A NEUROEVOLUTIONARY SYMBOLIC APPROACH TO THE INTERPRETABILITY–PERFORMANCE DILEMMA

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural networks achieve unmatched performance in image classification, but their reasoning remains opaque. Genetic Programming (GP) offers a more transparent alternative by evolving explicit programs, yet it struggles to scale to high-dimensional multiclass image tasks. We address this gap by distilling the latent space of a trained neural network into independent GP symbolic programs, then recombining their outputs through a lightweight classifier head. By training GP students on image-to-latent subobjectives, the proposed decomposition simplifies symbolic search and scales GP-based image classification. On four of the five studied multiclass medical imaging datasets, fully symbolic models are competitive with deep learning baselines. This neuroevolutionary symbolic model exposes an interpretability–performance tradeoff. More programs are typically required to approach teacher performance, but each additional program increases the executable logic to inspect. Because distilled student programs are executable and symbolic, they can be directly inspected, scored, and explained. We use LLM-based judges to score the interpretability of GP functions and function compositions for three stakeholder audiences, namely computer scientists, doctors, and patients. 50 participants of a human study, including clinicians, computer-vision practitioners, and non-experts, chose the LLM-preferred symbolic unit consistently across participant groups, judge families, and composed operation chains. Audience-specific scores induce distinct interpretability–performance frontiers, which we illustrate on BloodMNIST. Symbolic program diversity can therefore support stakeholder-aware model selection and attenuate the tradeoff by prioritizing programs tailored to a given audience. More broadly, this suggests a path toward expressing opaque neural representations as symbolic components that can be selected, inspected, and explained according to the needs of different stakeholders.

1 INTRODUCTION

Neural Networks (NNs) have become the dominant paradigm for image classification, driven by the success of architectures such as ResNet (He et al., 2016), DenseNet (Huang et al., 2017), and Vision Transformers (Dosovitskiy et al., 2021). Their impact extends to medical image analysis, where deep learning has enabled strong predictive models across a wide range of clinical tasks (Miotto et al., 2018). However, high predictive accuracy is not sufficient in critical domains: models should also support transparency, decomposability and simulatability (Lipton, 2018; Rudin, 2019). This is particularly important in healthcare, where black-box predictions can be difficult to trust, audit, or translate into clinically meaningful evidence (Ghassemi et al., 2021).

Explainable Artificial Intelligence (XAI) offers many ways to probe a trained classifier, but such explanations indicate what is associated with a prediction rather than exposing the executable computation that produces it, and they can be unstable or insufficient in healthcare settings (Adebayo et al., 2018; Ghassemi et al., 2021). This motivates complementary approaches in which the predictive structure is itself inspectable.

Genetic Programming (GP), a symbolic evolutionary approach, offers a complementary route to inspectability by searching directly over executable programs rather than optimizing network weights (Koza, 1992). In image analysis, GP programs can be built from human-designed visual operators such as filters, morphology, region operations, and scalar transformations, making the

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.

learned pipeline decomposable by construction. However, direct GP search becomes difficult for high-dimensional multiclass image classification: a single evolutionary process must simultaneously discover useful visual features, compose them into robust programs, and implement multiclass decision logic. Prior GP-based image analysis has largely addressed binary classification (De La Torre et al., 2025b) and segmentation (Cortacero et al., 2023), where the predictive output is simpler and the search space more tractable.

We address this scaling bottleneck by using neural models as guides for decomposition. A teacher network is trained on the original task, and its latent space is treated as a discriminative bottleneck. Instead of evolving a symbolic classifier directly from images to labels, we distill each latent coordinate into independent Multi-Modal Adaptive Graph Evolution (MAGE) programs, where MAGE is a typed graph-based GP method. This converts one coupled multiclass symbolic search into a collection of image-to-scalar approximation problems guided by the teacher representation. The squared latent reconstruction error decomposes additively across coordinates, so the sub-problems are independent and can be searched in parallel, and several seeds per coordinate yield complementary approximations that the classifier head can combine when no single program reproduces a coordinate perfectly. Adding programs can recover more of the teacher's predictive performance, but also increases the number of executable components to inspect. The method therefore exposes an explicit interpretability–performance frontier over subsets of symbolic programs.

Interpretability, however, is not one-size-fits-all. A program that is clear to a computer scientist may be difficult for a clinician to relate to medical image evidence, and a clinically meaningful operation may still be opaque to a patient. We therefore treat interpretability as stakeholder-conditioned: the relevant question is not only whether a model is interpretable, but for whom it is interpretable. To operationalize this view at scale, we score symbolic program units with LLM judges (Liu et al., 2023; Zheng et al., 2023), and check their rankings against human participants on a subsample. Given descriptions of the primitives available to MAGE, the judges score functions and observed function compositions from the perspective of different stakeholders: computer scientists, doctors, and patients. These scores induce different rankings of the same symbolic program dictionary.

We then use these rankings to assemble audience-specific symbolic models and trace interpretability–performance frontiers. Rather than interpreting a fixed model after training, stakeholder preferences become part of the model-selection process: programs judged more interpretable for a target audience can be prioritized while monitoring the predictive cost of doing so. In our experiments, this makes it possible to study whether different audiences can recover strong performance through different, audience-preferred symbolic subsets.

We organize the study around three questions: **RQ1:** Can latent-space distillation scale GP-based image classification toward neural performance on multiclass medical benchmarks? **RQ2:** What interpretability–performance tradeoff emerges when neural performance is recovered by adding symbolic programs? **RQ3:** Does conditioning interpretability on a target audience reshape this tradeoff, allowing different stakeholders to recover performance through different symbolic subsets?

Our contributions are a neuroevolutionary symbolic distillation pipeline that maps the latent representation of a neural teacher into a dictionary of executable MAGE programs for multiclass medical image classification, and a stakeholder-conditioned interpretability framework built on LLM-scored symbolic units.

2 Related Work

Interpretable neural image classification and medical XAI. XAI methods for neural image classifiers include local surrogate explanations (Ribeiro et al., 2016), saliency and attribution maps (Selvaraju et al., 2017; Sundararajan et al., 2017), concept-based explanations (Kim et al., 2018; Koh et al., 2020), prototype models (Chen et al., 2019), and interpretable CNN architectures (Zhang et al., 2018). They are useful inspection tools, but expose partial evidence about a trained predictor rather than the executable computation producing the decision, and in medical imaging a visually plausible explanation may still be unstable or insufficient for clinical use (Adebayo et al., 2018; Ghassemi et al., 2021). We instead distill neural representations into executable symbolic programs that can be inspected, scored, and recombined.

Genetic programming for image analysis. Evolutionary computation optimizes populations of candidate solutions (Back & Schwefel, 1996), and GP applies this principle to executable programs (Koza, 1992), making it well suited to white-box modeling (Mei et al., 2023). A major graph-based variant is Cartesian Genetic Programming (CGP), which represents programs as directed acyclic graphs and reuses intermediate computations (Miller & Thomson, 2000; Miller, 2020). In computer vision, GP and CGP have been used for image processing, feature construction, segmentation, and classification (Harding et al., 2013; Leitner et al., 2012; Bi et al., 2018; 2021; Khan et al., 2021; Bi et al., 2022; Cortacero et al., 2023). Some GP image-classification methods construct visual features that are passed to downstream classifiers or ensembles (Fan et al., 2024; Wu et al., 2025). Building on CGP, MAGE introduces typed multimodal graph programs, restricting search to type-valid compositions while allowing specialized image-processing function libraries (Torre et al., 2024), and has been competitive on binary biomedical image classification (De La Torre et al., 2025b; 2026) and interpretable vision control (De La Torre et al., 2025a). We address the difficulty of direct multiclass search by using a neural teacher to decompose it into latent image-to-scalar distillation targets.

Knowledge distillation and interpretable students. Knowledge distillation transfers information from a trained teacher to a student model, originally for compression and later for representation transfer across architectures (Buciluǎ et al., 2006; Hinton et al., 2015). Beyond matching final predictions, feature- and attention-based distillation train students to reproduce intermediate teacher representations, showing that internal neural signals can provide richer supervision than hard labels alone (Romero et al., 2015; Zagoruyko & Komodakis, 2016). Distillation has also been used to obtain more interpretable students, including tree-structured or rule-based approximations of neural models (Craven & Shavlik, 1995; Frosst & Hinton, 2017). More recent symbolic distillation work replaces neural components with symbolic expressions or surrogates (Cranmer et al., 2020; Tan et al., 2026). Symbolic distillation and GP for image classification are each established, but their combination is not. Prior symbolic distillation fits scalar functions or non-vision components, as in SymTorch (Tan et al., 2026), rather than image-to-latent targets, and prior GP vision work evolves image-to-label pipelines end to end on smaller or binary problems. The student here is a symbolic approximation of internal visual representations rather than a smaller network or a decision tree.

Stakeholder-dependent interpretability and LLM judges. Interpretability is a multifaceted concept covering simulatability, decomposability, algorithmic transparency, and post-hoc explanation (Lipton, 2018), and its ambiguity has motivated calls for rigorous, task-grounded evaluation (Doshi-Velez & Kim, 2017; Miller, 2019). Stakeholder-centered XAI shows that different users have different desiderata, so explanation needs vary across roles and contexts (Langer et al., 2021; Brennen, 2020). In clinical settings, explanations are more useful when they connect model logic to clinical reasoning (Cutillo et al., 2020; Tonekaboni et al., 2019). This motivates our audience-conditioned setting, where computer scientists, doctors, and patients induce different notions of program interpretability. Since exhaustive human evaluation is costly for large symbolic program libraries, we use LLM judges as scalable evaluators (Liu et al., 2023; Zheng et al., 2023), treating their scores as proxy measurements rather than ground truth and aggregating repeated judgments across models (Wang et al., 2024). Unlike prior work that evaluates explanations of fixed predictors, we use stakeholder-conditioned scores to select among executable symbolic components and construct audience-specific interpretability–performance frontiers.

3 METHOD

The method proceeds by training a neural teacher, distilling each latent coordinate into MAGE symbolic programs, and recombining selected programs through a lightweight classifier head. The same program library is then scored to construct audience-specific interpretability–performance frontiers. An illustration of the method is shown in Figure 1.

3.1 NEURAL TEACHER AND LATENT DECOMPOSITION

The first stage trains a neural teacher on the original multiclass classification task. The teacher is not used as the final deployed model; its role is to learn a compact discriminative representation that later serves as the target of symbolic distillation. Let $D = H \times W \times K$ denote the number of input pixels

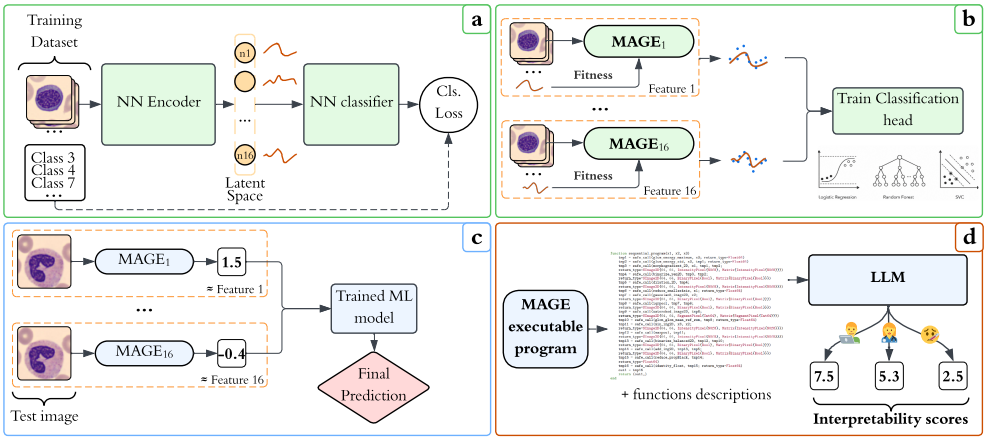


Figure 1: Neuroevolutionary symbolic workflow for image classification. (a) A neural teacher learns a controlled latent bottleneck. (b) Each latent coordinate is distilled into independent MAGE programs, and selected programs are recombined by a lightweight classifier. (c) Inference uses only the selected MAGE programs and trained head, discarding the neural encoder. (d) Programs and function descriptions are scored by LLM judges to obtain stakeholder-conditioned interpretability estimates.

for an image with height H, width W, and K channels. The teacher maps the high-dimensional image space to a controlled latent space of dimension d, with $d \ll D$. In this work, $d = 16$.

We factorize the teacher into a latent map and a classifier head. Given an image x, the latent representation is $z_\theta(x) = \rho(g_\theta(x)) \in \mathbb{R}^d$, where g_θ is the neural encoder and ρ the latent activation function, here asinh. The classifier head $h_\omega : \mathbb{R}^d \to \mathbb{R}^C$ maps the latent representation to class logits, and the teacher prediction is $T_{\theta,\omega}(x) = \mathrm{softmax}(h_\omega(z_\theta(x)))$, so $z_\theta(x)$ is the interface between image and teacher decision.

We consider supervised multiclass image classification with input space $\mathcal{X}$ and labels $\mathcal{Y} = \{1, \ldots, C\}$. A dataset is $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, with $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$, split into $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{val}}$, and $\mathcal{D}_{\text{test}}$. For a classifier $f : \mathcal{X} \to \mathcal{Y}$, we write $\hat{y}_i = f(x_i)$. Because the biomedical datasets considered in this work are class-imbalanced, we use Balanced Accuracy (BACC), the average per-class recall, as the primary evaluation metric.

The teacher is trained on the original labels by minimizing a class-weighted cross-entropy, with weights that compensate for class imbalance. After training, z_θ is fixed and used to define the targets for symbolic distillation.

We interpret z_θ as a discriminative bottleneck: its coordinates are not assumed to be independently meaningful or sufficient, but jointly define a low-dimensional representation on which the teacher head separates the classes. This bottleneck induces a decomposition of the symbolic learning problem. Direct end-to-end GP would require searching for a program that maps images directly to labels, $x \mapsto y$. Instead, the trained teacher exposes d scalar target functions, $x \mapsto z_{\theta,j}(x)$ for $j = 1, \ldots, d$, redirecting symbolic search from a coupled multiclass decision problem to a collection of image-to-scalar approximation problems.

The squared latent reconstruction error decomposes additively,

$$\|\hat{z}(x) - z_\theta(x)\|_2^2 = \sum_{j=1}^{d} (\hat{z}_j(x) - z_{\theta,j}(x))^2,$$

so the bottleneck approximation can be optimized coordinate-wise before recombination.

3.2 MAGE Symbolic Program Class

We instantiate each symbolic student with MAGE, a GP method extending CGP with typed, multimodal program graphs (Koza, 1992; Miller & Thomson, 2000; Miller, 2020; Torre et al., 2024). Let $\mathcal{H}_{\text{MAGE}}$ denote the hypothesis class searched by MAGE. Each $p \in \mathcal{H}_{\text{MAGE}}$ is a typed directed acyclic graph decoded into an executable image-processing program. MAGE restricts search to

type-valid compositions over intensity-image, binary-mask, segmentation, and scalar primitives. In this work, each student program is an image-to-scalar function $p : \mathcal{X} \to \mathbb{R}$, used to approximate one coordinate of the teacher bottleneck.

For latent coordinate j, MAGE evolves programs to match the teacher signal $z_{\theta,j}(x)$. For a mini-batch $B \subset \mathcal{D}_{\text{train}}$, the distillation loss of a program p is

$$\mathcal{L}_{\text{distill}}(p, j; B) = 1 - \left| r_{\text{S}} \left((p(x_i))_{i \in B}, (z_{\theta,j}(x_i))_{i \in B} \right) \right|,$$

where r_{S} is Spearman rank correlation. This objective rewards monotonic agreement with the teacher coordinate rather than exact numerical reconstruction. The absolute value makes the objective invariant to sign reversal, since the downstream classifier can invert or rescale symbolic features.

Evolution uses mini-batch fitness estimates for scalability. At each generation, the current elite is additionally evaluated on $\mathcal{D}_{\text{val}}$; the final program for a run is the elite encountered during the run with the best validation distillation score. Repeating this process for each coordinate j and random seed r yields candidate programs $p_{j,r}$, which form the symbolic dictionary used in the next stage.

3.3 Symbolic Bottleneck, Classifier Head, and Decision Preservation

After symbolic distillation, the method no longer relies on the neural encoder. Let $\mathcal{P}$ be the dictionary of MAGE programs obtained from the independent distillation runs. For any selected subset $S \subseteq \mathcal{P}$, we define the symbolic bottleneck map $\Phi_S(x) = (p(x))_{p \in S} \in \mathbb{R}^{|S|}$. The vector $\Phi_S(x)$ is the symbolic analogue of the teacher latent representation.

A lightweight classifier head $c_\beta : \mathbb{R}^{|S|} \to [0,1]^C$, with $\sum_{c=1}^{C} c_\beta(\Phi_S(x))_c = 1$, is trained on the original labels. The final symbolic model is $F_{S,\beta}(x) = c_\beta(\Phi_S(x))$. The programs in S are kept fixed while β is optimized: MAGE programs approximate the teacher's latent signals, and the classifier head recombines the resulting symbolic features for prediction.

Multiple imperfect programs can be useful when, together, they provide a richer basis for approximating the teacher bottleneck, once a simple calibration absorbs the differences of scale, sign and redundancy between symbolic outputs and neural coordinates. A calibrated bottleneck lying in the same decision region as $z_\theta(x)$ would leave the teacher's prediction unchanged (Appendix B). We train c_β on the symbolic features instead of reusing h_ω, so the deployed model can use a lighter and more inspectable classifier such as logistic regression. Adding programs comes at an immediate complexity cost, which we first measure as the number of selected programs $|S|$, and later refine into audience-specific interpretability costs.

The method therefore converts multiclass image classification into a symbolic bottleneck selection problem: choose a subset S that is predictive enough to recover performance while remaining small and interpretable enough to be inspected.

3.4 Stakeholder-Conditioned Interpretability Scoring

The symbolic bottleneck makes the final model inspectable at the program level, but interpretability is not audience-independent. We therefore define a set of stakeholder groups $\mathcal{G} = \{\text{Computer Scientist}, \text{Doctor}, \text{Patient}\}$, where each group $g \in \mathcal{G}$ induces a different interpretability scoring function over MAGE programs.

Let Ω be the set of MAGE primitives. We consider two scoring granularities. In unigram scoring, each unit is a primitive $u = f$, with $f \in \Omega$. In bi-gram scoring, units are empirically observed compositional patterns extracted from active programs: if f is applied to outputs produced by $g_1, \ldots, g_k$, the scored unit is $u = f(g_1, \ldots, g_k)$; if f is applied directly to the input, the scored unit is $u = f$.[1] Let $\mathcal{U}$ denote the set of scored units under the chosen granularity. LLM judges receive primitive names and descriptions, then score the relevant units for each stakeholder group.

For each scored unit $u \in \mathcal{U}$, stakeholder group g, judge model m, and repetition r, we obtain a score $s_{g,m,r}(u) \in \{1, \ldots, 10\}$, larger values meaning higher perceived interpretability, and average

[1] For example, $\text{add}(\text{sobel}(x), \text{erosion}(x))$ yields $\text{add}(\text{sobel}, \text{erosion})$, while $\text{power2}(x)$ yields the unigram power2.

repeated judgments as $\bar{s}_g(u) = \frac{1}{MR}\sum_{m=1}^{M}\sum_{r=1}^{R} s_{g,m,r}(u)$. We treat $\bar{s}_g(u)$ as a proxy score for unit u and group g.

We then lift unit-level scores to program-level scores. For a MAGE program p, let $U(p)$ be the multiset of scored units extracted from its active graph under the chosen granularity. We define average interpretability as $I_g^{\text{avg}}(p) = |U(p)|^{-1}\sum_{u\in U(p)} \bar{s}_g(u)$, and conservative interpretability as $I_g^{\min}(p) = \min_{u\in U(p)} \bar{s}_g(u)$. The average score captures the overall comprehensibility of the program, while the minimum score captures whether the program contains a particularly opaque operation or composition. Unless stated otherwise, we use $I_g(p) = I_g^{\text{avg}}(p)$.

This procedure scores the full library and every observed composition, which exhaustive human evaluation could not cover. We also report a human study on agreement with LLM rankings. LLMs are used to compare large libraries of symbolic programs under explicit stakeholder assumptions, allowing the same program library to be ranked differently for different target audiences. The scores are local, assigned to a primitive or to an observed composition, and I_g^{avg} and $I_g^{\min}$ recombine them into a program score. We do not claim either rule is the best way to score a whole program, since both abstract away structural properties. They are, however, simple and transparent aggregations which enable audience-conditioned selection that already induces different tradeoffs. Appendix I shows conditional structure penalized by two of the three audiences, and Appendix W details the study on agreement between LLM scores and human participants.

3.5 Audience-Specific Interpretability–Performance Frontiers

The final symbolic model is determined by a selected program subset $S \subseteq \mathcal{P}$. For stakeholder group g, we convert interpretability into a cost $C_g(p) = 10 - I_g(p)$, since scores lie on a 1–10 scale. The subset cost is $C_g(S) = \sum_{p\in S} C_g(p)$, increasing with both the number of programs and their audience-specific opacity.

For each subset S, predictive performance is $\text{Perf}(S) = \text{BACC}(F_{S,\beta}; \mathcal{D}_{\text{val}})$. Model selection is then a bi-objective problem: maximize $\text{Perf}(S)$ while minimizing $C_g(S)$. The non-dominated pairs $(C_g(S), \text{Perf}(S))$ define the audience-specific interpretability–performance frontier.

Since enumerating all subsets of $\mathcal{P}$ is intractable, we build empirical frontiers from ordered prefixes. A performance-first policy ranks programs solely by classifier's feature importance. An audience-first policy ranks programs by decreasing $I_g(p)$, using feature importance to break ties. Each prefix defines a subset S and therefore one point on the frontier.

These frontiers show how stakeholder preferences change which symbolic approximations are selected from the same program dictionary. The method therefore treats interpretability not only as a post-hoc description of a trained model, but as a criterion for assembling symbolic approximations of the neural bottleneck.

4 Experimental Setup

Datasets We evaluate on five multiclass MedMNIST benchmarks (Yang et al., 2021; 2023), namely BloodMNIST, OrganAMNIST, OrganCMNIST, OCTMNIST, and PathMNIST. They cover blood-cell microscopy, abdominal CT, retinal OCT, and colorectal histology, with 4 to 11 classes and fixed train/validation/test splits. This covers five of the eight multiclass MedMNIST datasets and spans color and grayscale images, small and large training sets, and class counts from 4 to 11. We use all datasets for predictive evaluation and BloodMNIST for stakeholder-conditioned interpretability analysis. A stakeholder-conditioned analysis is reported in full on BloodMNIST, where distillation performs best, and Appendix O repeats it on OrganAMNIST and OrganCMNIST with the same conclusions. Dataset details are in Appendix A.

Compared Methods We compare four model families. The **neural teacher** is a ResNet-18 initialized from ImageNet weights, trained with the latent bottleneck of Section 3.1. The value $d = 16$ is a controlled choice of the same parameter for every dataset, above the largest class count of 11, so that this bottleneck does not limit teacher accuracy or collapse into a one-hot signal; it is also far below the 512 dimensions of a standard ResNet-18. Full training hyperparameters, augmentation settings, and architecture details are reported in Appendix A. The **neural baseline** uses the same training

Table 1: Test BACC (%) comparison across MedMNIST datasets. Δ is the gap between the best neural baseline and the best distilled model.

Dataset	Dataset size	MAGE end-to-end		MAGE Distill									Baselines			Δ
		Best on Val.	Avg	Best per dim			Runs per dim			All per dim			ResNet18	ResNet34	ResNeXt50	
				RF	SVM	LR	RF	SVM	LR	RF	SVM	LR				
Blood	17,092	67.3	58.1 ± 7.0	95.3	94.0	91.1	96.8	96.7	96.4	97.0	97.3	97.6	98.8	98.5	99.1	1.5
Organ A	58,830	42.2	27.1 ± 10.7	81.3	80.8	78.2	89.5	90.1	89.0	90.1	91.0	91.0	95.8	96.1	95.8	5.1
Organ C	23,583	37.4	21.9 ± 10.1	80.6	80.5	79.3	86.6	87.8	87.7	88.0	90.4	90.3	93.2	93.8	94.4	4.0
OCT	109,309	46.7	41.3 ± 5.7	66.5	38.2	71.2	66.0	66.8	74.3	70.9	64.0	77.3	91.3	91.2	91.3	14.0
Path	107,180	54.7	42.8 ± 9.3	85.4	82.3	81.8	87.6	88.3	87.1	87.7	89.8	89.5	93.7	91.8	92.9	3.9

protocol without the controlled latent bottleneck, measuring standard neural-classifier performance. Convergence curves are shown in Appendix T.

The **end-to-end MAGE** baseline evolves MAGE programs directly for multiclass classification, with 16 independent runs per dataset. The **distilled MAGE** models are obtained by independently distilling each teacher latent coordinate. For each dataset, we run $d \times R$ searches, with $d = 16$ latent coordinates and $R = 8$ random seeds, using the objective of Section 3.2. For each dataset, distillation uses 128 searches and end-to-end MAGE uses 16, each run capped at 20 hours of wall-clock time. Because the runs are independent they execute in parallel, so the elapsed time is also 20 hours, and the totals are at most 640 node-hours for distillation and 80 for end-to-end MAGE. Distillation is required once per dataset. It runs on CPU alone, and inference afterwards executes a handful of small programs with no neural network and no accelerator. Appendix L details the computational cost, and Appendix M compares distillation and end-to-end MAGE at a matched evolutionary-search budget.

We evaluate three nested program dictionaries. *Best per dimension* contains, for each latent coordinate, the validation-best program among the 8 distillation runs. *Runs per dimension* contains the validation-best program from every run. *All per dimension* further includes intermediate training-batch elites saved during evolution, giving the largest dictionary.

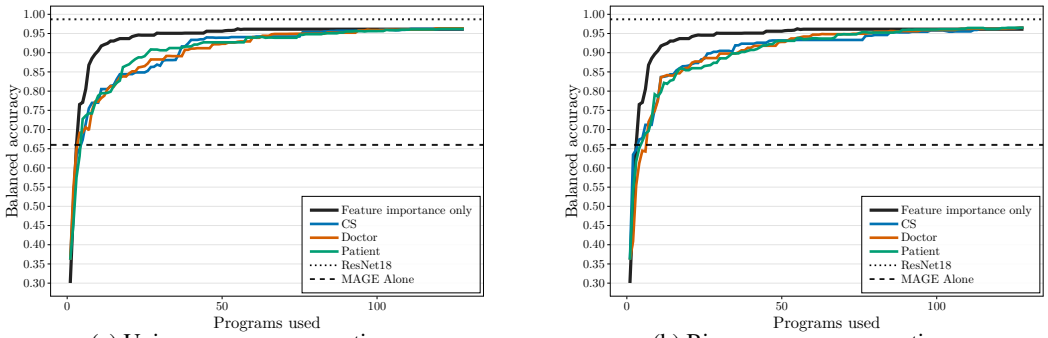
5 Results

5.1 RQ1: Latent-Space Distillation Scales GP-Based Image Classification

Table 1 presents the central empirical finding of this work. A fully symbolic pipeline composed entirely of named, executable image-processing programs can approach neural network performance on multiclass medical image classification. Both MAGE settings search the same primitive libraries, so the gap between direct evolution and distillation comes from how the problem is decomposed rather than from the operations available. To our knowledge this is the first GP-based latent neural-to-symbolic distillation pipeline demonstrated on multiclass biomedical images, and the first to come within a few balanced-accuracy points of neural baselines on several full-data MedMNIST tasks with no neural network retained at inference.

Direct end-to-end MAGE struggles on all five datasets, and on BloodMNIST, the best validation-selected run reaches only 67.3% BACC, with $58.1 \pm 7.0\%$ across runs, see Appendix V for more details. Latent-space distillation produces a qualitative leap. On BloodMNIST, distilled MAGE reaches 95.3% BACC with the best-per-dimension program dictionary and up to 97.6% with larger dictionaries, closing all but $1-2$ percentage points of the gap to the ResNet baselines ($98.8-99.1\%$). Comparable gains appear on OrganAMNIST (91.0% vs. 96.1% neural), OrganCMNIST (90.4% vs. 94.4% neural), and PathMNIST (89.8% vs. 93.7% neural), where distilled MAGE comes within 3.9 to 5.1 percentage points of the best neural baseline. OCTMNIST remains hardest (77.3% vs. 91.3% neural), suggesting dataset-dependent bottleneck-approximation limits, see Appendix U. Distillation still gains 30.6 points over the best validation-selected end-to-end run on this dataset. As the decomposition helps here as well, we believe the remaining gap comes from a mismatch between the current primitive library and the operations this task needs. Stronger or weaker, every model in this study would need expert verification before use in a real medical setting, and the OCTMNIST gap would have to be weighed there as well.

The same ordering holds under macro F1 and macro one-versus-rest AUROC, and the distilled models are far better calibrated than end-to-end MAGE, matching or improving on the best neural calibration



(a) Unigram, mean aggregation. (b) Bi-gram, mean aggregation.

Figure 2: Audience-specific interpretability–performance frontiers on BloodMNIST test data, under unigram and bi-gram scoring with mean aggregation of unit scores into program scores. Feature-importance ordering recovers performance fastest, stakeholder-conditioned orderings induce distinct frontiers and still recover high balanced accuracy, showing that audience preferences can guide symbolic program selection without preventing performance recovery. Appendix N gives the corresponding frontiers under minimum aggregation.

on several datasets. Appendix C reports every metric for every dictionary and head. Across datasets, the gap between direct and distilled MAGE shows that the performance gain comes from decomposing the task through the teacher latent space. These results support RQ1 and suggest a path by which symbolic methods can reach performance levels previously thought to require neural architectures. On top of that, since MAGE programs are executable code, one can follow their prediction logic and intermediate values. The audience can set how many they are willing to interpret, so the final model can be chosen to fit that budget rather than fixed and explained afterwards (Appendix P).

5.2 RQ2: Performance Recovery Creates an Interpretability–Performance Tradeoff

Figure 2 shows that balanced accuracy increases as more distilled programs are added to the symbolic bottleneck. The gains are front-loaded: the symbolic model surpasses the end-to-end MAGE baseline after only a handful of programs, reaches roughly 0.90 BACC with about 10 programs, and approaches 0.95 after roughly 25–30 programs. Beyond this point, gains become progressively smaller and the curve saturates around 0.96–0.97, within 2% of the ResNet baseline. The practical implication is that a model inspecting only ~ 10 programs already captures most of the symbolic system's ceiling performance, and recovering the remaining gap to neural performance requires a much larger program set with corresponding interpretability cost. A rising curve is partly expected, since adding variables to a classifier head tends to increase predictive capacity, so we test the value of the ordering against chance and structural controls. Against 20 random permutations of the same dictionary, informed ordering reaches 90% balanced accuracy with 10 programs where a random ordering needs 28 on average, and 95% with 29 against 73, so it recovers a given level of performance with roughly 40% as many components to inspect. Ordering instead by the number of active operations, a structural criterion computed without any LLM, reproduces the same diminishing-return shape, with programs of at most 11 operations already reaching 90.8%. Appendix K also compares the early subsets selected for different audiences.

5.3 RQ3: Stakeholder Preferences Reshape the Tradeoff

Figure 2 shows that feature importance recovers balanced accuracy fastest, while stakeholder-conditioned orderings, which prioritize programs judged more interpretable for their audience, trace distinct frontiers while also reaching high performance as more programs are added. Unigram and bi-gram scores produce similar frontiers. Appendix J reports audience-specific scores, and Appendix N gives validation and test curves for BloodMNIST. Feature importance and audience-aware orderings are shown for OrganAMNIST and OrganCMNIST in Appendix O.

The audience scores behind these orderings come from LLM judges, which can score every unit of the dictionary, whereas human scoring is only feasible on a subsample. We therefore asked 50 volunteers, in general, clinical, and computer-vision groups matched to the patient, doctor, and CS audiences, which of two units they could more easily explain to someone without a computer-vision

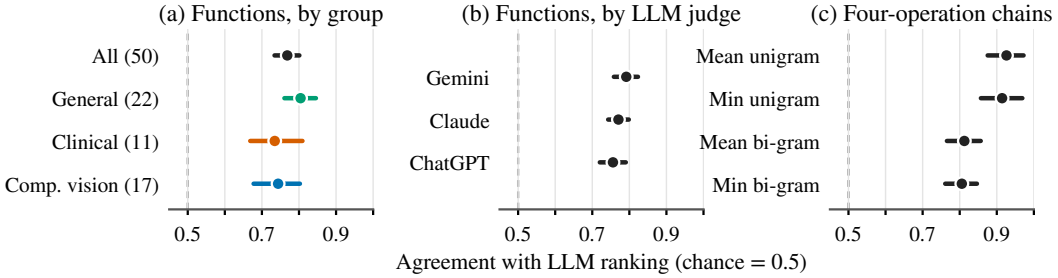


Figure 3: Share of decided pairwise comparisons in which participants chose the unit the LLM scored higher, with 95% participant-clustered bootstrap intervals. (a) Functions by participant group, scored for the matched audience. (b) Functions by judge family. (c) Four-operation chains by program-level score.

background, over the 302 MAGE primitives and 881 four-operation excerpts of the BloodMNIST programs. They chose the unit the LLM preferred in 77% of 1,559 decided function comparisons, and agreement is above chance in every group, for every judge family, and for every program-level score of Section 3.4 on composed chains (Figure 3, Appendix W). This agreement lets the LLM rankings scale human judgement across the full dictionary, but does not establish clinical validity or the interpretability of complete programs, which remain questions for evaluation with the intended users.

Together, these results support RQ3. Each audience can prioritize the programs it finds interpretable and still recover near-neural performance. Following our definition of $C_g(S)$, inspection cost rises with every program added, and we define that cost as additive, though for a given stakeholder it may grow differently. The ordering comes from the audience score with feature importance breaking ties, so another cost or another rule would trace a different route through the same dictionary. There is therefore no single tradeoff, only one per audience and rule, which is what makes it navigable.

6 Discussion

Our main contribution is demonstrating that symbolic methods can approach neural performance through decomposition. NNs and symbolic methods are often treated as occupying separate performance tiers, the first accurate but opaque, the second inspectable but assumed to sacrifice too much accuracy on high-dimensional tasks. Our results challenge that assumption. The key insight is that a NN does not need to be the final deployed model. It can serve as a decomposition oracle, and the programs that result are transparent, decomposable, and simulatable in ways that are inherently difficult with NNs. Coming within a small gap of the neural baselines is already a qualitatively different option than what GP-based image classification has offered before (Cortacero et al., 2023), and richer primitive libraries, interpretability-aware search, and more diverse image domains may narrow it further.

Limitations. All experiments use low-resolution MedMNIST images and one set of manually designed primitive libraries. Transfer to higher-resolution images or other imaging domains, and sensitivity to the choice of primitives, remain open. Search cost grows linearly with the latent dimension and the number of seeds, up to 640 node-hours per dataset (Appendix L), and we leave ablations of the latent dimension, the libraries, the search budget and the evolutionary parameters to future work. While the human study confirmed interpretability rankings from LLMs, it compared pairs of primitives and four-operation chains. This supports the LLM scores as a proxy for the relative inspectability of components, not their absolute values or the interpretability of complete programs and assembled classifiers. Finally, no model in this study is validated for clinical use, and none should be used for diagnosis.

While there remain many open directions, through this study we find that the interpretability–performance tradeoff is real, quantifiable, and navigable. Direct empirical measurements of this tradeoff are rare (Rudin, 2019), and this study gives one in explicit form. LLM-based scoring makes stakeholder-conditioned selection scalable. Computer scientists, doctors, and patients induce different frontiers that all reach high performance, which supports specifying interpretability relative to users and tasks (Lipton, 2018; Langer et al., 2021) and shows that the tradeoff can be attenuated by assembling programs in an audience-aware way. We hope that this encourages further study into accurate and inherently interpretable representations.

Ethics Statement

For evaluation of interpretability, we conducted a human study, detailed in Appendix W. Participants were volunteers who compared symbolic operations and four-operation chains for ease of understanding. Before starting, they confirmed that they were at least 18 years old and gave informed consent. The study was granted exempt status by an institutional review board on the basis of its scope. Participation was uncompensated, and participants could stop at any time without giving a reason. We recorded their choices, response times, and self-reported background category, but collected no names, email addresses, IP addresses, or medical information. The evaluated models are research prototypes and are not validated for clinical use.

AI Use Statement

During the preparation of this work the authors used LLM-based assistants to draft and edit text and to generate figure code. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the content of the paper. The LLM judges used to score interpretability are part of the method and are described in Section 3.4 and Appendix E.

Reproducibility Statement

The code for the experiments, the figure generation, and the analysis of the LLM scores is available at `https://anonymous.4open.science/r/NeuroEvoSymbo-for-image-classification-E636/README.md`. Appendix A documents the data preparation, the teacher and baseline training settings, and the dataset splits, all of which are the fixed splits distributed with MedMNIST. Appendix R gives the evolutionary parameters of both MAGE settings and Appendix S the classifier-head grids. Appendix L reports the search cost in full. Appendix E gives the judge models and the complete prompts, and Appendix W the protocol, the sampling procedure, and the exclusions of the human study.

References

Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *Advances in neural information processing systems*, 31, 2018.

Thomas Back and H-P Schwefel. Evolutionary computation: An overview. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 20–29. IEEE, 1996.

Ying Bi, Mengjie Zhang, and Bing Xue. Genetic programming for automatic global and local feature extraction to image classification. In *2018 IEEE congress on evolutionary computation (CEC)*, pp. 1–8. IEEE, 2018.

Ying Bi, Bing Xue, and Mengjie Zhang. *Genetic programming for image classification: An automated approach to feature learning*. Springer Nature, 2021.

Ying Bi, Bing Xue, Pablo Mesejo, Stefano Cagnoni, and Mengjie Zhang. A survey on evolutionary computation for computer vision and image analysis: Past, present, and future trends. *IEEE Transactions on Evolutionary Computation*, 27(1):5–25, 2022.

Andrea Brennen. What do people really want when they say they want" explainable ai?" we asked 60 stakeholders. In *Extended abstracts of the 2020 CHI conference on human factors in computing systems*, pp. 1–7, 2020.

Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 535–541, 2006.

Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: deep learning for interpretable image recognition. *Advances in neural information processing systems*, 32, 2019.

Kévin Cortacero, Brienne McKenzie, Sabina Müller, Roxana Khazen, Fanny Lafouresse, Gaëlle Corsaut, Nathalie Van Acker, François-Xavier Frenois, Laurence Lamant, Nicolas Meyer, et al. Evolutionary design of explainable algorithms for biomedical image segmentation. *Nature communications*, 14(1):7112, 2023.

Miles Cranmer, Alvaro Sanchez Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel, and Shirley Ho. Discovering symbolic models from deep learning with inductive biases. *Advances in neural information processing systems*, 33:17429–17442, 2020.

Mark Craven and Jude Shavlik. Extracting tree-structured representations of trained networks. *Advances in neural information processing systems*, 8, 1995.

Christine M Cutillo, Karlie R Sharma, Luca Foschini, Shinjini Kundu, Maxine Mackintosh, Kenneth D Mandl, and MI in Healthcare Workshop Working Group Beck Tyler 1 Collier Elaine 1 Colvis Christine 1 Gersing Kenneth 1 Gordon Valery 1 Jensen Roxanne 8 Shabestari Behrouz 9 Southall Noel 1. Machine intelligence in healthcare—perspectives on trustworthiness, explainability, usability, and transparency. *NPJ digital medicine*, 3(1):47, 2020.

Camilo De La Torre, Giorgia Nadizar, Yuri Lavinas, Hervé Luga, Dennis Wilson, and Sylvain Cussat-Blanc. Evolution of inherently interpretable visual control policies. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 358–367, 2025a.

Camilo De La Torre, Giorgia Nadizar, Yuri Lavinas, Robin Schwob, Camille Franchet, Hervé Luga, Dennis Wilson, and Sylvain Cussat-Blanc. Evolved and transparent pipelines for biomedical image classification. In *European Conference on Genetic Programming (Part of EvoStar)*, pp. 173–189. Springer, 2025b.

Camilo De La Torre, Stéphane Treillard, Camille Hervé Luga, Dennis Wilson, and Sylvain Cussat-Blanc Franchet. A comparative study on robustness in evolved image classifiers. In *Genetic Programming: 29th European Conference, EuroGP 2026, Held as Part of EvoStar 2026, Toulouse, France, April 8–10, 2026, Proceedings*, pp. 34. Springer Nature, 2026.

Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

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

Qinglan Fan, Ying Bi, Bing Xue, and Mengjie Zhang. A multi-tree genetic programming-based ensemble approach to image classification with limited training data [research frontier]. *IEEE Computational Intelligence Magazine*, 2024. doi: 10.1109/MCI.2024.3446148.

Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, 2017.

Marzyeh Ghassemi, Luke Oakden-Rayner, and Andrew L Beam. The false hope of current approaches to explainable artificial intelligence in health care. *The lancet digital health*, 3(11):e745–e750, 2021.

Simon Harding, Jürgen Leitner, and Juergen Schmidhuber. Cartesian genetic programming for image processing. In *Genetic programming theory and practice X*, pp. 31–44. Springer, 2013.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. 2016. doi: 10.1109/CVPR.2016.90.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Gao Huang, Zhuang Liu, Laurence Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4700–4708, 2017.

Asifullah Khan, Aqsa Saeed Qureshi, Noorul Wahab, Mutawarra Hussain, and Muhammad Yousaf Hamza. A recent survey on the applications of genetic programming in image processing. *Computational Intelligence*, 37(4):1745–1778, 2021.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pp. 2668–2677. PMLR, 2018.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International conference on machine learning*, pp. 5338–5348. PMLR, 2020.

John R. Koza. Genetic programming: On the programming of computers by means of natural selection. 1992.

Markus Langer, Daniel Oster, Timo Speith, Holger Hermanns, Lena Kästner, Eva Schmidt, Andreas Sesing, and Kevin Baum. What do we want from Explainable Artificial Intelligence (XAI)? A stakeholder perspective on XAI and a conceptual model guiding interdisciplinary XAI research. *Artificial Intelligence*, 296:103473, 2021.

Jürgen Leitner, Simon Harding, Alexander Förster, and Jürgen Schmidhuber. Mars terrain image classification using cartesian genetic programming. *International Symposium on Artificial Intelligence*, 2012.

Zachary C Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-Eval: NLG evaluation using GPT-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 2511–2522, 2023.

Yi Mei, Qi Chen, Andrew Lensen, Bing Xue, and Mengjie Zhang. Explainable artificial intelligence by genetic programming: A survey. *IEEE Transactions on Evolutionary Computation*, 27(3): 621–641, 2023. doi: 10.1109/TEVC.2022.3225509.

Julian F. Miller. Cartesian genetic programming: its status and future. *Genetic Programming and Evolvable Machines*, 2020. doi: 10.1007/S10710-019-09360-6.

Julian F. Miller and P.F. Thomson. Cartesian genetic programming. *European Conference on Genetic Programming*, 2000. doi: 10.1007/978-3-540-46239-2_9.

Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019.

Riccardo Miotto, Fei Wang, Shuang Wang, Xiaoqian Jiang, and Joel T Dudley. Deep learning for healthcare: review, opportunities and challenges. *Briefings in bioinformatics*, 19(6):1236–1246, 2018.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144, 2016.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets, 2015. URL `https://arxiv.org/abs/1412.6550`.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215, 2019.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pp. 618–626, 2017.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pp. 3319–3328. PMLR, 2017.

Elizabeth SZ Tan, Adil Soubki, and Miles Cranmer. Symtorch: A framework for symbolic distillation of deep neural networks. *arXiv preprint arXiv:2602.21307*, 2026.

Sana Tonekaboni, Shalmali Joshi, Melissa D McCradden, and Anna Goldenberg. What clinicians want: contextualizing explainable machine learning for clinical end use. In *Machine learning for healthcare conference*, pp. 359–380. PMLR, 2019.

Camilo De La Torre, Yuri Lavinas, Kévin Cortacero, Hervé Luga, Dennis G. Wilson, and Sylvain Cussat-Blanc. Multimodal adaptive graph evolution for program synthesis. *Lecture notes in computer science*, 2024. doi: 10.1007/978-3-031-70055-2_19.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, et al. Large language models are not fair evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 9440–9450, 2024.

Zhicheng Wu, Bing Xue, and Mengjie Zhang. Multitree gp-based feature learning for multimodal medical image classification. In *2025 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8. IEEE, 2025.

Jiancheng Yang, Rui Shi, and Bingbing Ni. MedMNIST classification decathlon: A lightweight automl benchmark for medical image analysis. In *2021 IEEE 18th international symposium on biomedical imaging (ISBI)*, pp. 191–195. IEEE, 2021.

Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. MedMNIST v2 – a large-scale lightweight benchmark for 2d and 3d biomedical image classification. *Scientific data*, 10(1):41, 2023.

Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016.

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

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P Xing, et al. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

A EXPERIMENTAL SETUP DETAILS

The teacher is trained on images resized to 224×224 and optimized with AdamW. The checkpoint with the highest validation BACC is used for distillation.

In both MAGE settings the program returned by a run is the validation-best elite encountered. Both use the primitive libraries in Appendix D, with evolutionary parameters in Appendix R.

We conduct the stakeholder-conditioned interpretability analysis on BloodMNIST, OrganAMNIST and OrganCMNIST for computer scientists, doctors, and patients, representing technical users, clinical users, and non-expert beneficiaries. LLM judges score symbolic units from MAGE primitive descriptions under each stakeholder definition on a 1–10 scale; repeated scores across judge models and repetitions are averaged into $\bar{s}_g(u)$. Appendix D lists primitives; Appendix E gives judge models, prompts, and examples; Appendices F, G, and H analyze audience, model, and transition effects.

For each dictionary or selected subset $S \subseteq \mathcal{P}$, we compute symbolic features $\Phi_S(x)$ and train a classifier head c_β on the original labels. Hyperparameters are selected by validation BACC, and reported predictive results are evaluated on the test split. For performance comparisons, we evaluate logistic regression, random forest, and SVM heads. For interpretability–performance frontiers, we use logistic regression on nested subsets of the runs-per-dimension dictionary because it is lightweight and decomposable. Classifier-head hyperparameters are reported in Appendix S.

For the interpretability–performance frontiers, programs are added one at a time either by feature importance alone or by stakeholder-conditioned interpretability score, using feature importance to break ties. Feature importance comes from a logistic-regression head trained on the full candidate dictionary. Along each ordering, subsets are evaluated on validation data and the corresponding test scores are reported in the main curves, with the matching validation and test curves in Appendix N. Interpretability costs use the program-level scores of Section 3.4.

For automated qualitative analysis, representative programs are rendered as code and given to an LLM with the primitive descriptions and stakeholder definitions, which produces stakeholder-specific explanations of the program. Examples are in Appendix Q.

We also manually explain a representative program on two images from different classes and discuss its implications for the decision process. The example is in appendix P.

Code availability. Code for experiments, figure generation, and LLM-score analysis is available at `https://anonymous.4open.science/r/NeuroEvoSymbo-for-image-classification-E636/README.md`.

In Table 2, we report the number of samples and classes for the five datasets used in this work. Each has a preset split of train, validation, and test images.

Dataset	Train	Val.	Test	# Classes
BloodMNIST	11,959	1,712	3,421	8
OrganAMNIST	34,561	6,491	17,778	11
OrganCMNIST	12,975	2,392	8,216	11
OCTMNIST	97,477	10,832	1,000	4
PathMNIST	89,996	10,004	7,180	9

Table 2: The MedMNIST datasets used in this work.

Table 3 reports the main hyperparameters used to train the neural teacher whose latent representation is later distilled by GP programs. The teacher uses a ResNet-18 backbone initialized with ImageNet-pretrained weights, and all input images are resized to 224×224. The latent bottleneck has dimension $d = 16$ and uses an asinh activation; in preliminary experiments, this activation reduced latent-coordinate collapse into near-unimodal or near-bimodal responses.

The teacher uses a 16-dimensional bottleneck, far narrower than the 512-dimensional representation of a standard ResNet-18, so it is worth checking that the bottleneck does not itself limit teacher performance. Table 4 compares the bottlenecked teacher with the three unconstrained baselines. The teacher matches or exceeds the standard ResNet-18 on four of the five datasets and is 0.3 points

Setting	Value
Backbone	ResNet-18
Initialization	ImageNet-pretrained
Input resolution	224×224
Optimizer	AdamW
Learning rate	10^{-4}
Weight decay	0.1
Epochs	100
Latent dimension	16
Latent activation	asinh

Table 3: Teacher training settings. The trained teacher provides the fixed latent targets distilled by the MAGE student programs.

below it on OrganAMNIST, showing that at $d = 16$ the constraint costs little, and the gap between the distilled symbolic models and the baselines in Table 1 reflects the difficulty of approximating the neural representation rather than a weak teacher. At the same time, we wanted a latent dimension that was larger than the maximum number of classes (11) to prevent collapse to one-hot encodings.

Table 4: Test BACC (%) of the bottlenecked teacher against the unconstrained neural baselines. The teacher is a ResNet-18 whose classification layer is replaced by a 16-dimensional bottleneck followed by a small head.

Dataset	Teacher ($d = 16$)	ResNet18	ResNet34	ResNeXt50
BloodMNIST	98.8	98.8	98.5	99.1
OrganAMNIST	95.5	95.8	96.1	95.8
OrganCMNIST	94.2	93.2	93.8	94.4
OCTMNIST	91.9	91.3	91.2	91.3
PathMNIST	94.4	93.7	91.8	92.9

Table 5 reports the main hyperparameters used to train the neural baselines. All baseline models use ImageNet-pretrained weights and receive images resized to 224×224. Training is run for 30 epochs with validation-based checkpoint selection; Figures 19 and 20 show that the validation curves have converged by this point.

Setting	Value
Models	ResNet-18, ResNet-34, ResNeXt-50
Initialization	ImageNet-pretrained
Input resolution	224×224
Optimizer	AdamW
Learning rate	10^{-4}
Weight decay	10^{-4}
Epochs	30

Table 5: Training settings for the neural baseline classifiers.

Table 6 reports the dataset-specific batch sizes and training augmentations used for the teacher and neural baselines. For ResNet-34 and ResNeXt-50, the batch size is set to 128 for all datasets due to memory constraints. Horizontal and vertical flips are used for BloodMNIST and PathMNIST, where class labels are expected to be largely invariant to image orientation. For OrganAMNIST, OrganCMNIST, and OCTMNIST, which are grayscale datasets, color jitter is restricted to brightness and contrast. All data loaders apply normalization using training-set mean and standard deviation.

Dataset	Batch size	RandomResizedCrop	ColorJitter	Rand. HFlip	Rand. VFlip
BloodMNIST	128	yes	B/C/S/H	yes	yes
OrganAMNIST	128	yes	B/C	no	no
OrganCMNIST	512	yes	B/C	no	no
OCTMNIST	512	yes	B/C	no	no
PathMNIST	512	yes	B/C/S/H	yes	yes

Table 6: Dataset-dependent training and augmentation settings. B/C/S/H denotes brightness, contrast, saturation, and hue jitter; B/C denotes brightness and contrast jitter only. For ResNet-34 and ResNeXt-50, the batch size is 128 for all datasets due to memory constraints.

Experiments involving Neural Networks were run on a workstation with an Intel Xeon W-11955M CPU and an NVIDIA RTX A5000 Laptop GPU with 16 GB of memory.

Teacher architecture The teacher uses a ResNet-18 backbone initialized with ImageNet-pretrained weights. The original ResNet classification layer is removed and replaced by a bottleneck head that maps the 512-dimensional pooled representation to a $d = 16$ latent vector. This latent vector is passed through the asinh activation and then used by the final classifier head.

Classifier head (for teacher training) architecture The teacher classifier head h_ω is a small MLP with dropout that maps the $d = 16$ latent vector to the C class logits during teacher training. After training, h_ω is discarded: only the fixed latent map z_θ is retained to generate distillation targets for the MAGE students.

B Bottleneck Approximation Criterion

The quality of a symbolic bottleneck approximation can be measured by its calibrated discrepancy from the teacher bottleneck,

$$\varepsilon(S) = \inf_{A \in \mathcal{A}} \mathbb{E}_x \left[\| A\Phi_S(x) - z_\theta(x) \|_2^2 \right],$$

where $\mathcal{A}$ is a simple calibration family, such as linear maps. The calibration accounts for differences of scale, sign, and redundancy between symbolic program outputs and neural latent coordinates, which is why several imperfect programs can be more useful together than any one of them alone.

If a calibration A gives $A\Phi_S(x) \approx z_\theta(x)$, an ideal classifier over the symbolic features can be written as $T_{A,S}(x) = \mathrm{softmax}(h_\omega(A\Phi_S(x)))$. However, the teacher's original classifier h_ω is not retained in this work, since we preferred to train a lighter classifier from scratch. It only shows that a good symbolic bottleneck can support the same decision rule as the teacher, provided $A\Phi_S(x)$ and $z_\theta(x)$ fall on the same side of the teacher head's decision boundaries, thus the two would assign the same label. The learned head c_β is the practical replacement, trained directly from symbolic features to labels.

C Additional Predictive Metrics

Balanced accuracy is the primary metric of comparison since the datasets are class-imbalanced and averaging per-class recall prevents majority classes from dominating. Because no single metric characterizes a classifier in its entirety, this section reports macro F1, macro one-versus-rest AUROC, and expected calibration error alongside BACC for every method, dictionary, and classifier head. One table per dataset details all of these results from Tables 8 to 12. Table 7 summarizes these detailed results by choosing the best head classifier per dictionary size as well as the best neural baseline, for each dataset.

For macro F1 and macro one-versus-rest AUROC, the classifier head parameters were tuned on validation to maximize each respective metric, following the same treatment as for the primary metric of the study, BACC, explained in Appendix S. Expected calibration error (ECE) was not tuned for. We

instead picked the same parameters that maximized BACC on validation, and report ECE alongside for each classifier head, dictionary and dataset.

Macro F1 closely follows BACC on all five datasets, and AUROC gives the same ordering, so the benefit from latent distillation is not an artifact of the primary metric. On calibration, the distilled models are far better calibrated than end-to-end MAGE, and on several datasets the best distilled head matches or improves on the best neural calibration, although calibration not always improved monotonically with dictionary size for a given classifier head c_β.

Table 7: Summary of additional test metrics across datasets. E2E denotes validation-selected end-to-end MAGE. Best and Runs report the strongest head among RF, SVM, and LR for the Best per dimension and Runs per dimension dictionaries. NN reports the strongest neural baseline.

Dataset	Macro F1 ↑					ECE ↓					AUROC ↑				
	E2E	Best	Runs	NN	Δ	E2E	Best	Runs	NN	Δ	E2E	Best	Runs	NN	Δ
Blood	63.5	95.2	97.0	**99.1**	2.1	50.4	1.3	0.7	**0.5**	-0.2	90.4	99.7	99.9	**100.0**	0.1
Organ A	35.3	82.8	90.5	**96.2**	5.7	26.5	**1.3**	3.0	2.0	0.7	75.0	98.3	99.4	**99.8**	0.4
Organ C	27.1	81.0	88.2	**94.4**	6.2	16.7	2.4	**0.9**	1.9	1.0	69.5	98.5	99.0	**99.8**	0.8
OCT	37.4	70.8	73.6	**91.2**	17.6	19.6	**4.9**	5.1	5.2	0.3	76.7	92.2	94.5	**99.5**	5.0
Path	51.3	84.5	87.5	**93.7**	6.2	25.7	1.2	**0.7**	2.4	1.7	85.4	98.7	99.0	**99.8**	0.8

Detailed results are reported for all datasets. In Tables 7 to 12, bold values mark the best result for each dataset and metric, including ties. Higher balanced accuracy, macro F1 and AUROC and lower ECE are preferred. Each Δ column reports the best neural result minus the best distilled result among the methods shown, in percentage points. Positive Δ therefore favors neural models for BACC, macro F1 and AUROC, and distilled models for ECE.

Table 8: Test performance on the BloodMNIST dataset.

Metric	MAGE end-to-end		MAGE Distill									Baselines			Δ
	Best on Val.	Avg	Best per dim			Runs per dim			All per dim			ResNet18	ResNet34	ResNeXt50	
			RF	SVM	LR	RF	SVM	LR	RF	SVM	LR				
BACC (%) ↑	67.3	58.1 ± 7.0	95.3	94.0	91.1	96.8	96.7	96.4	97.0	97.3	97.6	98.8	98.5	**99.1**	1.5
Macro F1 (%) ↑	63.5	50.9 ± 9.3	95.2	93.8	90.2	97.0	96.8	95.9	97.0	97.3	97.3	98.7	98.6	**99.1**	1.8
ECE (%) ↓	50.4	32.9 ± 10.8	7.5	1.3	2.3	8.0	0.7	1.4	8.2	**0.4**	0.6	0.7	0.5	0.6	0.1
AUROC (%) ↑	90.4	81.7 ± 4.2	99.7	99.6	99.3	99.8	99.9	99.8	99.8	99.9	99.9	99.9	99.9	**100.**	0.1

Table 9: Test performance on the OrganAMNIST dataset.

Metric	MAGE end-to-end		MAGE Distill									Baselines			Δ
	Best on Val.	Avg	Best per dim			Runs per dim			All per dim			ResNet18	ResNet34	ResNeXt50	
			RF	SVM	LR	RF	SVM	LR	RF	SVM	LR				
BACC (%)	42.2	27.1 ± 10.7	81.3	80.8	78.2	89.5	90.1	89.0	90.1	91.0	91.0	95.8	**96.1**	95.8	5.1
Macro F1 (%)	35.3	17.1 ± 12.4	82.8	80.7	77.8	89.9	90.5	88.9	90.7	91.5	91.1	95.8	**96.2**	96.0	4.7
ECE (%)	26.5	9.7 ± 9.7	12.3	5.7	1.3	14.2	3.0	3.3	13.4	2.9	**0.4**	2.0	2.0	2.3	1.6
AUROC (%)	75.0	74.3 ± 4.0	98.3	98.0	97.6	99.4	99.2	99.2	99.4	99.4	99.4	**99.8**	**99.8**	99.6	0.4

Table 10: Test performance on the OrganCMNIST dataset.

Metric	MAGE end-to-end		MAGE Distill									Baselines			Δ
	Best on Val.	Avg	Best per dim			Runs per dim			All per dim			ResNet18	ResNet34	ResNeXt50	
			RF	SVM	LR	RF	SVM	LR	RF	SVM	LR				
BACC (%) ↑	37.4	21.9 ±10.1	80.6	80.5	79.3	86.6	87.8	87.7	88.0	90.4	90.3	93.2	93.8	**94.4**	4.0
Macro F1 (%) ↑	27.1	13.0 ±10.7	81.0	80.7	75.9	87.8	88.2	87.6	87.9	90.8	91.0	93.5	94.0	**94.4**	3.4
ECE (%) ↓	16.7	9.1 ±9.8	10.7	3.4	2.4	16.1	2.5	0.9	10.9	1.7	**0.8**	2.7	1.9	3.3	1.1
AUROC (%) ↑	69.5	73.2 ±4.8	98.5	97.8	97.6	99.0	98.9	98.9	99.2	99.2	99.4	**99.8**	**99.8**	**99.8**	0.4

Table 11: Test performance on the OCTMNIST dataset.

Metric	MAGE end-to-end		MAGE Distill									Baselines			Δ
	Best on Val.	Avg	Best per dim			Runs per dim			All per dim			ResNet18	ResNet34	ResNeXt50	
			RF	SVM	LR	RF	SVM	LR	RF	SVM	LR				
BACC (%)	46.7	41.3 ±5.7	66.5	38.2	71.2	66.0	66.8	74.3	70.9	64.0	77.3	**91.3**	91.2	**91.3**	14.0
Macro F1 (%)	37.4	31.8 ±6.9	57.3	30.6	70.8	59.2	65.1	73.6	59.4	62.7	77.4	**91.2**	**91.2**	**91.2**	13.8
ECE (%)	19.6	14.2 ±5.5	**4.9**	55.0	6.2	5.1	23.0	5.7	8.0	24.2	6.1	5.7	5.2	5.7	0.3
AUROC (%)	76.7	67.2 ±6.8	92.2	70.9	92.2	93.4	92.6	94.5	93.0	92.6	95.9	99.0	99.2	**99.5**	3.6

Table 12: Test performance on the PathMNIST dataset.

Metric	MAGE end-to-end		MAGE Distill									Baselines			Δ
	Best on Val.	Avg	Best per dim			Runs per dim			All per dim			ResNet18	ResNet34	ResNeXt50	
			RF	SVM	LR	RF	SVM	LR	RF	SVM	LR				
BACC (%)	54.7	42.8 ±9.3	85.4	82.3	81.8	87.6	88.3	87.1	87.7	89.8	89.5	**93.7**	91.8	92.9	3.9
Macro F1 (%)	51.3	35.1 ±11.2	84.5	81.4	80.5	86.8	87.5	86.2	86.8	89.6	89.3	**93.7**	91.3	93.0	4.1
ECE (%)	25.7	23.9 ±10.1	7.8	1.2	3.3	10.5	2.1	**0.7**	10.1	1.7	1.3	2.4	3.5	4.0	1.7
AUROC (%)	85.4	77.2 ±5.7	98.7	98.2	98.0	99.0	98.9	98.4	99.0	99.3	98.3	**99.8**	99.5	99.6	0.5

Additionally, Tables 13 to 17 report per-class sensitivity and specificity for every dataset, computed one versus rest on the test split. Each entry gives sensitivity followed by specificity, and bold values mark the highest value of each measure separately within a row, including ties. Classifier-head hyperparameters are selected by validation macro F1. The neural baseline columns use the checkpoint with the highest validation BACC, as in Table 1, and the end-to-end column reports the validation-selected run.

The gap is almost entirely in sensitivity, with specificity within 0.4 points of the best neural baseline on every dataset except OCTMNIST, so the distilled models miss some positives rather than raising false alarms. On each dataset a few classes carry most of the sensitivity difference, while the rest stay close to the neural baselines.

On some classes the distilled models match or exceed the baselines, including background in PathMNIST and eosinophils in BloodMNIST at full sensitivity, and cancer-associated stroma in PathMNIST at 71.5 against 69.4, which is the hardest class for every model here. A second group is difficult for both, such as the kidneys in OrganAMNIST and OrganCMNIST and drusen in OCTMNIST, where the baselines also record their lowest sensitivity. The remaining differences are specific to the distilled models, on the heart in both Organ datasets, smooth muscle and normal colon mucosa in PathMNIST, and diabetic macular edema and the normal class in OCTMNIST.

Table 13: Per-class sensitivity and specificity (%) on the BloodMNIST test split.

Class	MAGE	Best per dim			Runs per dim			All per dim			Neural baselines		
	end-to-end	RF	SVC	LR	RF	SVC	LR	RF	SVC	LR	ResNet18	ResNet34	ResNeXt50
basophil	64.3/92.7	96.3/99.6	94.7/99.3	92.2/98.6	96.3/99.8	96.7/99.8	98.0/99.5	97.1/99.7	97.1/99.9	98.8/99.7	99.2/99.9	98.8/**100.0**	**99.6**/**100.0**
eosinophil	51.8/95.4	99.7/99.9	99.5/**100.0**	99.2/99.9	**100.0**/**100.0**	99.7/99.9	99.8/**100.0**	**100.0**/99.9	99.7/**100.0**	99.8/**100.0**	**100.0**/**100.0**	**100.0**/99.9	**100.0**/99.9
erythroblast	64.0/98.1	92.0/99.6	91.6/99.5	89.4/98.5	93.6/99.8	95.8/99.9	95.2/99.6	94.2/99.8	95.2/99.8	96.1/99.8	**99.4**/99.9	99.0/**100.0**	**99.4**/**100.0**
immature granulocytes	35.9/93.5	89.1/97.7	87.6/96.9	75.3/97.0	94.5/98.2	94.0/98.3	89.1/98.9	94.3/98.1	96.2/98.6	94.8/98.9	97.8/99.3	**98.4**/99.0	97.2/**99.5**
lymphocyte	78.6/93.4	97.5/99.5	95.9/99.5	94.2/99.1	98.4/99.8	97.5/99.7	97.9/99.6	98.4/99.8	98.4/99.7	98.4/99.7	98.4/**100.0**	99.2/99.8	**99.6**/99.9
monocyte	81.7/91.7	91.2/99.1	89.4/98.7	84.9/98.0	94.0/99.7	93.0/99.5	93.3/98.7	93.0/99.7	95.4/99.6	95.8/99.5	98.9/99.8	95.4/**99.9**	**99.3**/**99.9**
neutrophil	69.5/97.4	97.0/99.3	94.1/99.6	94.7/99.2	**97.9**/99.5	97.3/99.3	97.6/99.5	97.4/99.5	96.8/99.5	96.8/99.6	96.7/99.6	97.4/**99.7**	**97.9**/99.5
platelet	92.8/97.7	99.4/**100.0**	99.4/**100.0**	98.9/**100.0**	99.8/**100.0**	99.8/**100.0**	**100.0**/**100.0**	99.8/**100.0**	99.6/**100.0**	99.6/**100.0**	99.8/**100.0**	99.8/**100.0**	**100.0**/**100.0**

Table 14: Per-class sensitivity and specificity (%) on the OrganAMNIST test split.

Class	MAGE	Best per dim			Runs per dim			All per dim			Neural baselines		
	end-to-end	RF	SVC	LR	RF	SVC	LR	RF	SVC	LR	ResNet18	ResNet34	ResNeXt50
bladder	58.1/95.3	78.8/98.3	82.0/96.4	76.5/97.1	87.6/99.1	89.5/98.3	88.9/98.4	82.8/99.4	87.0/98.6	88.2/98.6	**95.2**/99.1	92.7/**99.8**	93.0/99.4
femur-left	72.1/95.5	73.0/99.3	74.2/98.9	67.2/98.3	91.3/99.6	90.4/99.7	92.0/99.3	91.7/99.7	91.8/99.9	91.6/99.7	88.6/**100.0**	**94.5**/99.9	90.8/**100.0**
femur-right	0.0/**100.0**	76.8/99.3	77.9/99.0	70.1/98.4	91.7/99.8	90.3/99.7	89.8/99.5	93.8/99.8	90.7/99.8	92.4/99.7	**96.8**/99.8	94.6/99.8	95.3/99.9
heart	59.6/94.3	69.8/99.7	68.0/99.6	64.6/99.4	77.5/99.8	77.7/**100.0**	77.1/99.8	77.8/99.9	82.4/**100.0**	81.1/99.9	**98.5**/**100.0**	96.8/**100.0**	**98.5**/**100.0**
kidney-left	5.7/96.0	67.5/97.3	66.3/97.5	64.2/97.0	75.0/98.4	76.4/98.3	73.7/98.0	76.2/98.2	78.7/98.4	75.9/98.3	83.5/**99.5**	**87.0**/99.1	84.0/99.2
kidney-right	31.7/92.3	77.3/97.1	73.1/97.4	67.2/97.2	83.8/98.1	82.4/98.2	79.7/98.0	85.5/98.3	84.0/98.6	82.6/98.6	93.5/99.2	93.6/99.4	**94.5**/**99.5**
liver	50.4/93.2	93.1/99.2	89.0/99.5	91.2/99.2	98.3/99.7	99.3/99.8	98.8/99.7	99.2/99.7	99.6/99.8	99.6/99.7	99.4/**100.0**	99.2/**100.0**	**99.8**/99.9
lung-left	40.9/95.6	98.5/98.9	97.1/99.2	97.1/99.0	98.7/99.1	99.7/99.9	98.8/99.5	99.6/99.2	99.7/99.9	99.9/99.5	**100.0**/99.9	**100.0**/99.9	**100.0**/**100.0**
lung-right	76.0/93.2	96.5/99.8	97.7/99.9	96.3/99.6	98.1/99.8	98.4/**100.0**	98.0/99.9	99.0/99.9	98.9/**100.0**	99.3/**100.0**	**99.9**/99.9	99.7/99.8	99.6/**100.0**
pancreas	69.9/79.2	86.8/98.9	83.4/98.8	86.1/98.6	93.0/99.2	92.5/99.6	92.3/99.4	93.3/99.3	92.7/99.4	95.7/99.2	**99.8**/99.6	99.4/**99.8**	99.4/99.5
spleen	0.0/**100.0**	85.1/94.9	80.2/94.4	78.6/95.1	89.6/96.9	94.3/96.9	89.6/97.3	92.4/97.0	95.1/96.8	94.2/97.8	98.6/98.7	**99.7**/98.7	99.3/98.6

Table 15: Per-class sensitivity and specificity (%) on the OrganCMNIST test split.

Class	MAGE	Best per dim			Runs per dim			All per dim			Neural baselines		
	end-to-end	RF	SVC	LR	RF	SVC	LR	RF	SVC	LR	ResNet18	ResNet34	ResNeXt50
bladder	22.2/88.9	84.4/96.5	82.2/97.1	78.7/96.5	86.7/97.6	87.2/98.0	86.4/98.5	85.4/97.6	89.5/98.2	91.1/98.7	89.7/98.8	**92.8**/99.1	92.5/**99.4**
femur-left	27.6/91.9	72.2/98.4	75.4/98.3	54.1/99.2	79.6/99.3	82.1/99.3	82.1/98.9	83.5/99.1	88.4/99.5	87.5/99.2	88.4/**99.6**	91.0/**99.6**	**93.7**/**99.6**
femur-right	69.1/89.2	70.1/98.2	70.5/98.5	62.5/98.6	89.5/99.0	86.9/99.2	87.2/98.8	86.7/99.2	91.7/99.5	88.4/99.2	90.0/**99.7**	**96.9**/99.3	95.0/99.6
heart	64.8/92.5	78.4/99.3	75.1/99.5	79.3/98.0	82.9/99.9	81.2/**100.0**	86.5/99.8	86.7/99.8	85.5/**100.0**	89.8/99.9	96.0/99.9	93.3/**100.0**	**96.4**/99.9
kidney-left	0.0/**100.0**	64.5/98.0	68.8/97.1	63.1/97.1	74.6/98.3	77.7/97.5	73.7/97.7	76.2/98.0	82.3/98.0	81.7/98.4	82.5/99.1	**84.6**/99.0	82.9/99.0
kidney-right	53.5/94.0	62.2/98.4	59.9/98.4	58.2/98.2	69.0/98.8	70.7/98.7	70.6/98.1	70.5/99.0	74.4/99.1	76.3/99.0	85.4/99.2	82.0/**99.5**	**86.5**/99.1
liver	0.0/**100.0**	95.0/99.1	93.6/98.9	93.7/98.7	97.6/99.3	97.9/99.5	97.1/99.7	97.4/99.5	98.6/99.4	98.6/99.8	99.7/99.8	**99.8**/99.7	**99.8**/99.7
lung-left	78.9/88.3	94.7/99.8	93.6/99.9	93.6/99.3	97.6/99.9	96.9/**100.0**	97.8/99.9	96.7/99.9	96.9/**100.0**	97.6/99.8	**98.2**/**100.0**	96.9/**100.0**	97.8/**100.0**
lung-right	0.0/**100.0**	97.8/99.8	97.7/99.9	97.8/99.2	98.7/99.8	98.4/99.9	98.6/99.9	98.7/99.6	98.2/99.9	98.4/99.9	**98.9**/99.9	98.0/**100.0**	98.7/99.9
pancreas	52.1/92.2	90.9/97.4	89.9/97.3	88.1/95.2	94.3/98.3	95.1/98.4	94.5/98.7	92.1/98.4	95.5/98.6	95.3/98.9	98.1/98.8	97.6/99.0	**98.9**/**99.1**
spleen	43.2/87.5	80.0/96.4	78.9/95.6	68.3/96.4	91.7/97.2	91.2/97.7	89.9/97.7	91.4/97.6	93.7/98.4	95.5/98.5	98.4/98.8	**99.1**/98.9	96.2/**99.2**

Table 16: Per-class sensitivity and specificity (%) on the OCTMNIST test split.

Class	MAGE	Best per dim			Runs per dim			All per dim			Neural baselines		
	end-to-end	RF	SVC	LR	RF	SVC	LR	RF	SVC	LR	ResNet18	ResNet34	ResNeXt50
choroidal neovascularization	67.2/83.3	98.0/68.4	34.4/**95.7**	95.2/77.5	98.4/65.2	93.6/82.4	95.2/78.7	98.8/65.2	97.2/74.7	98.0/78.8	**100.0**/90.3	99.6/90.4	99.6/91.2
diabetic macular edema	0.0/99.7	45.2/97.7	18.8/92.0	52.8/96.5	57.6/98.9	78.4/93.7	81.6/94.7	56.0/98.9	72.4/96.0	79.2/98.0	**94.4/99.9**	91.2/99.6	92.8/99.7
drusen	21.6/91.5	14.8/99.5	0.4/99.5	57.2/94.4	12.0/**100.0**	61.6/81.2	44.4/95.9	13.2/99.7	50.4/82.7	57.6/94.5	72.0/99.6	**75.2**/99.9	73.2/99.9
normal	98.0/54.4	92.4/84.5	99.2/30.4	80.4/93.5	88.4/88.0	33.6/98.4	76.0/96.5	88.4/88.3	36.0/**98.7**	74.4/98.4	98.8/**98.7**	98.8/98.4	**99.6**/97.6

Table 17: Per-class sensitivity and specificity (%) on the PathMNIST test split.

Class	MAGE	Best per dim			Runs per dim			All per dim			Neural baselines		
	end-to-end	RF	SVC	LR	RF	SVC	LR	RF	SVC	LR	ResNet18	ResNet34	ResNeXt50
adipose	95.1/97.5	**98.4**/99.6	97.2/99.5	98.0/99.6	91.7/99.8	92.5/99.8	94.0/99.7	89.8/99.8	94.5/**99.9**	95.8/99.7	96.6/**99.9**	97.8/99.6	95.1/99.1
background	99.9/99.2	**100.0**/99.6	**100.0**/98.7	**100.0**/99.5	**100.0**/99.8	**100.0/100.0**	**100.0**/99.5	**100.0**/99.9	**100.0/100.0**	**100.0**/99.8	**100.0/100.0**	97.8/**100.0**	95.3/**100.0**
debris	41.0/90.4	89.4/98.1	80.8/98.3	79.4/97.1	92.6/98.7	90.9/98.8	91.1/98.6	92.9/98.7	90.3/99.0	88.2/98.8	98.2/99.3	**99.4**/98.6	98.5/**99.7**
lymphocytes	87.4/99.1	95.3/99.5	93.2/99.6	94.2/99.6	97.2/99.8	96.2/99.7	97.5/99.7	97.6/99.7	97.0/99.8	96.5/99.7	97.3/**100.0**	96.2/99.8	**98.1**/99.4
mucus	21.6/94.6	87.7/99.0	78.7/98.5	79.9/98.8	90.7/99.5	91.0/99.2	92.0/98.6	95.0/**99.6**	97.1/**99.6**	94.9/98.4	**98.7**/99.1	95.3/99.5	96.9/99.0
smooth muscle	0.0/**100.0**	69.1/98.5	58.1/98.7	57.9/98.4	73.3/97.3	75.3/98.1	67.4/98.2	69.9/97.0	76.4/98.3	79.9/98.4	89.4/98.0	**94.9**/97.4	89.4/97.9
normal colon mucosa	38.1/92.7	80.7/97.9	86.9/97.0	83.0/97.3	88.8/98.7	88.4/98.3	89.1/97.5	88.3/98.7	87.4/98.9	93.1/98.7	98.1/**99.5**	97.6/99.4	**99.5**/98.7
cancer-associated stroma	43.9/89.7	62.7/96.3	63.7/94.8	64.6/94.2	62.7/96.8	69.1/97.0	66.7/97.5	64.8/96.9	**71.5**/98.2	65.1/98.6	67.0/99.8	51.5/**99.9**	69.4/99.8
colorectal adenocarcinoma epithelium	65.6/92.2	85.2/97.9	81.9/98.1	79.1/98.3	91.7/98.0	91.1/98.0	86.1/98.5	91.2/98.1	94.0/97.3	92.1/98.8	**98.0**/99.4	95.9/99.3	94.3/**99.8**

D MAGE PRIMITIVE LIBRARIES

In this work, MAGE individuals use typed graph programs composed from four primitive libraries: intensity-image, binary-mask, segmentation, and scalar operations. The implementation uses Julia's type system to enforce valid function signatures during graph construction and mutation. A node is valid only when the output types of its input connections match the argument types required by the primitive assigned to that node. This ensures that all decoded individuals are type-correct executable programs.

Figure 4 illustrates the same typed multimodal mechanism in a different domain taken from (Torre et al., 2024). In the present work, the same principle is applied to image-processing primitives rather than strings and vectors.

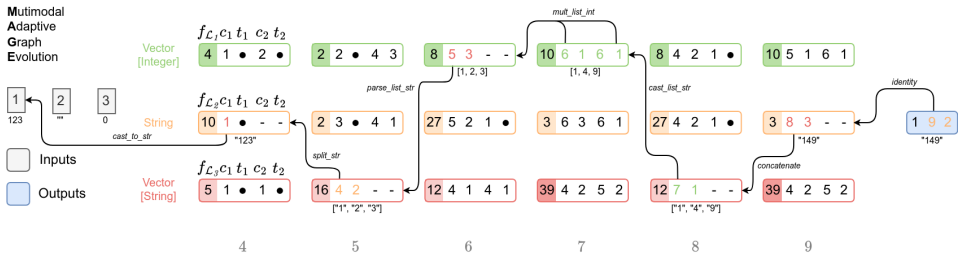


Figure 4: Example of a typed multimodal MAGE program taken from (Torre et al., 2024). The example shows a valid computational graph combining several data types; in this paper, MAGE uses the same type-valid graph mechanism for image, mask, segmentation, and scalar primitives.

Tables 18 to 21 list the primitives available in the four libraries, together with the descriptions provided to the LLM judges for interpretability scoring.

Table 18: Primitive functions and descriptions for Library 1. The functions in this library always return images of the same size in which pixels are values between 0 and 1.

Function	Description
identity_image2D	Returns the input image unchanged.
ones_2D	Creates an image filled with ones, matching input shape.
zeros_2D	Creates an image filled with zeros, matching input shape.
experimental_invert_2D	Inverts image intensities with respect to the [0, 1] range.
experimental_normalize_2D	Normalizes image values to the [0, 1] range.
experimental_standardize_2D	Standardizes image values using mean and standard deviation.
experimental_tointensity_image2D	Converts binary or segment images to intensity image representation.
erosion_2D	Applies morphological erosion to shrink bright regions.
dilation_2D	Applies morphological dilation to expand bright regions.
opening_2D	Applies opening (erosion followed by dilation) to remove small bright noise.
closing_2D	Applies closing (dilation followed by erosion) to fill small dark gaps.
tophat_2D	Computes top-hat transform to extract small bright structures.
bothat_2D	Computes (black) tophat transform to extract small dark structures.
morphogradient_2D	Computes morphological gradient to emphasize object boundaries.
morpholaplace_2D	Computes a morphological Laplace-style edge response.
subtract_img2D	Subtracts two images elementwise.
add_img2D	Adds two images elementwise.
mult_img2D	Multiplies two images elementwise.
max_img2D	Computes elementwise maximum between two images.
min_img2D	Computes elementwise minimum between two images.
bsubtract_image2D	Subtracts a scalar from every pixel of the image.
badd_image2D	Adds a scalar to every pixel of the image.
bmult_image2D	Multiplies every pixel of the image by a scalar.
sobelx_image2D	Applies Sobel X derivative filter.
sobely_image2D	Applies Sobel Y derivative filter.
sobelm_image2D	Computes Sobel gradient magnitude response.
ando3x_image2D	Applies Ando 3x3 X derivative filter.
ando3y_image2D	Applies Ando 3x3 Y derivative filter.
ando3m_image2D	Computes Ando 3x3 gradient magnitude response.
ando4x_image2D	Applies Ando 4x4 X derivative filter.
ando4y_image2D	Applies Ando 4x4 Y derivative filter.

Continued on next page

Function	Description
ando4m_image2D	Computes Ando 4x4 gradient magnitude response.
ando5x_image2D	Applies Ando 5x5 X derivative filter.
ando5y_image2D	Applies Ando 5x5 Y derivative filter.
ando5m_image2D	Computes Ando 5x5 gradient magnitude response.
bickleyx_image2D	Applies Bickley X derivative filter.
bickleyy_image2D	Applies Bickley Y derivative filter.
bickleym_image2D	Computes Bickley gradient magnitude response.
prewittx_image2D	Applies Prewitt X derivative filter.
prewitty_image2D	Applies Prewitt Y derivative filter.
prewittm_image2D	Computes Prewitt gradient magnitude response.
scharrx_image2D	Applies Scharr X derivative filter.
scharry_image2D	Applies Scharr Y derivative filter.
scharrm_image2D	Computes Scharr gradient magnitude response.
gaussian5_image2D	Applies Gaussian smoothing with sigma 1 : 5x5.
gaussian9_image2D	Applies Gaussian smoothing with sigma 2 : 9x9.
gaussian13_image2D	Applies Gaussian smoothing with sigma 3 : 13x13.
gaussian17_image2D	Applies Gaussian smoothing with sigma 4 : 17x17.
gaussian25_image2D	Applies Gaussian smoothing with sigma 6 : 25x25.
laplacian3_image2D	Applies Laplacian filter for second-order edge response.
dog_image2D	Applies Difference-of-Gaussians filtering.
moffat5_image2D	Applies Moffat smoothing filter with kernel size 5x5.
moffat13_image2D	Applies Moffat smoothing filter with kernel size 13x13.
moffat25_image2D	Applies Moffat smoothing filter with kernel size 25x25.
exp_image2D	Applies exponential transform to each pixel.
loginv_image2D	Applies logarithm to intensity pixels and inverses sign.
log_image2D	Applies logarithm to intensity pixels and normalizes.
powerof_image2D	Raises each pixel value to a provided exponent.
if_else_multiplexer	Selects between two same-type values based on whether cond is greater than zero.
avgpool_blocks	Pools each non-overlapping block by mean and broadcasts that value within the block.
avgpool_cross_blocks	Pools each non-overlapping block by averaging its center row and column.
maxpool_blocks	Pools each non-overlapping block by maximum and broadcasts that value within the block.
maxpool_cross_blocks	Pools each non-overlapping block by max over its center row and column.
minpool_blocks	Pools each non-overlapping block by minimum and broadcasts that value within the block.
minpool_cross_blocks	Pools each non-overlapping block by min over its center row and column.
meanpool	Applies sliding-window mean pooling and resizes back to the original image size.
maxpool	Applies sliding-window max pooling and resizes back to the original image size.
minpool	Applies sliding-window min pooling and resizes back to the original image size.
stdpool	Applies sliding-window standard-deviation pooling and resizes back to the original image size.
medianpool	Applies sliding-window median pooling and resizes back to the original image size.
uniquecountpool	Applies sliding-window unique-count pooling and rescales values to [0, 1].
argmaxcountpool	Applies sliding-window argmax-count pooling and rescales values to [0, 1].
argmincountpool	Applies sliding-window argmin-count pooling and rescales values to [0, 1].
iqrpool	Applies sliding-window interquartile-range pooling and resizes back to the original image size.
grad_magnitude	Computes gradient magnitude map of the input image.
grad_orientation	Computes gradient orientation map of the input image.
orientation_select	Subsets image gradients near a target orientation with configurable bandwidth.

Table 19: Primitive functions and descriptions for Library 2. The functions in this library always return images of the same size in which pixels are booleans (0 or 1).

Function	Description
identity_image2D	Returns the input image unchanged.
ones_2D	Creates an image filled with ones, matching input shape.
zeros_2D	Creates an image filled with zeros, matching input shape.
experimental_invert_2D	Inverts binary image.
tobinary_image2D	Converts an intensity image to binary using default thresholding.
tobinary_th_image2D_factory	Converts an intensity image to binary using a provided threshold.
subtract_img2D	Subtracts two images elementwise and rounds.
add_img2D	Adds two images elementwise and rounds.
mult_img2D	Multiplies two images elementwise and rounds.
max_img2D	Computes elementwise maximum between two images and rounds.
min_img2D	Computes elementwise minimum between two images and rounds.
binarize_adaptive2D	Binarizes image with adaptive local thresholding.
binarize_niblack2D	Binarizes image using Niblack local thresholding.
binarize_polysegment2D	Binarizes image using polysegment thresholding.
binarize_sauvola2D	Binarizes image using Sauvola local thresholding.
binarize_otsu2D	Binarizes image using Otsu histogram threshold.
binarize_minimumintermodes2D	Binarizes image using minimum-intermodes histogram threshold.
binarize_intermodes2D	Binarizes image using intermodes histogram threshold.
binarize_minimumerror2D	Binarizes image using minimum-error threshold estimation.
binarize_moments2D	Binarizes image using moments-based histogram threshold.
binarize_unimodalrosin2D	Binarizes image using unimodal Rosin threshold.
binarize_entropy2D	Binarizes image using entropy-based thresholding.
binarize_balanced2D	Binarizes image using balanced histogram thresholding.

Continued on next page

Function	Description
binarize_yen2D	Binarizes image using Yen thresholding method.
binarize_manual2D	Binarizes image using threshold parameters.
sobelx_image2D	Applies Sobel X derivative filter and rounds.
sobely_image2D	Applies Sobel Y derivative filter and rounds.
sobelm_image2D	Computes Sobel gradient magnitude response and rounds.
ando3x_image2D	Applies Ando 3x3 X derivative filter and rounds.
ando3y_image2D	Applies Ando 3x3 Y derivative filter and rounds.
ando3m_image2D	Computes Ando 3x3 gradient magnitude response and rounds.
ando4x_image2D	Applies Ando 4x4 X derivative filter and rounds.
ando4y_image2D	Applies Ando 4x4 Y derivative filter and rounds.
ando4m_image2D	Computes Ando 4x4 gradient magnitude response and rounds.
ando5x_image2D	Applies Ando 5x5 X derivative filter and rounds.
ando5y_image2D	Applies Ando 5x5 Y derivative filter and rounds.
ando5m_image2D	Computes Ando 5x5 gradient magnitude response and rounds.
bickleyx_image2D	Applies Bickley X derivative filter and rounds.
bickleyy_image2D	Applies Bickley Y derivative filter and rounds.
bickleym_image2D	Computes Bickley gradient magnitude response and rounds.
prewittx_image2D	Applies Prewitt X derivative filter and rounds.
prewitty_image2D	Applies Prewitt Y derivative filter and rounds.
prewittm_image2D	Computes Prewitt gradient magnitude response and rounds.
scharrx_image2D	Applies Scharr X derivative filter and rounds.
scharry_image2D	Applies Scharr Y derivative filter and rounds.
scharrm_image2D	Computes Scharr gradient magnitude response and rounds.
gaussian5_image2D	Applies Gaussian smoothing with sigma 1 : 5x5 and rounds.
gaussian9_image2D	Applies Gaussian smoothing with sigma 2 : 9x9 and rounds.
gaussian13_image2D	Applies Gaussian smoothing with sigma 3 : 13x13 and rounds.
gaussian17_image2D	Applies Gaussian smoothing with sigma 4 : 17x17 and rounds.
gaussian25_image2D	Applies Gaussian smoothing with sigma 6 : 25x25 and rounds.
laplacian3_image2D	Applies Laplacian filter for second-order edge response and rounds.
dog_image2D	Applies Difference-of-Gaussians filtering and rounds.
moffat5_image2D	Applies Moffat smoothing filter with kernel size 5x5 and rounds.
moffat13_image2D	Applies Moffat smoothing filter with kernel size 13x13 and rounds.
moffat25_image2D	Applies Moffat smoothing filter with kernel size 25x25 and rounds.
findlocalminima_image2D	Marks local minima locations in the image.
findlocalmaxima_image2D	Marks local maxima locations in the image.
erosion_2D	Applies morphological erosion to shrink bright regions.
dilation_2D	Applies morphological dilation to expand bright regions.
opening_2D	Applies opening (erosion followed by dilation) to remove small bright noise.
closing_2D	Applies closing (dilation followed by erosion) to fill small dark gaps.
tophat_2D	Computes top-hat transform to extract small bright structures.
bothat_2D	Computes (black) tophat transform to extract small dark structures.
morphogradient_2D	Computes morphological gradient to emphasize object boundaries.
morpholaplace_2D	Computes a morphological Laplace-style edge response.
if_else_multiplexer	Selects between two same-type values based on whether cond is greater than zero.
avgpool_blocks	Pools each non-overlapping block by mean and broadcasts that value within the block and rounds.
avgpool_cross_blocks	Pools each non-overlapping block by averaging its center row and column and rounds.
maxpool_blocks	Pools each non-overlapping block by maximum and broadcasts that value within the block and rounds.
maxpool_cross_blocks	Pools each non-overlapping block by max over its center row and column and rounds.
minpool_blocks	Pools each non-overlapping block by minimum and broadcasts that value within the block and rounds.
minpool_cross_blocks	Pools each non-overlapping block by min over its center row and column and rounds.
meanpool	Applies sliding-window mean pooling and resizes back to the original image size and rounds.
maxpool	Applies sliding-window max pooling and resizes back to the original image size and rounds.
minpool	Applies sliding-window min pooling and resizes back to the original image size and rounds.
stdpool	Applies sliding-window standard-deviation pooling and resizes back to the original image size and rounds.
medianpool	Applies sliding-window median pooling and resizes back to the original image size and rounds.
uniquecountpool	Applies sliding-window unique-count pooling and rescales values to [0, 1] and rounds.
argmaxcountpool	Applies sliding-window argmax-count pooling and rescales values to [0, 1] and rounds.
argmincountpool	Applies sliding-window argmin-count pooling and rescales values to [0, 1] and rounds.
iqrpool	Applies sliding-window interquartile-range pooling and resizes back to the original image size and rounds.

Table 20: Primitive functions and descriptions for Library 3. The functions in this library always return images of the same size in which pixels are Integers (each integer correspond to a different segment).

Function	Description
identity_image2D	Returns the input image unchanged.
ones_2D	Creates an image filled with ones, matching input shape.
zeros_2D	Creates an image filled with zeros, matching input shape.
experimental_tosegment_image2D	Converts a binary image to segment image representation (separates background and foreground).

Continued on next page

Function	Description
fastscanning_image2D	Segments the image using a fast scanning region-labeling strategy.
watershed_image2D	Segments the image using watershed transform over intensity topology.

Table 21: Primitive functions and descriptions for Library 4. The functions in this library always return a scalar value.

Function	Description
identity_float	Returns the input Float64 value unchanged.
ret_1	Returns the constant float value 1.0.
tanh	Applies hyperbolic tangent to a numeric input.
relu	Applies ReLU by returning max(x, 0) for numeric inputs.
modulo	Computes the modulo remainder a % b for two numeric inputs.
is_eq_to	Returns 1 when two numeric inputs are equal, otherwise 0.
experimental_is_gt	Returns 1 when the first number is greater than the second, otherwise 0.
experimental_is_lt	Returns 1 when the first number is lower than the second, otherwise 0.
experimental_not	Returns 1 when the input is at most 0.5, otherwise 0.
number_sum	Adds two numeric inputs.
number_minus	Subtracts the second numeric input from the first.
number_mult	Multiplies two numeric inputs.
number_div	Divides the first numeric input by the second and throws on division by zero.
safe_div	Divides two numeric inputs and returns 0 when the divisor is zero.
power_of	Raises the first numeric input to the power of the second.
pi_	Returns the mathematical constant pi as Float64.
exp_	Computes the exponential of the numeric input.
log_	Computes the natural logarithm after clipping the input to a positive minimum.
log10_	Computes the base-10 logarithm after clipping the input to a positive minimum.
reduce_length	Returns the number of pixels in the input image.
reduce_biggestAxis	Returns the largest image dimension size.
reduce_smallerAxis	Returns the smallest image dimension size.
reduce_histMode	Returns the most frequent pixel value in the image.
reduce_histModeCount	Returns the count of the most frequent pixel value in the image.
reduce_propWhite	Returns the proportion of white pixels in a binary image.
reduce_propBlack	Returns the proportion of black pixels in a binary image.
reduce_nColors	Returns the number of unique pixel values in the image.
reduce_mean	Returns the mean intensity of the image.
reduce_median	Returns the median intensity of the image.
reduce_std	Returns the standard deviation of image intensities.
reduce_maximum	Returns the maximum intensity value in the image.
reduce_minimum	Returns the minimum intensity value in the image.
if_else_multiplexer	Selects between two same-type values based on whether cond is greater than zero.
region_mean	Computes local mean intensity inside a patch centered at normalized coordinates.
region_std	Computes local standard deviation inside a patch centered at normalized coordinates.
region_min	Computes the minimum local value inside a centered patch.
region_max	Computes the maximum local value inside a centered patch.
region_sum	Computes the local sum of values inside a centered patch.
region_median	Computes the local median value inside a centered patch.
region_range	Computes local range as max minus min inside a centered patch.
region_contrast	Computes local contrast between a center patch and its surrounding ring.
region_energy	Computes local energy as the mean squared value inside a centered patch.
region_entropy	Computes local entropy from a coarse histogram inside a centered patch.
region_mean_5p	Computes local mean intensity using a percentage-sized patch centered at normalized coordinates (5%).
region_std_5p	Computes local standard deviation using a percentage-sized patch centered at normalized coordinates (5%).
region_min_5p	Computes the local minimum using a percentage-sized patch centered at normalized coordinates (5%).
region_max_5p	Computes the local maximum using a percentage-sized patch centered at normalized coordinates (5%).
region_sum_5p	Computes local sum using a percentage-sized patch centered at normalized coordinates (5%).
region_median_5p	Computes local median using a percentage-sized patch centered at normalized coordinates (5%).
region_range_5p	Computes local range using a percentage-sized patch centered at normalized coordinates (5%).
region_contrast_5p	Computes center-versus-ring local contrast using a percentage-sized patch (5%).
region_energy_5p	Computes local energy using a percentage-sized patch centered at normalized coordinates (5%).
region_entropy_5p	Computes local entropy from histogram bins in a percentage-sized patch (5%).
region_mean_10p	Computes local mean intensity using a percentage-sized patch centered at normalized coordinates (10%).
region_std_10p	Computes local standard deviation using a percentage-sized patch centered at normalized coordinates (10%).
region_min_10p	Computes the local minimum using a percentage-sized patch centered at normalized coordinates (10%).
region_max_10p	Computes the local maximum using a percentage-sized patch centered at normalized coordinates (10%).
region_sum_10p	Computes local sum using a percentage-sized patch centered at normalized coordinates (10%).
region_median_10p	Computes local median using a percentage-sized patch centered at normalized coordinates (10%).

Continued on next page

Function	Description
region_range_10p	Computes local range using a percentage-sized patch centered at normalized coordinates (10%).
region_contrast_10p	Computes center-versus-ring local contrast using a percentage-sized patch (10%).
region_energy_10p	Computes local energy using a percentage-sized patch centered at normalized coordinates (10%).
region_entropy_10p	Computes local entropy from histogram bins in a percentage-sized patch (10%).
region_mean_20p	Computes local mean intensity using a percentage-sized patch centered at normalized coordinates (20%).
region_std_20p	Computes local standard deviation using a percentage-sized patch centered at normalized coordinates (20%).
region_min_20p	Computes the local minimum using a percentage-sized patch centered at normalized coordinates (20%).
region_max_20p	Computes the local maximum using a percentage-sized patch centered at normalized coordinates (20%).
region_sum_20p	Computes local sum using a percentage-sized patch centered at normalized coordinates (20%).
region_median_20p	Computes local median using a percentage-sized patch centered at normalized coordinates (20%).
region_range_20p	Computes local range using a percentage-sized patch centered at normalized coordinates (20%).
region_contrast_20p	Computes center-versus-ring local contrast using a percentage-sized patch (20%).
region_energy_20p	Computes local energy using a percentage-sized patch centered at normalized coordinates (20%).
region_entropy_20p	Computes local entropy from histogram bins in a percentage-sized patch (20%).
haar_lr	Computes left-versus-right Haar contrast in a local image region.
haar_tb	Computes top-versus-bottom Haar contrast in a local image region.
haar_diag_main	Computes diagonal Haar contrast favoring main-diagonal quadrants.
haar_diag_anti	Computes diagonal Haar contrast favoring anti-diagonal quadrants.
haar_center_surround	Computes center-versus-surround Haar contrast in a local region.
haar_three_h	Computes horizontal three-band Haar contrast in a local region.
haar_three_v	Computes vertical three-band Haar contrast in a local region.
orientation_coherence	Computes orientation coherence of the image gradient field.
dominant_orientation	Returns the strongest coarse orientation bin center among $\{0, \pi/4, \pi/2, 3\pi/4\}$, normalized by π.
orientation_energy_0	Returns orientation energy proportion near 0 degrees.
orientation_energy_45	Returns orientation energy proportion near 45 degrees.
orientation_energy_90	Returns orientation energy proportion near 90 degrees.
orientation_energy_135	Returns orientation energy proportion near 135 degrees.
orientation_spread	Measures how widely orientation energy is distributed across directions.
glcm_glcm_mean_ref_mean	Computes glcm mean metric across angles [pi, 3pi/4, pi/2, pi/4], then applies mean reducer.
glcm_glcm_mean_ref_sum	Computes glcm mean metric across angles [pi, 3pi/4, pi/2, pi/4], then applies sum reducer.
glcm_glcm_mean_ref_std	Computes glcm mean metric across angles [pi, 3pi/4, pi/2, pi/4], then applies standard deviation reducer.
glcm_glcm_mean_ref_minimum	Computes glcm mean metric across angles [pi, 3pi/4, pi/2, pi/4], then applies minimum reducer.
glcm_glcm_mean_ref_maximum	Computes glcm mean metric across angles [pi, 3pi/4, pi/2, pi/4], then applies maximum reducer.
glcm_glcm_var_ref_mean	Computes glcm var metric across angles [pi, 3pi/4, pi/2, pi/4], then applies mean reducer.
glcm_glcm_var_ref_sum	Computes glcm var metric across angles [pi, 3pi/4, pi/2, pi/4], then applies sum reducer.
glcm_glcm_var_ref_std	Computes glcm var metric across angles [pi, 3pi/4, pi/2, pi/4], then applies standard deviation reducer.
glcm_glcm_var_ref_minimum	Computes glcm var metric across angles [pi, 3pi/4, pi/2, pi/4], then applies minimum reducer.
glcm_glcm_var_ref_maximum	Computes glcm var metric across angles [pi, 3pi/4, pi/2, pi/4], then applies maximum reducer.
glcm_correlation_mean	Computes GLCM correlation across angles [pi, 3pi/4, pi/2, pi/4], then applies mean reducer.
glcm_correlation_sum	Computes GLCM correlation across angles [pi, 3pi/4, pi/2, pi/4], then applies sum reducer.
glcm_correlation_std	Computes GLCM correlation across angles [pi, 3pi/4, pi/2, pi/4], then applies standard deviation reducer.
glcm_correlation_minimum	Computes GLCM correlation across angles [pi, 3pi/4, pi/2, pi/4], then applies minimum reducer.
glcm_correlation_maximum	Computes GLCM correlation across angles [pi, 3pi/4, pi/2, pi/4], then applies maximum reducer.
glcm_contrast_mean	Computes GLCM contrast across angles [pi, 3pi/4, pi/2, pi/4], then applies mean reducer.
glcm_contrast_sum	Computes GLCM contrast across angles [pi, 3pi/4, pi/2, pi/4], then applies sum reducer.
glcm_contrast_std	Computes GLCM contrast across angles [pi, 3pi/4, pi/2, pi/4], then applies standard deviation reducer.
glcm_contrast_minimum	Computes GLCM contrast across angles [pi, 3pi/4, pi/2, pi/4], then applies minimum reducer.
glcm_contrast_maximum	Computes GLCM contrast across angles [pi, 3pi/4, pi/2, pi/4], then applies maximum reducer.
glcm_IDM_mean	Computes GLCM inverse-difference-moment across angles [pi, 3pi/4, pi/2, pi/4], then applies mean reducer.
glcm_IDM_sum	Computes GLCM inverse-difference-moment across angles [pi, 3pi/4, pi/2, pi/4], then applies sum reducer.
glcm_IDM_std	Computes GLCM inverse-difference-moment across angles [pi, 3pi/4, pi/2, pi/4], then applies standard deviation reducer.
glcm_IDM_minimum	Computes GLCM inverse-difference-moment across angles [pi, 3pi/4, pi/2, pi/4], then applies minimum reducer.
glcm_IDM_maximum	Computes GLCM inverse-difference-moment across angles [pi, 3pi/4, pi/2, pi/4], then applies maximum reducer.
glcm_ASM_mean	Computes GLCM angular-second-moment across angles [pi, 3pi/4, pi/2, pi/4], then applies mean reducer.
glcm_ASM_sum	Computes GLCM angular-second-moment across angles [pi, 3pi/4, pi/2, pi/4], then applies sum reducer.
glcm_ASM_std	Computes GLCM angular-second-moment across angles [pi, 3pi/4, pi/2, pi/4], then applies standard deviation reducer.
glcm_ASM_minimum	Computes GLCM angular-second-moment across angles [pi, 3pi/4, pi/2, pi/4], then applies minimum reducer.
glcm_ASM_maximum	Computes GLCM angular-second-moment across angles [pi, 3pi/4, pi/2, pi/4], then applies maximum reducer.
glcm_glcm_entropy_mean	Computes GLCM entropy across angles [pi, 3pi/4, pi/2, pi/4], then applies mean reducer.

Continued on next page

Function	Description
glcm_glcm_entropy_sum	Computes GLCM entropy across angles [pi, 3pi/4, pi/2, pi/4], then applies sum reducer.
glcm_glcm_entropy_std	Computes GLCM entropy across angles [pi, 3pi/4, pi/2, pi/4], then applies standard deviation reducer.
glcm_glcm_entropy_minimum	Computes GLCM entropy across angles [pi, 3pi/4, pi/2, pi/4], then applies minimum reducer.
glcm_glcm_entropy_maximum	Computes GLCM entropy across angles [pi, 3pi/4, pi/2, pi/4], then applies maximum reducer.
glcm_max_prob_mean	Computes GLCM max-probability across angles [pi, 3pi/4, pi/2, pi/4], then applies mean reducer.
glcm_max_prob_sum	Computes GLCM max-probability across angles [pi, 3pi/4, pi/2, pi/4], then applies sum reducer.
glcm_max_prob_std	Computes GLCM max-probability across angles [pi, 3pi/4, pi/2, pi/4], then applies standard deviation reducer.
glcm_max_prob_minimum	Computes GLCM max-probability across angles [pi, 3pi/4, pi/2, pi/4], then applies minimum reducer.
glcm_max_prob_maximum	Computes GLCM max-probability across angles [pi, 3pi/4, pi/2, pi/4], then applies maximum reducer.
glcm_energy_mean	Computes GLCM energy across angles [pi, 3pi/4, pi/2, pi/4], then applies mean reducer.
glcm_energy_sum	Computes GLCM energy across angles [pi, 3pi/4, pi/2, pi/4], then applies sum reducer.
glcm_energy_std	Computes GLCM energy across angles [pi, 3pi/4, pi/2, pi/4], then applies standard deviation reducer.
glcm_energy_minimum	Computes GLCM energy across angles [pi, 3pi/4, pi/2, pi/4], then applies minimum reducer.
glcm_energy_maximum	Computes GLCM energy across angles [pi, 3pi/4, pi/2, pi/4], then applies maximum reducer.
glcm_dissimilarity_mean	Computes GLCM dissimilarity across angles [pi, 3pi/4, pi/2, pi/4], then applies mean reducer.
glcm_dissimilarity_sum	Computes GLCM dissimilarity across angles [pi, 3pi/4, pi/2, pi/4], then applies sum reducer.
glcm_dissimilarity_std	Computes GLCM dissimilarity across angles [pi, 3pi/4, pi/2, pi/4], then applies standard deviation reducer.
glcm_dissimilarity_minimum	Computes GLCM dissimilarity across angles [pi, 3pi/4, pi/2, pi/4], then applies minimum reducer.
glcm_dissimilarity_maximum	Computes GLCM dissimilarity across angles [pi, 3pi/4, pi/2, pi/4], then applies maximum reducer.

E LLM PROMPT TEMPLATES

We use three LLM judges for interpretability scoring: GPT-5 from OpenAI, Claude Sonnet 4.6 from Anthropic, and Gemini 3.1 Pro Preview from Google, which we refer to as ChatGPT, Claude and Gemini. All models are accessed through their respective APIs. Using multiple judge models helps reduce the influence of biases specific to any single model.

Prompts are assembled modularly. Each query contains exactly one audience definition, exactly one scoring-task template, and the function information required for that task. In the unigram setting, the function information is the table of MAGE primitives and their descriptions (Tables 18 to 21). In the bi-gram setting, the same primitive descriptions are provided, together with the empirical list of local function compositions to be scored. This design keeps the audience perspective fixed within a query.

E.1 AUDIENCE DEFINITIONS

Computer Scientist

You are evaluating interpretability from the perspective of a computer scientist or machine learning researcher. This stakeholder understands algorithms, mathematical transformations, computer vision operations, and program structure. They may understand filters, morphology, segmentation, scalar transformations, feature extraction, and classifier pipelines. They do not necessarily have medical domain expertise.

Doctor

You are evaluating interpretability from the perspective of a medical doctor using an image-classification system. This stakeholder has clinical and biomedical knowledge, and may understand visual evidence such as tissue structure, cell morphology, boundaries, contrast, lesions, organ shape, or abnormal regions. They are not assumed to have expertise in machine learning, genetic programming, or low-level computer vision implementation details.

Patient

You are evaluating interpretability from the perspective of a patient or non-expert affected by a medical image-classification decision. This stakeholder is not assumed to know medicine,

machine learning, or computer vision. They may understand simple visual ideas such as brightness, darkness, shape, size, edges, regions, or comparing parts of an image.

E.2 SCORING TASK

Unigram scoring

You will score the interpretability of symbolic program primitives from the perspective of the stakeholder defined above.
You are given:
1. A library identifier.
2. A table of primitive function names and descriptions.
Each primitive is a function that may appear inside a symbolic program. Score each primitive independently, using its name and description.

Scoring
Score from 1 to 10, where:
1 = not interpretable at all for this stakeholder;
5 = moderately interpretable, but requiring effort or background assumptions;
10 = very interpretable, clear and easy for this stakeholder to understand.
Use integers only.
Score interpretability, not predictive usefulness, clinical correctness, or whether the primitive is likely to improve classification.

Output
Return only valid JSON. Do not include markdown, comments, or explanatory text outside the JSON.
Use this exact schema:
[{ "library_id": "<library identifier>", "scores": [{ "unit": "<unit exactly as provided>", "score": <integer from 1 to 10>, "rationale": "<one short sentence>" }] }, { "library_id": "<library identifier>", "scores": [{ "unit": "<unit exactly as provided>", "score": <integer from 1 to 10>, "rationale": "<one short sentence>" }] }]
Return one score for every primitive in the provided table. Preserve the exact function names. Do not omit primitives. Do not add primitives that were not provided.

Bigrams

You will score the interpretability of symbolic program units from the perspective of the stakeholder defined above.
You are given:
1. A library identifier.
2. A table of primitive function names and descriptions.
3. A list of symbolic units extracted empirically from active programs.
Each symbolic unit is either:
- a standalone primitive, such as ‘lib4::reduce_std‘, meaning the primitive (from library 4) is applied directly; or
- a local composition, such as ‘lib4::region_mean(lib1::ando5x_image2D, lib4::reduce_biggestAxis, lib4::region_median)‘, meaning the caller function ‘region_mean‘ (from library 4) receives inputs produced by the listed functions.
Use the primitive descriptions to infer what each unit does. For a composition, score the interpretability of the local composed operation as a whole, not only the caller function.

Scoring
Score from 1 to 10, where:
1 = not interpretable at all for this stakeholder;
5 = moderately interpretable, but requiring effort or background assumptions;

10 = very interpretable, clear and easy for this stakeholder to understand.
Use integers only.
Score interpretability, not predictive usefulness, clinical correctness, or whether the unit is likely to improve classification.

Output
Return only valid JSON. Do not include markdown, comments, or explanatory text outside the JSON.
Use this exact schema:
[{ "library_id": "<library identifier>", "scores": [{ "unit": "<unit exactly as provided (primitive or composition)>", "score": <integer from 1 to 10>, "rationale": "<one short sentence>" }] }, { "library_id": "<library identifier>", "scores": [{ "unit": "<unit exactly as provided (primitive or composition)>", "score": <integer from 1 to 10>, "rationale": "<one short sentence>" }] }]
Return one score for every unit in the provided unit list. Preserve the exact unit string. Do not omit units. Do not add units that were not provided.

For unigram scoring, each query contained one audience definition, the unigram scoring template, and the full set of primitive descriptions. This fit within the context window for all judge models. For bi-gram scoring, the empirical composition inventory was larger, so we split the task by primitive library: each query still received the audience definition, the bi-gram scoring template, and the descriptions for all four libraries, but only the bi-gram units associated with one library were scored at a time.

We verified the completeness of the LLM annotations by comparing every raw scoring file against the canonical unigram and bi-gram inventories. The expected inventories contained 302 unigram units and 1225 bi-gram units. Coverage was essentially complete. No model–audience group had any unit that was absent across all three repeated runs. The only unigram omission was one missing unit in a single ChatGPT/Computer Scientist run. For bi-grams, omissions were rare and occurred only at the seed level: Claude missed 3 units in total, Gemini missed 2, and ChatGPT missed 12, with the largest single omission being 6 units in one ChatGPT/Computer Scientist run. These were therefore occasional incomplete generations rather than systematic gaps. When a unit was missing from one repeated run, its aggregate score was computed from the available runs only; no missing scores were imputed. Minor formatting inconsistencies, such as variant library names, were corrected manually; no scores were added, modified, or deleted.

E.3 Example LLM Output for Unigram Scoring.

This section shows representative unigram interpretability scores produced by Gemini 3.1 Pro Preview for the three stakeholder groups. The examples illustrate how the same primitive functions can receive different scores depending on the assumed audience: computer scientist, doctor, or patient.

E.3.1 Computer Scientist perspective


```
[
  {
    "library_id": "Library 1",
    "scores": [
      {
        "unit": "identity_image2D",
        "score": 10,
        "rationale": "Returns the exact input, trivially understandable."
      },
      {
        "unit": "ones_2D",
        "score": 10,
        "rationale": "Standard matrix initialization with ones."
      },
      {
        "unit": "zeros_2D",
        "score": 10,
        "rationale": "Standard matrix initialization with zeros."
      },
      {
        "unit": "experimental_invert_2D",
```

```
      "score": 10,
      "rationale": "Simple logical inversion of normalized intensities."
    },
    {
      "unit": "experimental_normalize_2D",
      "score": 10,
      "rationale": "Standard min-max normalization mapping values to [0, 1]."
    },
    {
      "unit": "experimental_standardize_2D",
      "score": 10,
      "rationale": "Standard z-score standardization."
    },
    {
      "unit": "experimental_tointensity_image2D",
      "score": 9,
      "rationale": "Straightforward casting of labels/masks to intensity images."
    },
    {
      "unit": "erosion_2D",
      "score": 10,
      "rationale": "Fundamental mathematical morphology operation."
    },
    ...
```


E.3.2 DOCTOR PERSPECTIVE


```
[
  {
    "library_id": "Library 1",
    "scores": [
      {
        "unit": "identity_image2D",
        "score": 10,
        "rationale": "Returning the image unchanged is a trivial and fully interpretable
            operation."
      },
      {
        "unit": "ones_2D",
        "score": 9,
        "rationale": "An all-white image is easily understood as a uniform blank canvas."
      },
      {
        "unit": "zeros_2D",
        "score": 9,
        "rationale": "An all-black image is easily understood as an empty mask."
      },
      {
        "unit": "experimental_invert_2D",
        "score": 10,
        "rationale": "Inverting an image is identical to a photographic negative, which is
            highly familiar in medicine."
      },
      {
        "unit": "experimental_normalize_2D",
        "score": 8,
        "rationale": "Adjusting values to a standard range is a basic concept akin to
            standardizing contrast."
      },
      {
        "unit": "experimental_standardize_2D",
        "score": 8,
        "rationale": "Using mean and standard deviation for standardization is a basic
            statistical concept familiar to doctors."
      },
      {
        "unit": "experimental_tointensity_image2D",
        "score": 8,
        "rationale": "Converting a mask to a grayscale image is conceptually straightforward."
      },
      {
        "unit": "erosion_2D",
        "score": 9,
        "rationale": "Shrinking bright regions is easily visualizable as making white
            structures like calcifications smaller."
      },
      ...
```

E.3.3 PATIENT PERSPECTIVE


```
[
  {
    "library_id": "Library 1",
    "scores": [
      {
        "unit": "identity_image2D",
        "score": 10,
        "rationale": "Returning the input unchanged is completely intuitive."
      },
      {
        "unit": "ones_2D",
        "score": 8,
        "rationale": "Creating an image of uniform value is understandable, though 'ones'
            requires basic awareness of numerical pixel representation."
      },
      {
        "unit": "zeros_2D",
        "score": 8,
        "rationale": "Creating a black or blank image is understandable, similar to ones_2D."
      },
      {
        "unit": "experimental_invert_2D",
        "score": 9,
        "rationale": "Inverting an image is a familiar concept akin to a photographic negative
            ."
      },
      {
        "unit": "experimental_normalize_2D",
        "score": 4,
        "rationale": "The term 'normalize' is mathematical and non-experts may not immediately
             grasp what it visually alters."
      },
      {
        "unit": "experimental_standardize_2D",
        "score": 3,
        "rationale": "Standard deviation and mean are statistical concepts that require
            background knowledge to visualize."
      },
      {
        "unit": "experimental_tointensity_image2D",
        "score": 4,
        "rationale": "Converting to 'intensity representation' is somewhat technical and
            confusing without knowing image data types."
      },
      {
        "unit": "erosion_2D",
        "score": 8,
        "rationale": "The description 'shrink bright regions' is very visual and easy to
            understand."
      },
      ...
```


E.4 EXAMPLE LLM OUTPUT FOR BIGRAMS SCORING.

This section shows representative bigram interpretability scores produced by Gemini 3.1 Pro Preview for the three stakeholder groups.

E.4.1 COMPUTER SCIENTIST PERSPECTIVE


```
[
  {
    "library_id": "Library 1",
    "scores": [
      {
        "unit": "lib1::add_img2D",
        "score": 10,
        "rationale": "Direct application of a standard image processing primitive is fully
            interpretable."
      },
      {
        "unit": "lib1::add_img2D(lib1::add_img2D)",
        "score": 9,
        "rationale": "Sequentially applying or composing standard image filters and arithmetic
             is highly interpretable."
```

```
    },
    {
      "unit": "lib1::add_img2D(lib1::add_img2D, lib1::add_img2D)",
      "score": 9,
      "rationale": "Sequentially applying or composing standard image filters and arithmetic
          is highly interpretable."
    },
    {
      "unit": "lib1::add_img2D(lib1::add_img2D, lib1::ando5m_image2D)",
      "score": 9,
      "rationale": "Sequentially applying or composing standard image filters and arithmetic
          is highly interpretable."
    },
    {
      "unit": "lib1::add_img2D(lib1::ando4m_image2D, lib1::minpool)",
      "score": 9,
      "rationale": "Sequentially applying or composing standard image filters and arithmetic
          is highly interpretable."
    },
    {
      "unit": "lib1::add_img2D(lib1::avgpool_blocks, lib1::loginv_image2D)",
      "score": 9,
      "rationale": "Sequentially applying or composing standard image filters and arithmetic
          is highly interpretable."
    },
    {
      "unit": "lib1::add_img2D(lib1::gaussian25_image2D)",
      "score": 9,
      "rationale": "Sequentially applying or composing standard image filters and arithmetic
          is highly interpretable."
    },
    {
      "unit": "lib1::add_img2D(lib1::gaussian9_image2D)",
      "score": 9,
      "rationale": "Sequentially applying or composing standard image filters and arithmetic
          is highly interpretable."
    },
    ...
```


E.4.2 DOCTOR PERSPECTIVE


```
[
  {
    "library_id": "Library 1",
    "scores": [
      {
        "unit": "lib1::add_img2D",
        "score": 6,
        "rationale": "Elementwise addition provides a fairly intuitive way to combine visual
            features, though the specific biological meaning depends on the inputs."
      },
      {
        "unit": "lib1::add_img2D(lib1::add_img2D)",
        "score": 5,
        "rationale": "Chaining additions is moderately understandable as blending multiple
            image layers, but the clinical interpretation becomes murky."
      },
      {
        "unit": "lib1::add_img2D(lib1::add_img2D, lib1::add_img2D)",
        "score": 4,
        "rationale": "Adding multiple blended images together creates an overly complex
            composite that is hard to relate to underlying tissue morphology."
      },
      {
        "unit": "lib1::add_img2D(lib1::add_img2D, lib1::ando5m_image2D)",
        "score": 2,
        "rationale": "Combining basic image sums with an obscure Ando gradient magnitude
            filter lacks any clear clinical intuition."
      },
      {
        "unit": "lib1::add_img2D(lib1::ando4m_image2D, lib1::minpool)",
        "score": 2,
        "rationale": "Adding a highly specific computer vision edge map to a min-pooled image
            is technically abstract and uninterpretable to a doctor."
      },
      {
        "unit": "lib1::add_img2D(lib1::avgpool_blocks, lib1::loginv_image2D)",
        "score": 3,
```

```
      "rationale": "Blending a block-averaged image with a logarithmically inverted one is a
           mathematical abstraction detached from physical anatomy."
    },
    {
      "unit": "lib1::add_img2D(lib1::gaussian25_image2D)",
      "score": 6,
      "rationale": "Adding a heavily smoothed image acts like adding a blurred background or
           base intensity, which is moderately easy to visualize."
    },
    {
      "unit": "lib1::add_img2D(lib1::gaussian9_image2D)",
      "score": 6,
      "rationale": "Adding a moderately smoothed image to another is a standard visual
          overlay that is mostly understandable."
    },
    ...
```


E.4.3 PATIENT PERSPECTIVE


```
[
  {
    "library_id": "Library 1",
    "scores": [
      {
        "unit": "lib1::add_img2D",
        "score": 8,
        "rationale": "Adding two images together is visually intuitive for a layperson to
            grasp."
      },
      {
        "unit": "lib1::add_img2D(lib1::add_img2D)",
        "score": 7,
        "rationale": "Combining images through multiple additions remains relatively easy to
            visualize."
      },
      {
        "unit": "lib1::add_img2D(lib1::add_img2D, lib1::add_img2D)",
        "score": 7,
        "rationale": "Summing multiple combinations of images is a basic composable visual
            idea."
      },
      {
        "unit": "lib1::add_img2D(lib1::add_img2D, lib1::ando5m_image2D)",
        "score": 3,
        "rationale": "Adding an image to a highly technical 'gradient magnitude' result is
            hard to interpret visually."
      },
      {
        "unit": "lib1::add_img2D(lib1::ando4m_image2D, lib1::minpool)",
        "score": 3,
        "rationale": "Combining technical gradient magnitudes and min-pooling is obscure for
            non-experts."
      },
      {
        "unit": "lib1::add_img2D(lib1::avgpool_blocks, lib1::loginv_image2D)",
        "score": 4,
        "rationale": "Adding block averages to a logarithmic inversion creates a visually
            confusing result."
      },
      {
        "unit": "lib1::add_img2D(lib1::gaussian25_image2D)",
        "score": 6,
        "rationale": "Adding an image to its blurred version can be understood as enhancing
            overall brightness."
      },
      {
        "unit": "lib1::add_img2D(lib1::gaussian9_image2D)",
        "score": 6,
        "rationale": "Combining an image with a blurred copy is a fairly relatable visual
            effect."
      },
      ...
```

F Interpretability Score Distributions by Audience

For each scored unit, we first average the three repeated scores within each LLM judge. The resulting distributions show a consistent audience effect: computer scientists receive the highest interpretability scores on average, followed by doctors and then patients. This ordering is expected because the primitive libraries contain many computer-vision and numerical operations that are more familiar to technical users than to clinical or non-expert audiences.

Figures 5 and 6 compare the distributions for unigram and bi-gram scoring. Function compositions generally receive lower scores than individual primitives, especially for the computer-scientist audience, indicating that even familiar operations become less interpretable when combined into local program patterns.

F.1 Primitive Functions

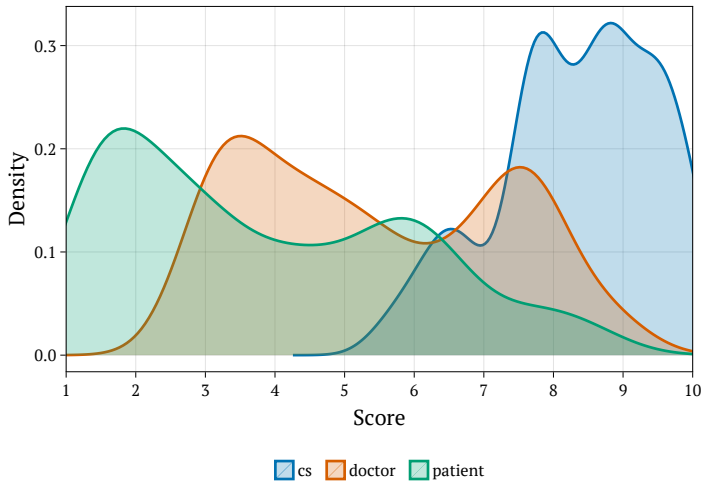


Figure 5: Distribution of unigram interpretability scores by stakeholder audience. Scores are averaged over repeated judgments within each LLM judge.

F.2 Function Compositions

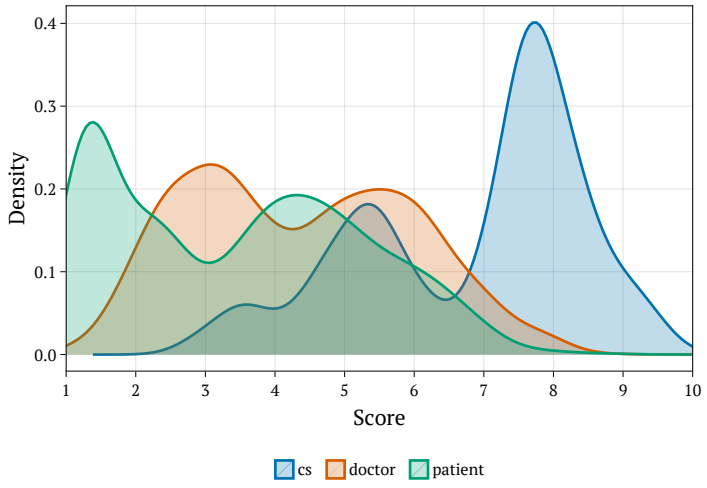


Figure 6: Distribution of bi-gram interpretability scores by stakeholder audience. Scores are averaged over repeated judgments within each LLM judge.

G Interpretability Score Distributions by LLM Judge

This section compares the interpretability-score distributions produced by each LLM judge for each stakeholder audience. Overall, the judges show similar broad trends, but their distributions are not identical: some are shifted toward higher or lower scores, and some assign scores with different modality. These differences should be interpreted cautiously because each judge was sampled only three times. For unigram scoring, the computer-scientist distributions are generally higher than the doctor and patient distributions, but differ in skew across judges. For doctors, ChatGPT produces a more concentrated distribution around intermediate scores, whereas Gemini and Claude show more separation between low- and high-interpretability functions. Patient scores are generally lower across judges. For bi-gram scoring, the same qualitative patterns remain, but the distributions shift toward lower scores, reflecting the increased complexity of interpreting local function compositions.

G.1 Primitive Functions

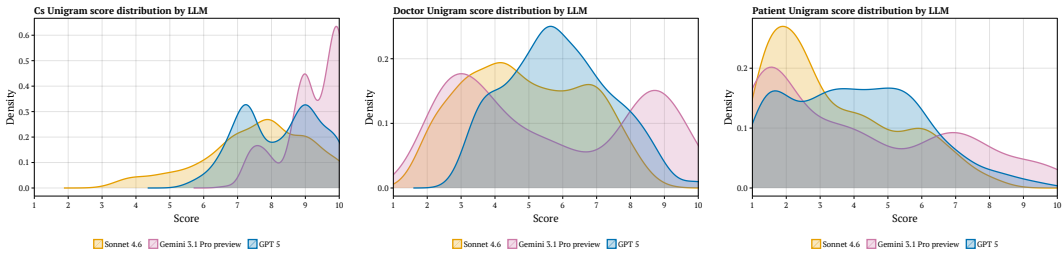


Figure 7: Unigram interpretability-score distributions by LLM judge and stakeholder audience. From left to right: computer scientist, doctor, and patient.

G.2 Function Compositions

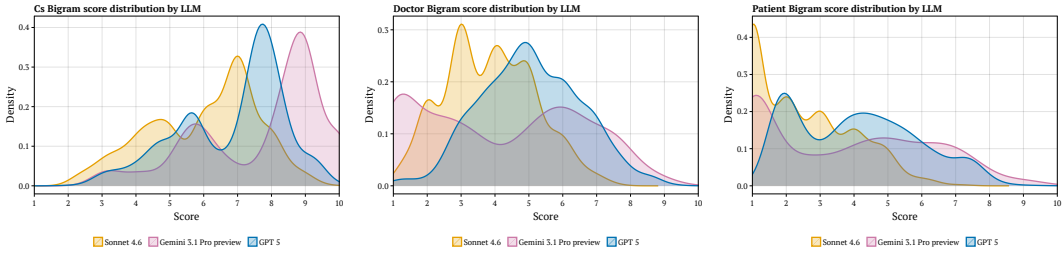


Figure 8: Bi-gram interpretability-score distributions by LLM judge and stakeholder audience. From left to right: computer scientist, doctor, and patient.

H Interpretability Score Shifts Across Audiences

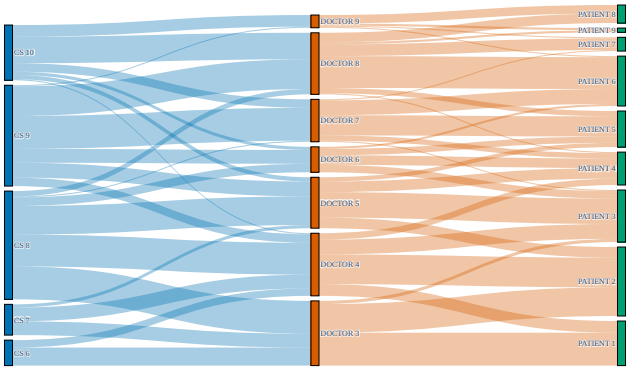


Figure 9: Sankey diagram showing how unigram interpretability scores shift across stakeholder audiences. Scores are averaged across LLM judges and repetitions.

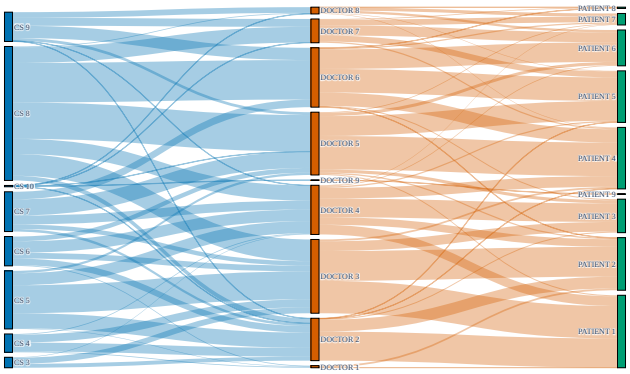


Figure 10: Sankey diagram showing how bi-gram interpretability scores shift across stakeholder audiences. Scores are averaged across LLM judges and repetitions.

I Implicit Consideration of Conditional Operations

Although our aggregation rules for scoring a full program do not explicitly consider conditional operations, the complexity they might bring is captured implicitly by the LLM scores given to `if_else_multiplexer` operator.

The `if_else_multiplexer` primitive is the only function, in the libraries, that performs ternary conditional operations, and it is scored like any other unit. Table 22 gives its unigram scores. The audiences react differently to this operator. Computer scientists, who are the audience most accustomed to conditional logic, give it 9.0 in every library, while doctors and patients score it between 4.7 and 6.1, so a program that branches explicitly is penalized for two of the three audiences, which therefore affects their priority when building audience-dependent subsets.

Table 22: Unigram interpretability scores of the `if_else_multiplexer` primitive, by audience and function library. Library 3 does not contain this primitive.

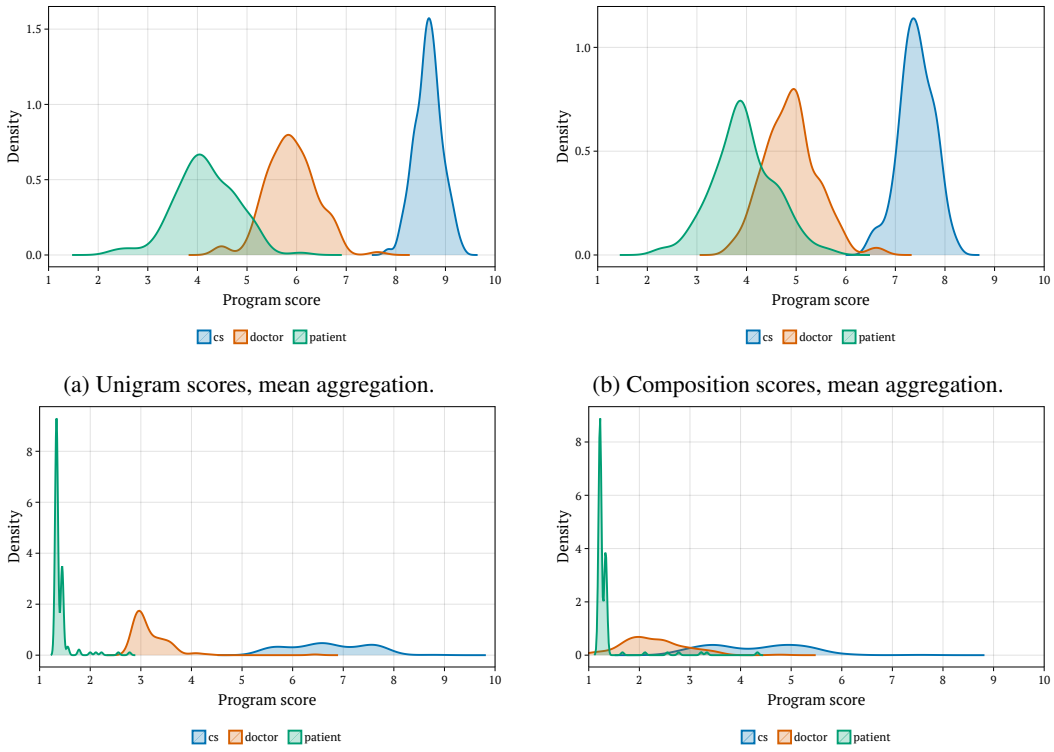
Audience	Library 1 (intensity image)	Library 2 (binary image)	Library 4 (scalar)
Computer scientist	9.00	9.00	9.00
Doctor	5.44	5.44	6.11
Patient	4.67	4.67	5.22

J PROGRAM-LEVEL INTERPRETABILITY SCORES ON BLOODMNIST

Figure 11 shows the distribution of program-level interpretability scores for the BloodMNIST symbolic dictionary. Program scores are obtained by aggregating unit-level scores over the active graph of each program, using either unigram scores or function-composition scores, and either mean or minimum aggregation as defined in Section 3.4. Mean aggregation measures the average interpretability of the operations used by a program, whereas minimum aggregation assigns the program the score of its least interpretable unit.

Under mean aggregation, unigram-based scores are generally higher than composition-based scores, especially for the computer-scientist and doctor audiences. This indicates that individual primitives are often easier to interpret than the local compositions in which they appear. The separation between audiences is also clear: computer-scientist scores are highest, doctor scores are intermediate, and patient scores are lower, reflecting the technical nature of many MAGE primitives.

Minimum aggregation produces a more conservative view of program interpretability. Because the least interpretable unit determines the full program score, distributions shift downward, especially for the patient audience. This effect is less pronounced for doctors and computer scientists, whose scores retain more variation across the scale. In the corresponding interpretability–performance curves in Figure 2, this aggregation leads to faster performance recovery than might be expected from a strictly audience-first ordering. The reason is that many programs receive the same low minimum score, especially at the bottom of the discrete 1–10 scale; within these tied groups, programs are ordered by feature importance. Minimum aggregation therefore imposes a conservative constraint by prioritizing programs with no strongly opaque unit, while also showing the importance of using sufficiently fine-grained scores to preserve meaningful preferences among candidate programs.



(a) Unigram scores, mean aggregation.

(b) Composition scores, mean aggregation.

(c) Unigram scores, minimum aggregation.

(d) Composition scores, minimum aggregation.

Figure 11: Program-level interpretability score distributions on BloodMNIST. Columns compare unigram and function-composition scoring; rows compare mean and minimum aggregation over the units in each program. Mean aggregation preserves graded audience differences, while minimum aggregation strongly penalizes programs containing at least one low-scoring unit.

K CONTROLS ON PROGRAM SELECTION

It is natural for BACC to rise as programs are added to the symbolic subset, and part of that rise is the generic effect of giving a classifier head more inputs that carry a signal, since derived from the inputs. Chance and structural orderings test how selection affects predictive recovery. A third control compares the subsets selected for different audiences. All three use the BloodMNIST runs-per-dimension dictionary of 128 programs and the nested-subset protocol of Section 3.5. The predictive controls use a logistic-regression head.

Chance ordering. We evaluated 20 random permutations of the dictionary. Feature-importance ordering beats the mean random ordering at every subset size short of the full dictionary, where every ordering necessarily contains the same programs and converges.

Table 23: Test BACC (%) for feature-importance ordering against 20 random permutations of the same dictionary.

Programs used	Feature-importance ordering	Random ordering (mean $\pm$ sd)
10	90.5	80.6 ± 3.3
20	93.7	87.5 ± 2.2
40	95.1	92.4 ± 1.0
128	96.1	96.3 ± 0.2

Reaching 90% balanced accuracy takes 10 programs under the informed ordering against 28 under the mean random ordering, and reaching 95% takes 29 against 73. Informed selection therefore reaches a given level of performance using less symbolic components.

Structural ordering. Program size is an independent selection criterion that involves no LLM. Grouping programs by their number of active operations and adding them shortest first produces the same diminishing-return shape.

Table 24: Test BACC (%) when programs are admitted in order of increasing active size.

Max. active operations	Programs	Cumulative operations	Test BACC (%)
≤ 11	47	418	90.8
≤ 14	78	812	94.6
≤ 16	96	1,088	95.8
full dictionary	128	1,726	96.5

Most predictive performance is recoverable from the shorter programs, and the final 32 programs, whose lengths range from 17 to 28 operations, buy roughly 0.8 percentage points. Program count and operation count both grow along this ordering, so the comparison does not isolate a causal effect of program length. It does show that a purely structural complexity measure can also be effective as a selection constraint, alongside the audience-conditioned scores used in the main text. Here again, feature-importance ordering was more efficient, as it only needed 10 programs to obtain 90% BACC (compared to 47 small programs needed to obtain 90.8% BACC).

Audience overlap. Another hypothesis one may think is that the patient ranking can serve as a conservative choice when the intended audience is unknown, since patients are the least specialized audience considered here. That is, we could select the same subsets for all audiences based on the patient rankings alone. However, it turns out that the same programs, selected by the patient audience, are not selected first for doctors or computer scientists. We compare each patient top-k set with the corresponding sets from doctor scores, computer-scientist scores, and feature importance. Overlap is the size of the intersection divided by k.

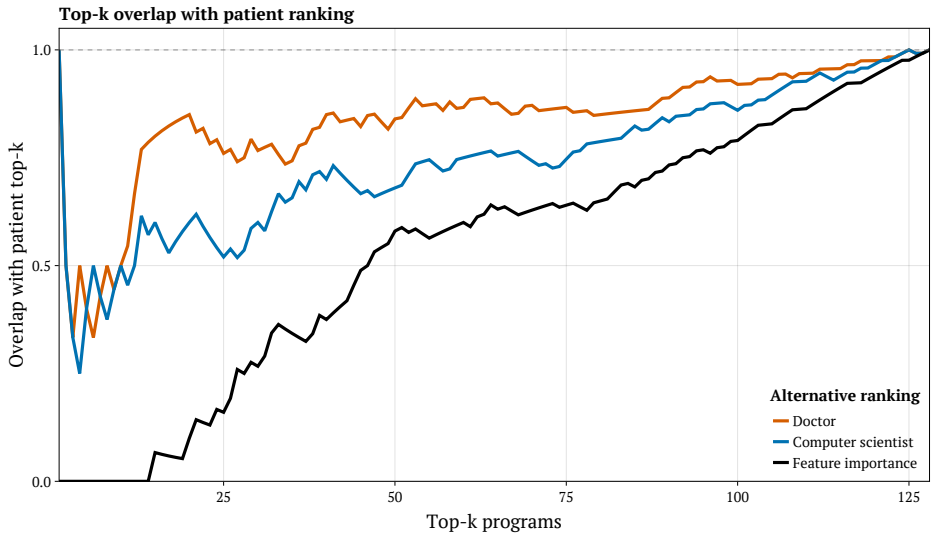


Figure 12: Top-k overlap between the patient ranking and alternative rankings on the BloodMNIST symbolic dictionary. The doctor and computer-scientist curves compare audience-conditioned rankings, while the feature-importance curve compares the patient ranking with a purely predictive ranking. An overlap of 1 indicates identical top-k sets, whereas 0 indicates no overlap.

The patient and feature-importance rankings select disjoint subsets through $k = 14$, and their overlap is 0.58 at $k = 50$. The patient and computer-scientist rankings share the first program, but overlap is 0.52 at $k = 25$ and 0.68 at $k = 50$. The doctor ranking is closer to the patient ranking, with overlaps of 0.76 and 0.84 at the same subset sizes. These differences are largest for small subsets and diminish as all rankings approach the full dictionary. The patient selection is therefore not a stricter subset of that of every other audience. These differences may come from audience-specific scores or from feature-importance tie breaks applied to different groups of equally scored programs.

L Computational Cost

The neural teachers and baselines were trained on a workstation with an Intel Xeon W-11955M CPU and an NVIDIA RTX A5000 Laptop GPU with 16 GB of memory, for 100 epochs per model. Distillation ran on nodes with two Intel Xeon Gold 6248 processors, giving 40 physical cores and 80 hardware threads per node. Four distillation searches ran concurrently on each node, with 20 threads per search. End-to-end searches used both this cluster and nodes with two 2.3 GHz Intel Skylake processors, giving 36 physical cores and 72 hardware threads per Skylake node. Four datasets used Xeon nodes for the end-to-end searches and one used Skylake nodes. Four searches ran concurrently on each node.

For every dataset, distillation performs $d \times R = 16 \times 8 = 128$ independent searches and the end-to-end baseline performs 16 searches. Every search runs under a soft budget checked between generations and a hard 20-hour limit, the latter set by the maximum wall time the cluster allows for this class of resources. The soft budget is 19 hours for the distillation searches and 20 hours for the end-to-end searches. A generation that begins before the soft budget still runs to completion, ending after it, but always under the hard limit which bounds every search. We therefore account for all searches at 20 hours. Because coordinates and seeds are independent, the searches are embarrassingly parallel, so the distillation stage for one dataset completes in about 20 hours of elapsed time when the 32 required nodes are available. Cluster rules capped the duration of each end-to-end search at 20 hours but allowed searches to run in parallel.

Table 25: Search cost for evolutionary methods. Totals are ceilings, since runs can technically converge slightly earlier, not consuming the full budget. Node-hours assume four concurrent searches per node. End-to-end physical core-hours per dataset are 2,880 on Skylake and 3,200 on Xeon.

	Per dataset		All five datasets	
	Distillation	End-to-end	Distillation	End-to-end
Independent searches	128	16	640	80
Soft budget per search (h)	19	20	19	20
Hard limit per search (h)	20	20	20	20
Run-hours	2,560	320	12,800	1,600
Node-hours	640	80	3,200	400
Physical CPU core-hours	25,600	2,880 to 3,200	128,000	15,680

The total search cost of the study is therefore at most 3,600 node-hours, or 143,680 physical CPU core-hours across the two clusters. This substantial upfront cost led us to prioritize breadth across datasets over broad ablations. The budgets were chosen conservatively to avoid truncating evolutionary runs, rather than to identify minimal search budgets. Distillation cost scales linearly with the latent dimension d and the number of seeds R. Reducing either would lower search cost, potentially at the expense of program diversity or final predictive performance. We chose $d = 16$ conservatively, above the largest class count of 11 and well below the 512-dimensional pooled representation of a standard ResNet-18.

Deployment cost is separate. Once distillation is complete the neural encoder is discarded, and inference uses only the selected programs and a lightweight classifier head. It executes entirely on CPU with modest memory and never evaluates or stores the teacher. When this cost is justified ultimately depends on the domain, as we expect some fields to be willing to spend CPU hours to turn their black-box model into a symbolic white-box approximation.

M Distillation Outperforms End-to-End MAGE at a Matched Search Budget

Distillation ran eight independent searches for each of the 16 teacher latent coordinates. Grouping their outputs by run identifier gives eight dictionaries. The first search for every coordinate supplies one dictionary of 16 programs, the second search supplies another, and so on. Each dictionary contains one program per coordinate and thus represents 16 searches, rather than the 128 searches behind the full eight-search-per-coordinate dictionary. A logistic-regression head was tuned on validation BACC and evaluated on the test split for each dictionary, giving eight distilled test scores per dataset. End-to-end MAGE also used 16 independent searches per dataset, each yielding one test score. We therefore compare eight distilled dictionaries with the 16 end-to-end runs on each of the five datasets. The dictionaries were assembled after the original searches, by using their initial run index. They let us ask whether distillation would still outperform end-to-end MAGE if we had run only one search per coordinate.

Table 26: Test BACC (%) for eight index-matched 16-program distilled dictionaries and 16 end-to-end MAGE runs per dataset. The best end-to-end score is the maximum among 16 test scores, used as a conservative comparison rather than for model selection, and can differ from the validation-selected score. Pairwise wins count the distilled versus end-to-end comparisons with higher distilled BACC. The p-values are from two-sided Mann–Whitney U tests on the eight dictionary scores and 16 end-to-end scores.

Dataset	Distilled mean $\pm$ sd	Distilled range	Best end-to-end	Pairwise wins	p
BloodMNIST	84.5 ± 1.7	81.9 to 87.2	67.3	128 of 128	2.72×10^{-6}
OrganAMNIST	74.5 ± 2.7	70.8 to 78.6	43.5	128 of 128	2.72×10^{-6}
OrganCMNIST	72.8 ± 2.0	70.5 to 77.2	39.9	128 of 128	2.72×10^{-6}
OCTMNIST	63.0 ± 4.1	57.3 to 68.8	48.2	128 of 128	2.72×10^{-6}
PathMNIST	75.3 ± 2.1	71.5 to 78.0	57.0	128 of 128	2.72×10^{-6}

On every dataset, even the lowest-scoring of the eight run-indexed dictionaries exceeds the best of the 16 end-to-end runs. The smallest gap is on OCTMNIST, where the distilled minimum is 57.3% and the end-to-end maximum is 48.2%. A two-sided Mann–Whitney U test comparing the eight dictionary scores with the 16 end-to-end scores finds a statistically significant difference in favor of distillation for each dataset, with $p = 2.72 \times 10^{-6}$ (Table 26).

The table shows that even with a matched search budget, distillation outperforms end-to-end MAGE. The consistent separation across all five datasets supports the advantage of latent decomposition even with one search per coordinate.

N INTERPRETABILITY-PERFORMANCE TRADEOFFS PER AUDIENCE ON BLOODMNIST

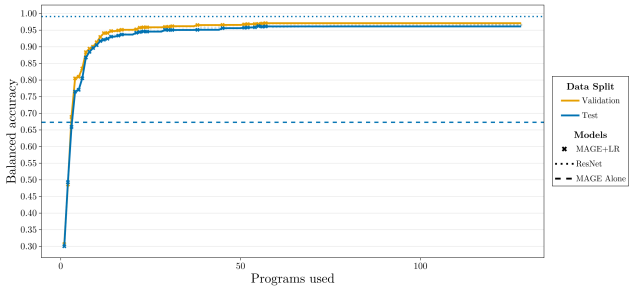
This section reports the per-audience interpretability–performance curves used to analyze stakeholder-conditioned program selection on BloodMNIST. Each plot shows how validation and test BACC changes as programs are added according to an audience-specific interpretability ordering. Vertical markers indicate when the ordered selection first includes programs below discretized interpretability-score thresholds. The feature-importance-only curve is shown as a performance-first reference. All curves in this section use a logistic-regression head; other head classifiers produced similar trends, but logistic regression was retained because it is lightweight and decomposable.

For example for the doctor audience in BloodMNIST, Figure 13d shows that it is possible to assemble a subset achieving around 82% BACC without using programs having an interpretability score less than 6.5. To achieve around 92% BACC, we would need to add programs with average scores between 6.5 and 6.0, for that audience, while adding less interpretable programs would bring marginal performance gains.

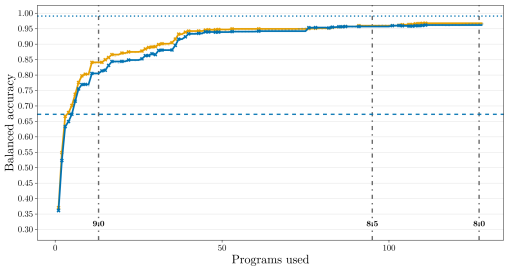
Figure 13c shows that for the computer-science audience, it is possible to reach about 90% test BACC, while only using programs were the least interpretable function has a score of more than 7.5. Accepting programs where their least interpretable function is scored at least 6.5 yields around 95%.

N.1 UNIGRAM SCORING

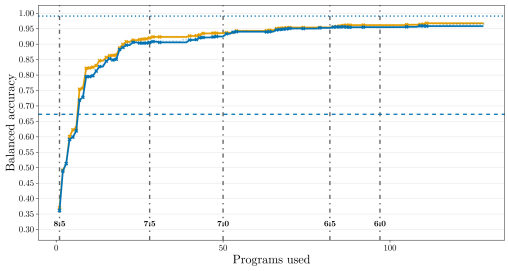
Unigram scoring assigns interpretability scores to individual MAGE primitives. For each audience, a program-level score is obtained by aggregating the scores of the primitives in the program's active graph, using either mean or minimum aggregation. This program-level score determines the order in which programs are added to the stakeholder-conditioned frontier.



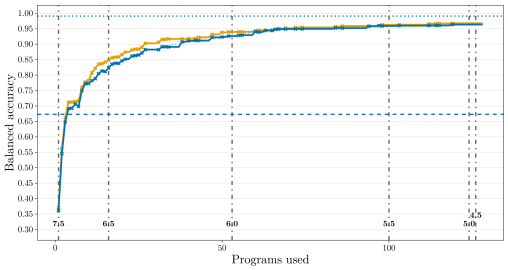
(a) Performance-first frontier on BloodMNIST using feature-importance ordering only.



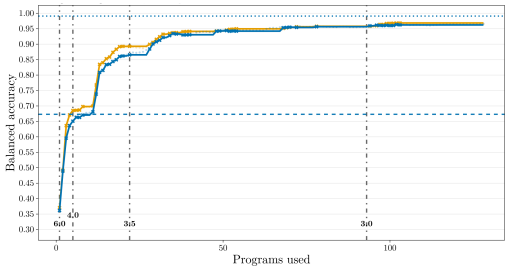
(b) Computer scientist, unigram, mean.



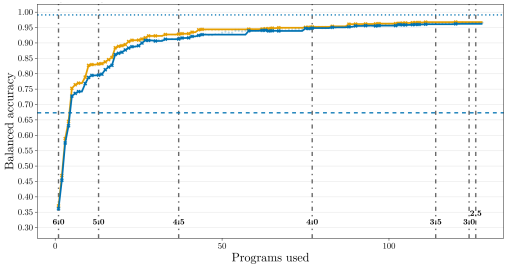
(c) Computer scientist, unigram, minimum.



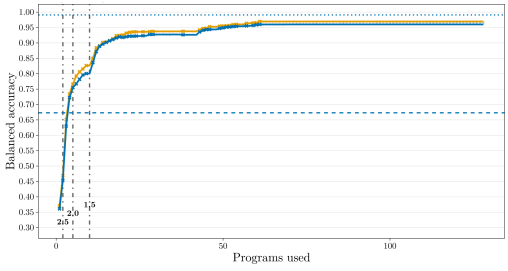
(d) Doctor, unigram, mean.



(e) Doctor, unigram, minimum.



(f) Patient, unigram, mean.

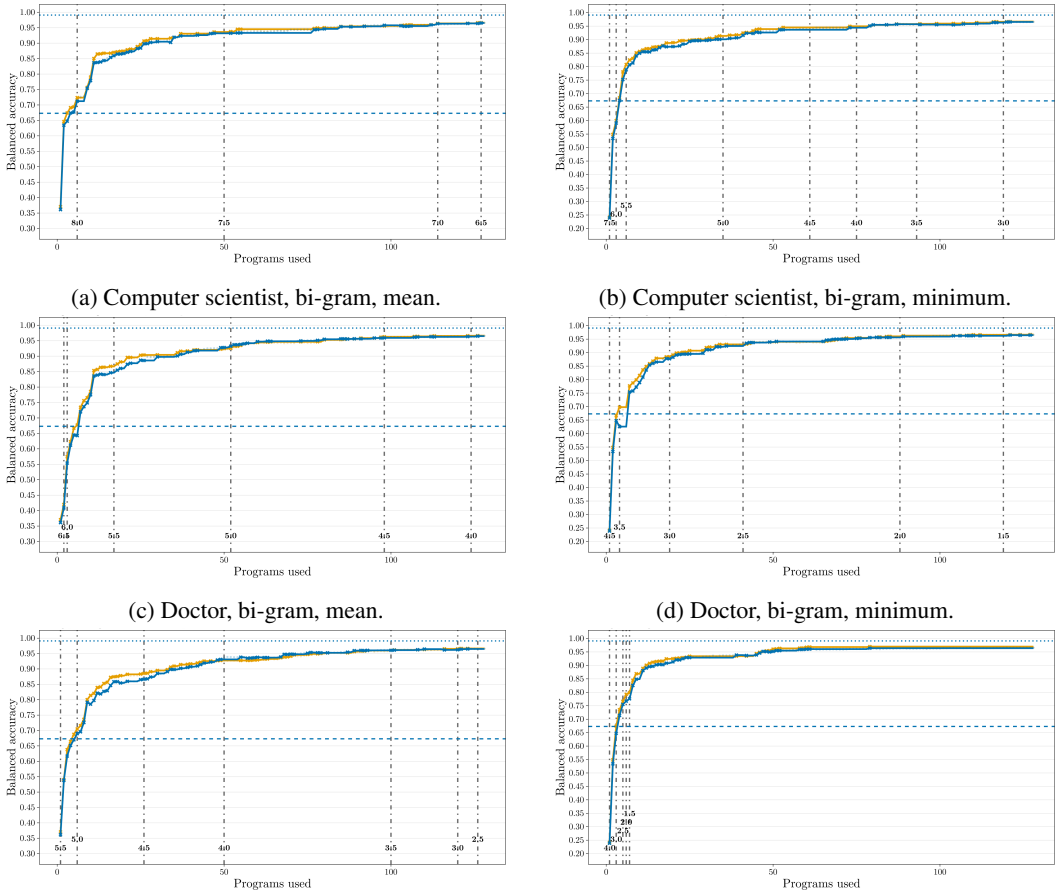


(g) Patient, unigram, minimum.

Figure 13: BloodMNIST interpretability–performance curves for the computer-scientist, doctor and patient audiences using unigram scoring, with the performance-first frontier using feature-importance ordering only in panel (a). All panels share the legend of panel (a).

N.2 Bi-gram Scoring

Bi-gram scoring assigns interpretability scores to empirically observed local function compositions. Compared with unigram scoring, it accounts for how primitives are combined inside the active program graph, and is therefore more sensitive to compositional complexity. For each audience, we report the frontier obtained with mean aggregation and minimum aggregation over the bi-gram units in each program.



(a) Computer scientist, bi-gram, mean.

(b) Computer scientist, bi-gram, minimum.

(c) Doctor, bi-gram, mean.

(d) Doctor, bi-gram, minimum.

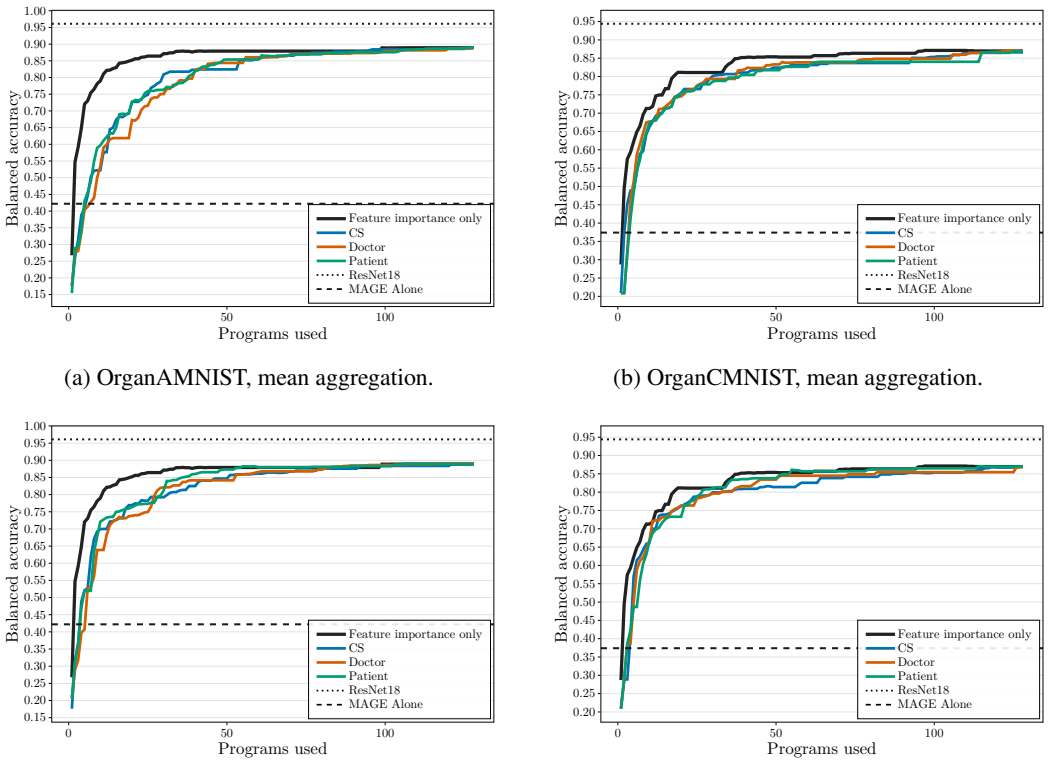
(e) Patient, bi-gram, mean.

(f) Patient, bi-gram, minimum.

Figure 14: BloodMNIST interpretability–performance curves for the computer-scientist, doctor and patient audiences using bi-gram scoring. All panels share the legend of panel (a) of Figure 13.

O INTERPRETABILITY-PERFORMANCE TRADEOFFS PER AUDIENCE ON ORGANAMNIST AND ORGANCMNIST

This section repeats the BloodMNIST analysis of Appendix N on OrganAMNIST and OrganCMNIST, using the same runs-per-dimension dictionary of 128 programs, the same logistic-regression head, and the same nested-subset protocol. Bi-gram scores are only available for the compositions observed in the BloodMNIST programs, so the audience-conditioned orderings on these datasets use unigram scores with mean and minimum aggregation. The feature-importance-only ordering is shown in each panel as the performance-first reference, together with one curve per audience. As for BloodMNIST, we find the same consistent interpretability-performance tradeoff as we add more programs to the subset.



(a) OrganAMNIST, mean aggregation.

(b) OrganCMNIST, mean aggregation.

(c) OrganAMNIST, minimum aggregation.

(d) OrganCMNIST, minimum aggregation.

Figure 15: Audience-specific interpretability–performance frontiers on OrganAMNIST and OrganCMNIST test data, under unigram scoring with mean and minimum aggregation of unit scores into program scores. Feature-importance ordering recovers performance fastest, stakeholder-conditioned orderings induce distinct frontiers and still recover high balanced accuracy. Dotted and dashed lines mark the best ResNet baseline and the validation-selected end-to-end MAGE run.

P Program Walkthrough and Local Attributions

The proposed method is not necessarily post hoc, as we envision a more participatory role between audiences and Machine Learning (ML) designers which can happen before the final model is made. An audience can state how many components it is willing to interpret and spend some time studying, and the ordering of Section 3.5 returns the programs scoring highest under that audience within that budget, with feature importance breaking ties so that predictive performance is the best available at the chosen audience interpretability score. The final model is then composed from the selected programs suited to the audience and a lightweight classifier.

Once the selection is fixed, a program can first be interpreted globally, to understand in general what it does and on what it bases its predictions. It can even be judged unsuitable, in which case another one can replace it, without repeating the evolutionary search.

For a single prediction, the programs can be ranked by the marginal effect of their output on the probability of a chosen class, either the predicted one or a class the stakeholder wants to investigate. Reading the internal computations of the highest ranked programs then exposes the visual evidence behind a prediction score, in other words what is mainly driving the prediction. The example below shows a program whose output seems to be very related to the area fraction of the nucleus and which lowers the probability of the neutrophil class as that fraction grows, a statement a stakeholder, potentially an expert, can check against what they already know about these cell types.

Because the selected components are executable, a prediction can be followed from the input image through every intermediate value to its contribution to a class probability. This appendix works through one such example. The programs considered here are the twenty highest scoring for the

doctor audience among the runs-per-dimension dictionary, and a logistic-regression head is trained on those twenty alone. The program shown is the fifth of that ranking, chosen because it acts in opposite directions on the two images below, which makes its reading informative. It is executed on two test images of different classes, and each figure reports the interpretability scores of the program for the three audiences, its output, and the effect of that output on the probability of the ground-truth class.

The effect on the prediction is reported as a marginal effect, the local derivative $\partial p_c / \partial x_j$ of the probability of class c with respect to the output of program j, evaluated at the image at hand.

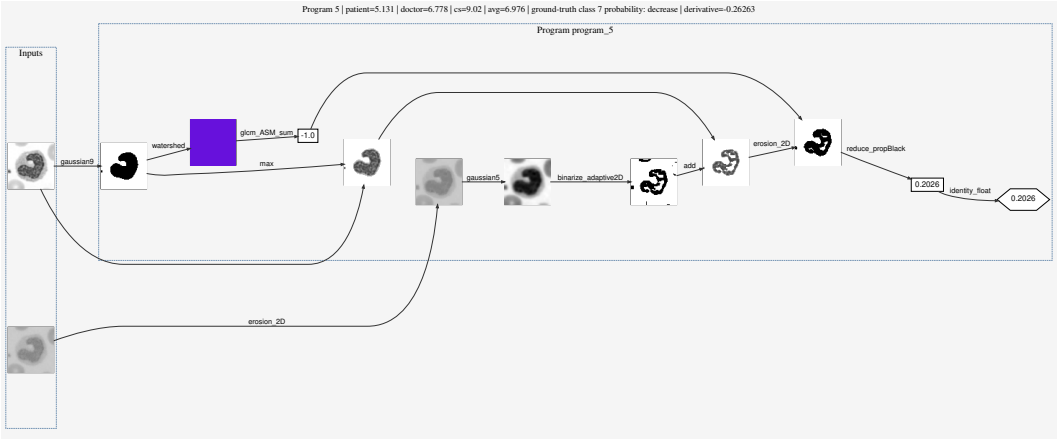


Figure 16: One distilled program executed on one BloodMNIST test image, with every intermediate value shown. The program segments the cell, isolates the nuclear structure by smoothing and adaptive thresholding, removes the background with the initial mask, and returns the proportion of black pixels, here 0.20. The head gives it a coefficient of -0.59 for the neutrophil class, so a larger nuclear area fraction lowers that class probability. The program scores 5.13 for the patient audience, 6.78 for the doctor audience and 9.02 for the computer-scientist audience, for an average of 6.98, and its local derivative on the ground-truth class probability here is -0.26.

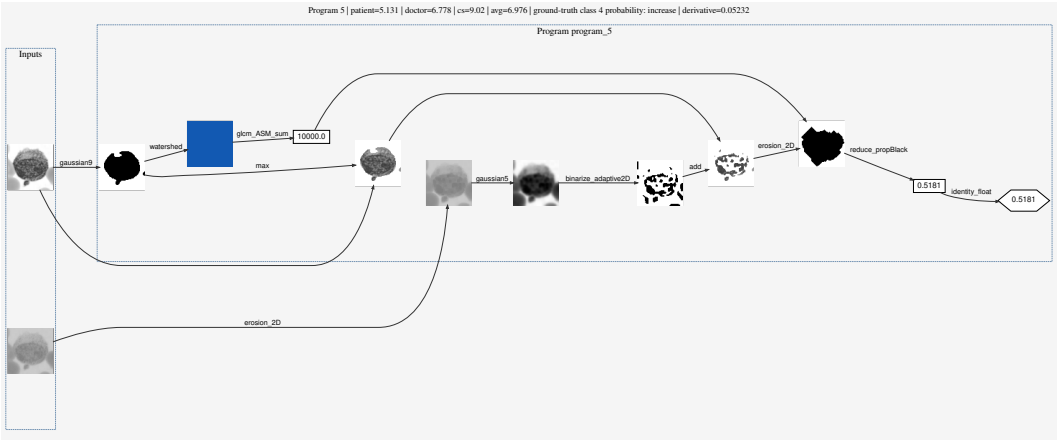
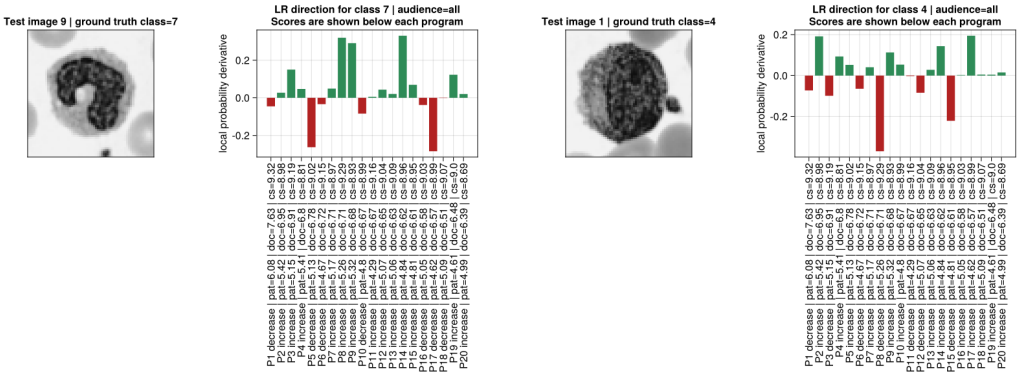


Figure 17: The program of Figure 16 executed on an immature granulocyte, with the same audience scores of 5.13 for patients, 6.78 for doctors and 9.02 for computer scientists. The thresholded nuclear structure occupies much more of the cell, and the output rises from 0.20 to 0.50. The coefficient for this class is $+0.60$ and the local probability derivative is $+0.05$, so the same measurement that argues against neutrophil argues for immature granulocyte.



(a) Neutrophil.

(b) Immature granulocyte.

Figure 18: Local probability derivatives of those twenty programs for the two images above, with the scores of each program for the three audiences printed beneath it. The thumbnail is drawn from the raw matrix and is therefore transposed with respect to the images inside the program graphs. Green raises the probability of the ground-truth class and red lowers it. Program 5 is the strongest negative contributor for the neutrophil and a positive one for the immature granulocyte.

These figures are a demonstration of inspectability rather than a clinical claim. They show that a prediction of the symbolic model can be followed from pixels to class probability through named operations, and that each step is available for a domain expert to accept or reject.

Q Qualitative Program Explanations

We use Gemini 3.1 Pro Preview to generate stakeholder-specific explanations for one representative BloodMNIST program. The program is the validation-best individual from an independent distillation run for the first teacher latent coordinate.

The program contains 16 active computation steps. Its prompt follows the same modular structure as the scoring prompts. It includes one stakeholder definition, the explanation task, the MAGE primitive names and descriptions, and the rendered program code. The multimodal LLM also receives the first three training images as visual context.

This example illustrates how an executable symbolic program can be translated into audience-dependent natural-language explanations. It demonstrates program inspectability and is not a clinically validated interpretation. Medical professionals have not reviewed the generated explanations.

```
function sequential_program(x1, x2, x3)
    tmp1 = glcm_energy_maximum(x3)
    tmp2 = glcm_energy_std(x3, tmp1)
    tmp3 = morphogradient_2D(x1, tmp1, tmp2)
    tmp4 = binarize_yen2D(tmp3, tmp2)
    tmp5 = dilation_2D(tmp4)
    tmp6 = reduce_smallerAxis(x1)
    tmp7 = gaussian5_image2D(x2)
    tmp8 = iqrpool(tmp7, tmp6)
    tmp9 = watershed_image2D(tmp8)
    tmp10 = glcm_glcm_mean_ref_sum(tmp9)
    tmp11 = min_img2D(x3, x2)
    tmp12 = maxpool(tmp11)
    tmp13 = binarize_balanced2D(tmp12, tmp10)
    tmp14 = add_img2D(tmp13, tmp5)
    tmp15 = reduce_propBlack(tmp14)
    tmp16 = identity_float(tmp15)
    out1 = tmp16
```

```
    return (out1,)
end
```

Computer Scientist Perspective

Program_interpretability_score: 5
Short summary: The program computes the proportion of background pixels by combining a boundary mask of the red channel with a texture-parameterized thresholded mask of the blue and green channels.
Stakeholder_explanation: This program acts as a specialized feature extractor that estimates the relative background area of the image after masking out cellular structures. It operates via a dual-branch pipeline. The first branch identifies cell edges by computing the morphological gradient of the red channel, binarizing it, and dilating it to create a thick edge mask. The morphological parameters are dynamically driven by textural energy computed from the blue channel. The second branch targets dense cellular regions (like nuclei) by taking the minimum intensities of the blue and green channels, max-pooling the result, and thresholding it. This thresholding uses a global parameter derived from a watershed segmentation of the green channel's local variance. Finally, the program combines both the edge mask and the regional mask using a logical OR (addition) and returns the fraction of pixels that remain unmasked (black). This scalar output essentially serves as an inverse proxy for the total area occupied by the cell and its prominent features."

Doctor perspective

Program_interpretability_score: 5
Short summary: The program estimates the relative area of the cell nucleus by isolating dense nuclear material and masking out the cell boundaries.
Stakeholder_explanation: This program calculates a structural feature of the blood cell that is highly relevant for clinical classification: the proportional area of the cell nucleus. First, it identifies the physical boundaries of the cells by detecting sharp contrast changes in the red color channel, creating a map of thickened cell outlines. Second, it isolates the dense, dark regions of the cell—mainly the nucleus—by combining the blue and green channels (where hematoxylin-stained nuclei appear darkest) and separating them from the brighter background and cytoplasm. By merging these two maps, the program essentially 'masks out' the background, cytoplasm, and cell edges, leaving only the solid interior of the nucleus represented as black pixels. Finally, it calculates the percentage of the image covered by these black pixels. For a physician, this measurement directly correlates to the nuclear size or nuclear-to-cytoplasmic ratio, which is a fundamental visual criteria used to differentiate types of white blood cells (for instance, distinguishing lymphocytes, which have large nuclei, from other leukocytes).

Patient perspective

Program_interpretability_score: 3
Short summary: The program creates a combined black-and-white mask of the image's thickened boundaries and its prominent colored regions, then calculates the percentage of the image that remains dark (background).
Stakeholder_explanation: Imagine we are looking at a blood cell image separated into three different color layers. The program processes this image in two main ways to figure out what is cell and what is background. First, it looks for the outlines or edges of the cell structures in one color layer and makes those outlines thicker so they stand out. Second, it compares two of the other color layers, picks out specific dark and bright spots, and turns them into a simple black-and-white shape. After doing this, it stacks the thickened outlines on top of the black-and-white shapes. The final number the program gives you is simply the percentage of this final stacked picture that remains completely dark. Essentially, it is measuring how much 'empty space' is left after highlighting the most important parts of the cell.

R MAGE PARAMETERS

R.1 FUNCTION LIBRARIES FOR ALL MAGE INDIVIDUALS

MAGE individuals have access to four typed function libraries. Each primitive returns an output of a type associated with its library, such as an intensity image, binary mask, segmentation map, or scalar value. However, primitives may accept inputs from multiple compatible types; for example, `reduce_nColors` can operate on either an intensity image or a binary image to count the number of unique pixel values.

Library	Description	# Functions
Intensity Image	Matrix filled with values between 0 and 1	77
Binary Image	Matrix filled with binary values (0 or 1)	82
Segmented Image	Matrix filled with integer valued pixels	6
Scalar	Floating point numbers	137

Table 27: Function libraries used by MAGE individuals.

R.2 MAGE END-TO-END EA SEARCH PARAMETERS

For each dataset, we include an end-to-end MAGE baseline in which evolution directly searches for a multiclass image classifier. Each individual p outputs a vector of class scores $p(x) \in \mathbb{R}^C$, and prediction is made by

$$\hat{y}(x) = \arg \max_{c \in \{1,\dots,C\}} p_c(x).$$

Thus, unlike the distilled MAGE students, which approximate a single teacher latent coordinate, the end-to-end baseline must solve the full image-to-label task within one program.

MAGE is evolved with a simple genetic algorithm. At each generation, individuals are evaluated on a class-balanced mini-batch B_g sampled from the training set, with 8 examples per class. This gives batch sizes of 64 for BloodMNIST, 88 for OrganAMNIST, 32 for OCTMNIST, and 72 for PathMNIST. For an individual p, we compute the balanced error rate

$$\mathrm{BER}(p; B_g) = 1 - \mathrm{BACC}(p; B_g),$$

using its argmax predictions. We also compute the negative log-likelihood of the softmax-normalized class scores. The minimized fitness is

$$\mathcal{F}_{\mathrm{e2e}}(p; B_g) = 0.8\,\mathrm{BER}(p; B_g) + 0.2\,\mathcal{L}_{\mathrm{NLL}}(p; B_g) + \frac{\log(\tau(p; B_g))}{\lambda_{\mathrm{time}}},$$

where $\tau(p; B_g)$ is the evaluation time of the program on the batch and λ_{time} is the time-penalty parameter. The NLL term provides a smoother optimization signal than balanced error rate alone, while the time penalty discourages computational bloat.

Selection keeps the best 16 individuals on the current mini-batch as elites. The best elite of each generation is additionally evaluated on the full validation set, and the final program returned by a run is the validation-best elite encountered during the search. New non-elite individuals are produced as mutated copies of tournament-selected parents from the previous population. Mutation is applied to active graph material by modifying the function or connections of a randomly selected active node.

Compared with the latent-distillation setting, the end-to-end baseline uses larger programs and more independent runs. We allow 100 nodes per individual, rather than 30, because a multiclass classifier is expected to require a larger computation graph than a program approximating a single teacher coordinate. We also perform 16 independent runs per dataset (instead of the 8 used for each distillation coordinate). The time-penalty parameter is set to 100, compared with 1000 in the distillation runs, to more strongly control computational bloat induced by the larger graphs.

R.3 MAGE (STUDENTS) EA SEARCH PARAMETERS

The MAGE distillation runs use the same GA structure as the end-to-end MAGE baseline, but with a smaller genome size and a different objective. Each student program approximates a single

Parameter	Value
Population size	144
Elite size	16
Number of nodes	100
Tournament size	7
Independent runs	16
Generations	Until time budget
Time budget	20 hours soft and hard
Examples per class per batch	8
Time-penalty parameter λ_{time}	100
Fitness	$0.8\,\text{BER} + 0.2\,\text{NLL} + \log(\tau)/\lambda_{\text{time}}$
Outputs per program	C, the number of dataset classes

Table 28: Genetic algorithm parameters for the end-to-end MAGE multiclass baseline. Each individual outputs one scalar per class, and predictions are obtained by taking the argmax over class scores.

teacher latent coordinate, so we use 30 nodes per individual and one scalar output. The fitness is the Spearman-correlation distillation loss defined in Section 3.2, with the same runtime penalty used to discourage slow programs. Because the distilled programs are smaller than the end-to-end multiclass programs, the time-penalty denominator is set to 1000.

MAGE students used a sigmoid output activation during distillation, whereas teacher latent coordinates use asinh. Because the distillation loss is Spearman-based, the objective is invariant to monotone rescalings of the student output, so this affects calibration rather than the learned ranking signal. For classifier-head training and inference, retained program outputs are transformed with asinh and standardized before supervised recombination.

Parameter	Value
Population size	144
Elite size	16
Number of nodes	30
Tournament size	7
Independent runs	8 per latent coordinate
Generations	Until time budget
Time budget	19 h soft, 20 h hard
Time-penalty parameter λ_{time}	1000
Fitness	$1 - \lvert r_{\text{S}}\rvert + \log(\tau)/\lambda_{\text{time}}$
Latent activation	sigmoid
Outputs per program	1

Table 29: Genetic algorithm parameters for MAGE latent-coordinate distillation. Each run evolves one image-to-scalar program to approximate one teacher latent coordinate.

S Machine Learning head parameters

S.1 Logistic Regression parameters

From each program dictionary, we train a lightweight classifier head to map the selected program outputs directly to class labels. We consider three nested dictionaries. The *best-per-dimension* dictionary contains one validation-best program for each of the $d = 16$ teacher latent coordinates. The *runs-per-dimension* dictionary contains the validation-best program from each independent distillation run, giving 16×8 programs. The *all-history* dictionary further includes intermediate elites encountered during evolution, namely the validation-best program of each run together with the individual that was elite every 1000 generations. Its size therefore depends on the number of generations completed by each run and varies across datasets.

For each dictionary and selected subset of programs, we train a logistic-regression head using validation BACC for model selection. Table 30 reports the grid used for hyperparameter tuning. Logistic regression is used as the main head because it is lightweight, decomposable, and consistent with the goal of keeping the symbolic bottleneck inspectable.

Feature importance for logistic regression is computed as the absolute value of each fitted coefficient, $|\beta_p|$, for program feature p.

Parameter	Values used for tuning
C	$\{10^{-3}, 10^{-2}, 10^{-1}\}$
Penalty	`{elasticnet}`
l_1 ratio	$\{0.0, 0.25, 0.5, 0.75, 1.0\}$
Solver	`saga`
Class weight	`balanced`
Max iterations	3000
Random state	1

Table 30: Logistic regression hyperparameters used for parameter tuning. Fixed parameters are also reported for reproducibility.

S.2 Random Forest parameters

Table 31 reports the random-forest hyperparameter grid used for the results in Table 1. Random forests are included as an additional head-classifier comparison to assess whether the distilled program dictionaries become more predictive when recombined by a nonlinear classifier, rather than only by the logistic-regression head used in the main interpretability analysis.

Parameter	Values
Maximum depth	$\{10, 20, 30\}$
Minimum samples split	$\{5, 10, 20\}$
Maximum samples	$\{0.6, 0.7, 0.8\}$
Minimum samples leaf	$\{2, 5\}$
Maximum features	$\{$`sqrt`$, 0.5, 1.0\}$
Criterion	$\{$`gini`$,$ `entropy`$\}$
Number of trees	300
Class weight	`balanced`
Random state	1

Table 31: Random-forest hyperparameters used for classifier-head tuning. The top block reports the grid-search parameters; the bottom block reports fixed parameters used for all runs.

S.3 SVM parameters

For the SVM head, we use a balanced third-degree polynomial kernel with $C = 5.0$, `random_state`=1, `cache_size`=4000, and a maximum of 10,000 iterations.

T Baselines convergence curves

This section reports validation convergence curves for the neural baselines on each dataset. The curves show that the 30-epoch training budget was sufficient for the ResNet-18, ResNet-34, and ResNeXt-50 baselines to reach stable validation performance. As a check, we also trained selected architectures and datasets for 100 epochs and obtained very similar validation and test results. For each model, the checkpoint with the highest validation BACC is retained for test evaluation.

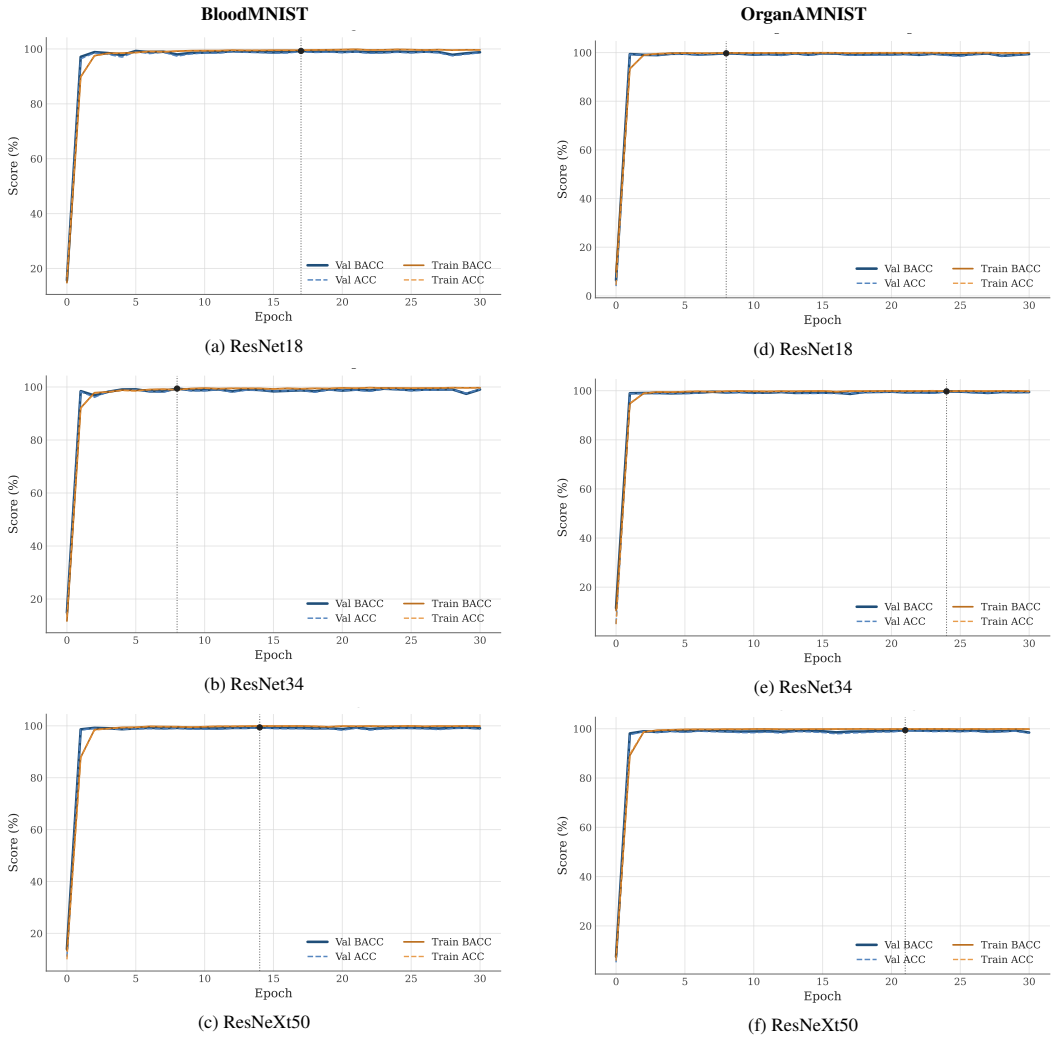


Figure 19: Convergence curves for neural baselines on BloodMNIST and OrganAMNIST. For each run, the checkpoint with the highest validation BACC was retained for test inference.

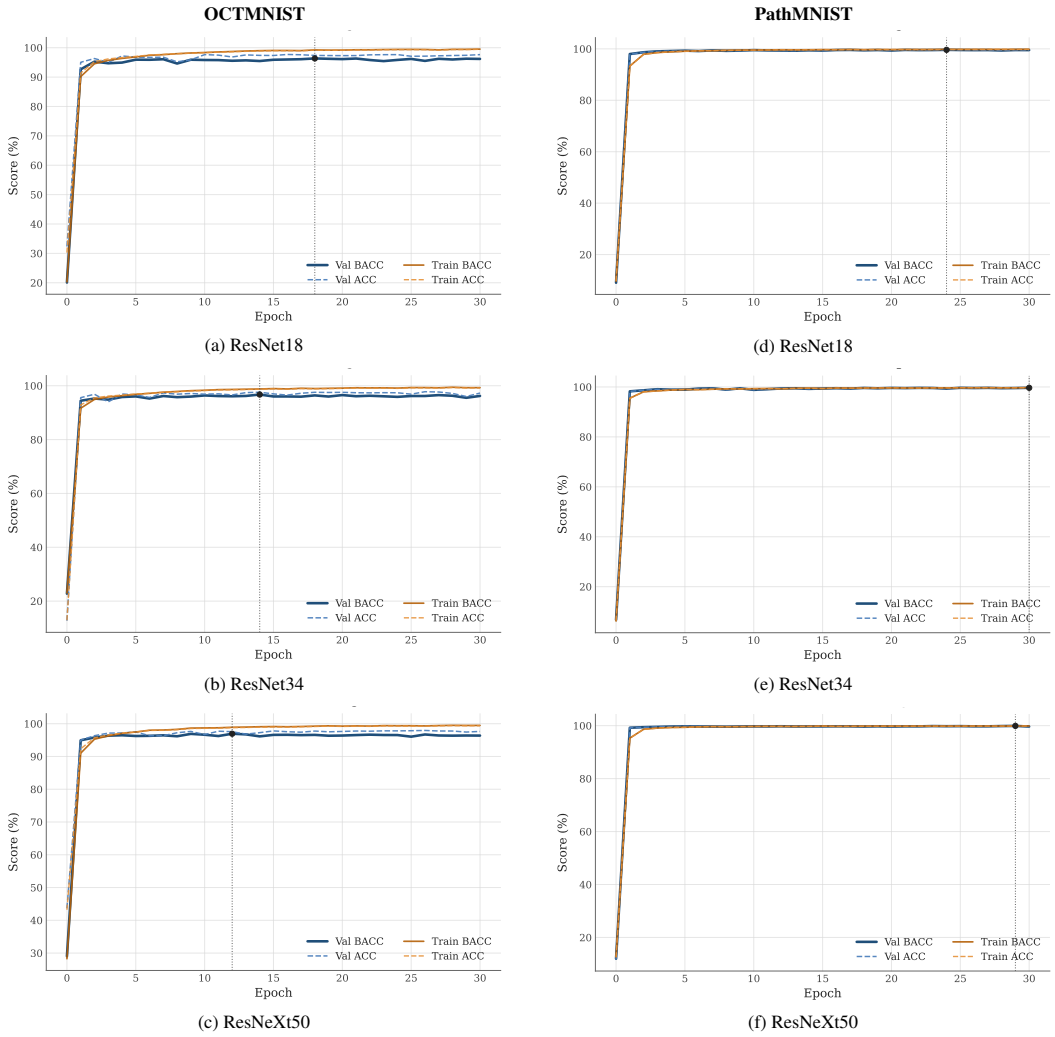


Figure 20: Convergence curves for neural baselines on OCTMNIST and PathMNIST. For each run, the checkpoint with the highest validation BACC was retained for test inference.

U DISTILLING LOSSES

For stochastic evolutionary components, we report variability across independent runs. We do not perform formal hypothesis testing, since the main comparison concerns large performance gaps rather than marginal improvements, see Table 1. The following boxplots summarize the run-level distillation fitness for each latent coordinate and dataset.

U.1 BLOODMNIST LOSS PER DIMENSION

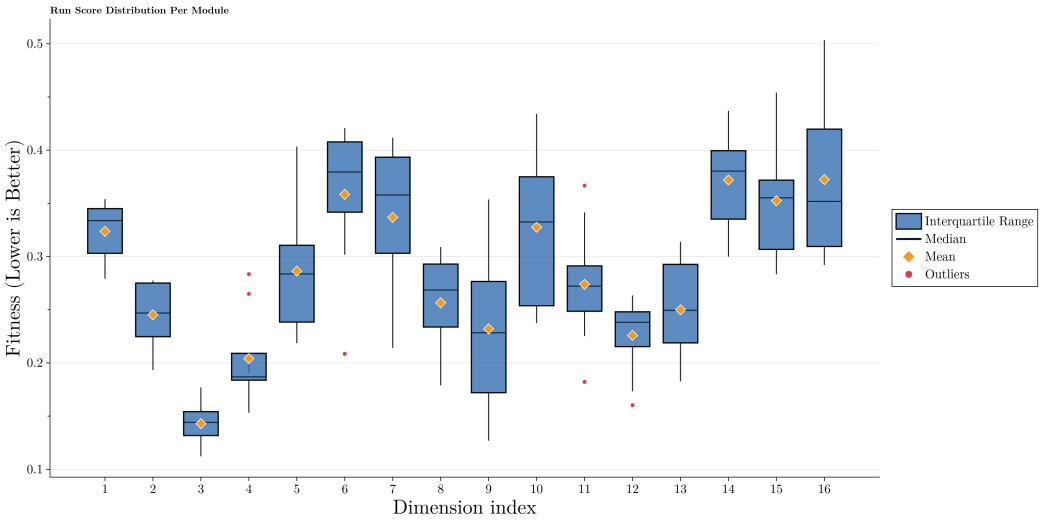


Figure 21: BloodMNIST distillation fitness by latent coordinate. Lower values are better; fitness is $1 - |r_S|$, where r_S is the Spearman correlation between the MAGE program output and the teacher latent coordinate. 8 runs per coordinate.

U.2 ORGANAMNIST LOSS PER DIMENSION

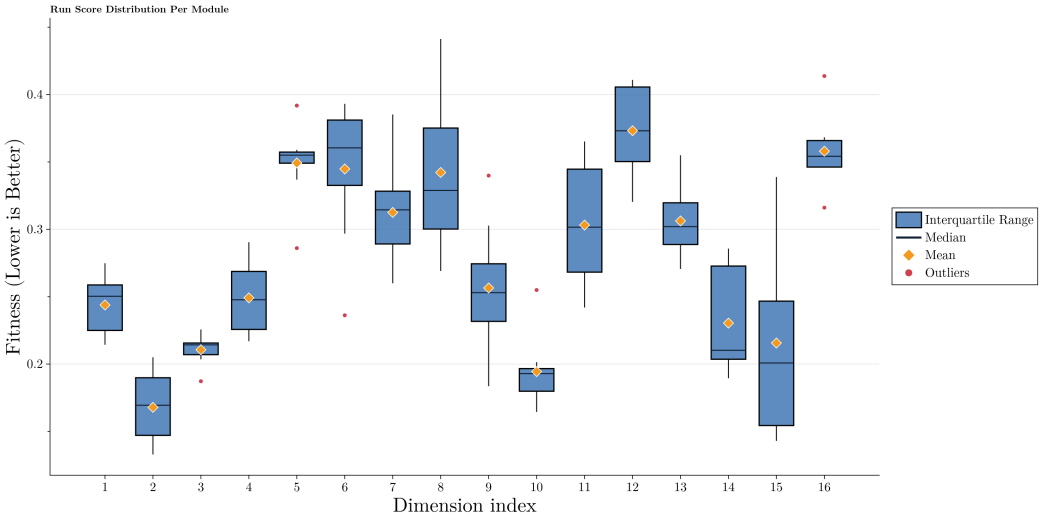


Figure 22: OrganAMNIST distillation fitness by latent coordinate. Lower values are better; fitness is $1 - |r_S|$, where r_S is the Spearman correlation between the MAGE program output and the teacher latent coordinate. 8 runs per coordinate.

U.3 ORGANCMNIST LOSS PER DIMENSION

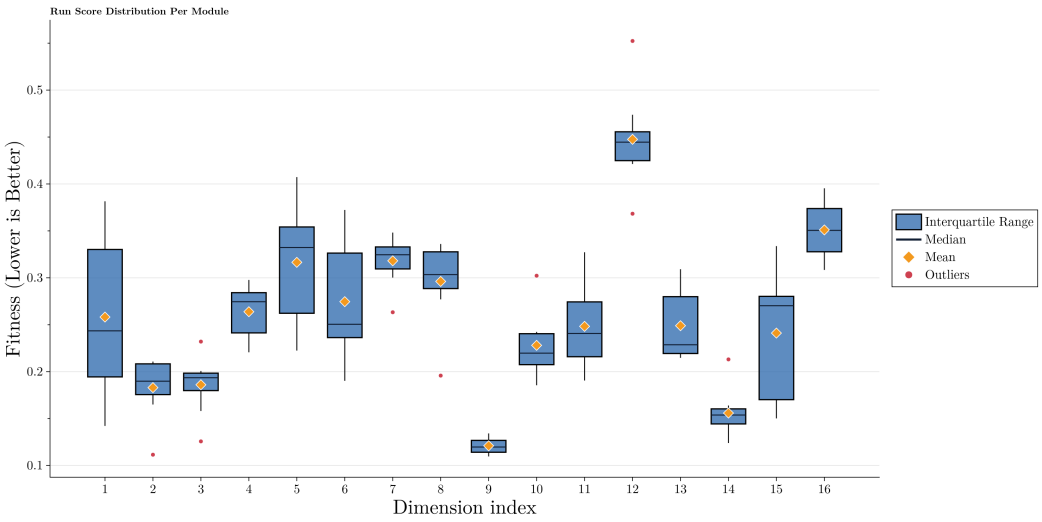


Figure 23: OrganCMNIST distillation fitness by latent coordinate. Lower values are better, and fitness is $1 - |r_S|$, where r_S is the Spearman correlation between the MAGE program output and the teacher latent coordinate. 8 runs per coordinate.

U.4 OCTMNIST LOSS PER DIMENSION

Compared with the other four datasets, OCTMNIST has higher distillation loss across several latent coordinates. Seven coordinates have median loss above 0.5, compared with none on the other datasets. For coordinate 5, the median loss is about 0.79, corresponding to an absolute Spearman correlation of about 0.21 with the teacher coordinate. We hypothesize that weak approximation of these coordinates contributes to OCTMNIST's lower BACC. Targeted hyperparameter tuning might improve their approximation, but we did not tune individual coordinates in order to maintain a common search budget across datasets. The high losses may also reflect operations missing from the current primitive libraries, potentially contributing to weaker discrimination of some OCTMNIST classes.

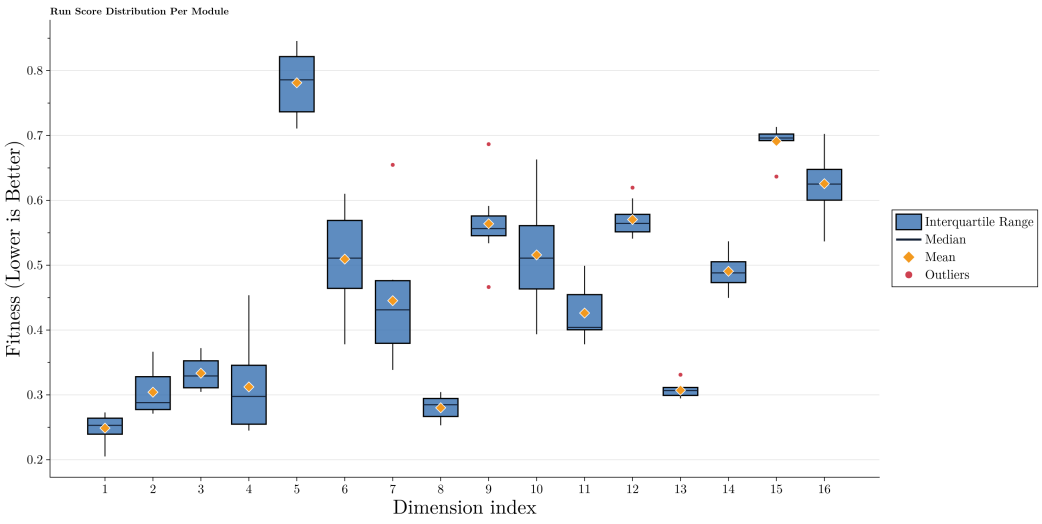


Figure 24: OCTMNIST distillation fitness by latent coordinate. Lower values are better; fitness is $1 - |r_S|$, where r_S is the Spearman correlation between the MAGE program output and the teacher latent coordinate. 8 runs per coordinate.

U.5 PathMNIST loss per dimension

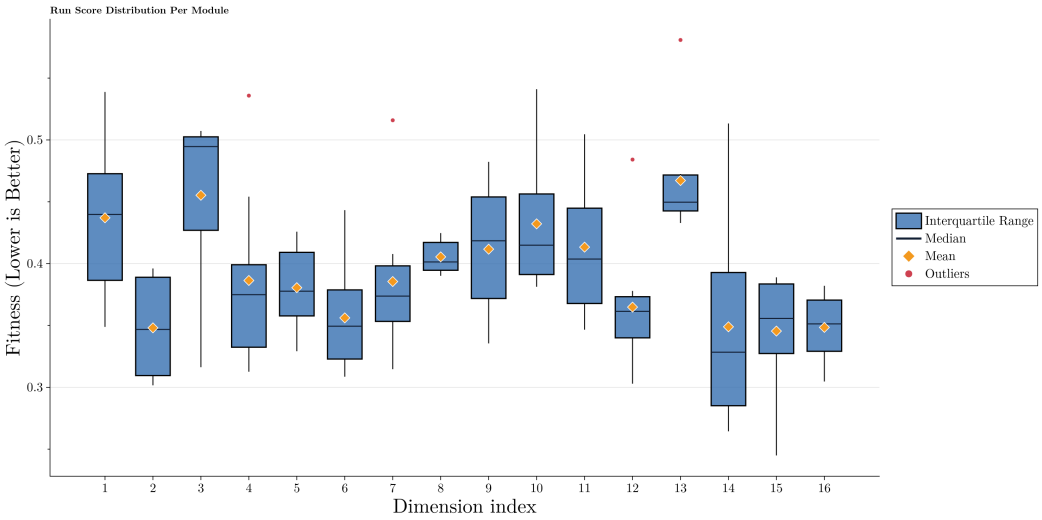


Figure 25: PathMNIST distillation fitness by latent coordinate. Lower values are better; fitness is $1 - |r_S|$, where r_S is the Spearman correlation between the MAGE program output and the teacher latent coordinate. 8 runs per coordinate.

V MAGE E2E scores

Table 32 reports validation and test BACC for the end-to-end MAGE baseline across independent runs. The column "Val & Best." denotes the run with the highest validation BACC; "Test & Of Best Val." reports the corresponding test score and is the value used in the main comparison table (Table 1). The remaining columns summarize the distribution of validation and test performance across runs.

Table 32: End-to-end MAGE validation and test BACC across independent runs. "Test of Best Val." is the test score of the run selected by validation performance and is the value reported in the main results table.

Dataset	Validation			Test			
	Mean	Best	Min	Of Best Val.	Mean	Min	Best
BloodMNIST	59.6 $\pm$7.4	70.4	46.3	67.3	58.1 $\pm$7.0	46.4	67.3
OrganAMNIST	32.8 $\pm$14.9	59.7	18.4	42.2	27.1 $\pm$10.7	15.6	43.5
OrganCMNIST	26.8 $\pm$12.0	47.7	16.8	37.4	21.9 $\pm$10.1	13.9	39.9
OCTMNIST	52.6 $\pm$5.4	61.0	38.1	46.7	41.3 $\pm$5.7	25.2	48.2
PathMNIST	41.9 $\pm$8.5	54.0	26.6	54.7	42.8 $\pm$9.3	27.9	57.0

W Human Validation Study

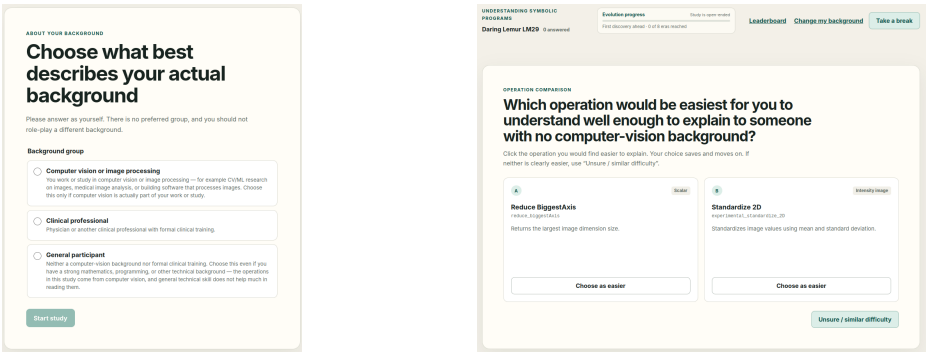
Participants compared pairs of symbolic units to assess whether their choices agreed with LLM rankings of which unit was easier to understand. Pairwise choices provide a direct agreement measure without requiring volunteers to assign absolute interpretability scores (Section 5.3).

W.1 Task and interface

Each comparison presented two cards side by side (Figure 26). Each card gave a human-readable name, the canonical raw primitive name, a short description, and the output type of the unit. Participants chose the unit they could more easily understand well enough to explain to someone without

a computer-vision background. They could also mark the units as equally difficult or indicate uncertainty. Sampled pairs of symbolic units were judged by participants from different background groups. No LLM-generated explanations or scores were shown to participants or used to construct the survey. Each session began with a short tutorial, whose responses were excluded from analysis.

Function comparisons paired individual primitives, while chain comparisons paired four-operation excerpts of evolved programs. In a pilot comparing whole programs, participants chose the shorter program. Fixing chain length at four operations therefore removed program length as a confound when assessing composition.



(a) Background questions.

(b) One function comparison.

Figure 26: The comparison interface. Panel (a) collects the self-reported background that assigns the participant to a group. Panel (b) shows one function comparison, with the two cards and the options for equal difficulty and for uncertainty.

W.2 Materials and pair construction

The function inventory contained the 302 primitives of the MAGE libraries (Appendix D). The chain inventory contained 881 four-operation excerpts from the 16 exported BloodMNIST programs. Comparison pairs were selected in advance and stratified so the two cards came from different primitive families. Each family's share of card slots was proportional to the square root of its size and capped at 10 percent. This prevented large families of similar operations from dominating the comparisons. For example, the GLCM family contained 50 functions, while the Local Extrema family contained only 2 operators.

W.3 Participants

Volunteers self-selected into general, clinical, and computer-vision groups. The clinical group included physicians and other explicitly defined clinical professionals, and the computer-vision group included participants with a computer-vision or image-processing background. Everyone else entered the general group. The groups were evaluated against the patient, doctor, and computer-scientist scores, respectively (Section 3.4). Of 63 enrolled sessions, 53 participants answered at least one comparison and contributed 2,560 responses. Fifty participants contributed function responses, including 1,559 discriminating comparisons, 405 responses marked as unsure, and 16 responses on pairs whose LLM scores were tied. Thirty-six participants contributed 527 chain responses, of which 352 were decided.

Participants were recruited through direct messages, email, and posts on Slack, Discord and LinkedIn. Participation was voluntary and uncompensated. The study recorded each participant's choices, response times, and selected background category, and collected no name, email address, IP address, or medical information. Participants gave informed consent on the screen reproduced below before beginning, and could stop at any time without giving a reason. The study was granted exempt status by an institutional review board on the basis of its scope.

The text shown to participants before they agreed to take part was the following.

Help us learn how people read short operations and symbolic chains. Answer as many comparisons as you like.

This study is conducted for academic purposes only. We do not collect personal information.

Before you begin

Taking part is voluntary.
You may stop at any time without giving a reason.
We collect your choices, response times, and selected background category, but no name, email address, IP address, or medical information.
This study concerns ease of understanding. It does not assess your intelligence or professional ability.

I am at least 18 years old, have read the information above, and voluntarily agree to take part.

W.4 Agreement measure and statistical analysis

For each discriminating comparison, agreement was 1 if the participant chose the unit with the higher aggregate LLM score $\bar{s}_g(u)$ for the audience g matched to their group, and 0 otherwise. Unsure responses were retained but not counted as directional choices, and pairs with tied LLM scores were excluded. Chance agreement was therefore 0.5. The agreement rate was the proportion of discriminating comparisons with agreement 1. We computed 95 percent percentile confidence intervals from 2,000 bootstrap replicates that resampled participants rather than trials, accounting for responses clustered within participants. Judge-family agreement used each family's scores in place of the aggregate. Chain agreement used each of the four program-level scores from Section 3.4. The minimum aggregations had 350 rather than 352 decided trials because two pairs were tied under those scores.

A regularized Bradley–Terry model estimated a relative score for each primitive from the human pairwise preferences. Spearman ρ and Kendall τ_b compared these human-derived scores with the aggregate LLM scores across primitives. Both correlations range from -1 to 1, with 0 indicating no rank association. Kendall τ_b adjusts for tied scores if present.

W.5 Results

Table 33: Function-level agreement by participant group. Agreement is measured against the aggregate LLM score for the audience matched to each group. Chance is 0.5.

Participant group	Participants	Trials	Agreement	95% CI
Overall	50	1,559	0.768	[0.734, 0.801]
General	22	700	0.804	[0.761, 0.846]
Clinical professional	11	422	0.735	[0.669, 0.809]
Computer vision / image processing	17	437	0.744	[0.678, 0.803]

Table 34: Function-level agreement by judge family, using the scores of a single family in place of the aggregate.

Judge family	Trials	Agreement	95% CI
Gemini	1,469	0.792	[0.758, 0.823]
Claude	1,508	0.771	[0.743, 0.799]
ChatGPT	1,511	0.756	[0.720, 0.790]

Table 35: Agreement on four-operation chains for each program-level score, over 352 decided comparisons from 36 participants.

Chain predictor	Trials	Agreement	95% CI
Mean unigram score	352	0.926	[0.875, 0.972]
Minimum unigram score	350	0.914	[0.857, 0.969]
Mean bi-gram score	352	0.813	[0.765, 0.857]
Minimum bi-gram score	350	0.806	[0.761, 0.846]

Participants chose the LLM-preferred function in 76.8 percent of discriminating comparisons (Table 33). Every participant group's confidence interval excludes chance, with the highest agreement in the general group. Agreement also exceeds chance for each judge family separately (Table 34). Gemini has the highest estimate and ChatGPT the lowest, although their intervals overlap substantially.

For composed chains, all four scoring rules agreed with human choices above chance (Table 35). Mean aggregation had slightly higher agreement than minimum aggregation, and unigram scoring exceeded bi-gram scoring. The higher unigram agreement alone does not establish that unigram scores capture compositions better. Participants marked 20.5 percent of function responses as unsure, while 33.2 percent of chain responses were unsure or tied. The higher non-directional share for chains is consistent with greater difficulty comparing composed units than single operations.

The human-derived and LLM scores showed positive rank correlation, with Spearman $\rho = 0.658$ and Kendall $\tau_b = 0.463$. Bootstrap values of Spearman ρ were concentrated between approximately 0.52 and 0.66. Both the comparison-level agreement and the item-level correlations indicate that LLM scores carry information about which symbolic units participants find easier to inspect.

W.6 Scope of the validation

Human choices support the LLM rankings of individual primitives and four-operation compositions. Because participants compared pairs, the study does not validate the absolute interpretability scores assigned to those units. Absolute ratings would require a shared scale across all functions so that scores are comparable. This is easier to implement with LLM judges than with human participants, who would need an overview of the inventory to calibrate their ratings. Whole-program judgments would require participants to inspect more structure and likely spend longer on each question. Participants were volunteers with self-reported backgrounds, so the agreement rates describe this sample rather than a population of intended users. Mean and minimum program aggregation remain simple and transparent selection rules, but this study does not validate them as measures of whole-program interpretability.