral fingerprinting of Noël (2025; 2026). It uses the same normalized token representations and attention probabilities as MAGE but replaces the value–output transformation in D_l with the identity matrix I, yielding the layerwise energy

$$E_l^{\text{scalar}} = \frac{1}{HT} \sum_{h,i,j} \alpha_{ij}^{(l,h)} \left\| \bar{x}_i^{(l)} - \bar{x}_j^{(l)} \right\|_2^2 .$$

Here, H and T denote the number of attention heads in the layer and the input token count, respectively. Causal/local masks and valid token ranges are identical to those used for MAGE. The scalar baseline uses this single energy channel, whereas MAGE uses its three metrics.

We evaluate the 23 reference checkpoints in Appendix A.1 and the 15 ordinary unseen models listed in Table 21, drawn from the 30-model evaluation set. The scalar baseline and MAGE correctly classified 10/15 and 13/15 unseen models, respectively. Table 21 reports the model-level results.

We selected 10% of the prompts within each category from the 1,000-prompt set in Appendix D.1 using seed 42, yielding the same 100 prompts for every reference and query model. Prompt selection was fixed before inspecting baseline results. Extraction used the prompt-only protocol without chat templates or response generation. Weights were stored in float16; attention and metric computations used float32, while profile preprocessing and DTW used float64.

Model and family representatives were reconstructed from the 100 prompts for each method. Following Appendix A, we linearly interpolated profiles onto a 64-point relative-depth grid, applied z-score normalization separately for each prompt and metric, and averaged over prompts. Family representatives were equally weighted averages of their reference model profiles. We applied the common scaling $1/\sqrt{64K}$, with $K = 1$ for the scalar baseline and $K = 3$ for MAGE. Both methods used a DTW window of three, squared Euclidean local costs, and family prediction by minimum accumulated cost. This comparison evaluates scalar energy and MAGE under a matched classification protocol; sensitivity to metric combinations is examined separately in Appendix G.4.

H.2 Model-Level Classification Results

Table 21: Classification results for 15 unseen models using the same 23 reference models and 100 prompts. Model names omit the repository organization. Incorrect prediction cells are shaded light red and marked with a dagger (†).

Model	True family	Scalar baseline prediction	MAGE prediction
Llama-3.1-8B	Llama	Llama	Llama
Llama-3.1-8B-Instruct	Llama	Llama	Llama
Qwen2.5-3B	Qwen	Qwen	Qwen
Qwen2.5-3B-Instruct	Qwen	Qwen	SmolLM2 †
gemma-3-4b-pt	Gemma	Gemma	Gemma
gemma-3-4b-it	Gemma	Gemma	Gemma
SmolLM2-135M	SmolLM2	Llama †	SmolLM2
SmolLM2-135M-Instruct	SmolLM2	DeepSeek †	SmolLM2
pythia-70m	Pythia	Qwen †	Gemma †
pythia-160m	Pythia	Qwen †	Pythia
gemma-3-12b-pt	Gemma	Gemma	Gemma
Qwen2.5-7B	Qwen	Qwen	Qwen
Meta-Llama-3-8B	Llama	Llama	Llama
gemma-2-9b	Gemma	Llama †	Gemma
pythia-6.9b	Pythia	Pythia	Pythia
Correct / Total	—	10/15	13/15

Table 22: DTW costs and ranks of the predicted and true families for the two models misclassified by MAGE. Ranks use unrounded costs across the six families. The gap is the true-family cost minus the predicted-family cost; values are shown to six decimal places.

Model	Predicted family: cost (rank)	True family: cost (rank)	Cost gap
Qwen2.5-3B-Instruct	SmolLM2: 0.570948 (1)	Qwen: 0.571429 (2)	0.000481
pythia-70m	Gemma: 0.670617 (1)	Pythia: 1.077489 (2)	0.406871

For Qwen2.5-3B-Instruct, the true family Qwen had the second-lowest cost, differing from the predicted family SmolLM2 by 0.000481. The true family also ranked second for Pythia-70M, but with a larger gap of 0.406871. Both cases are counted as misclassifications.