FREEDOM CAUSES NEURAL GENERALISATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Can increasing freedom improve neural generalisation? In theory, maximising freedom is necessary to maximise generality. Furthermore, freedom of function measures behaviour, whereas complexity- or gradient-based alternatives are measures of form. This means freedom is reparameterisation invariant, and should yield more consistent performance improvements across architectures, embodiments and datasets. Yet in practice, freedom has proven difficult or intractable to operationalise. Here I present a fully neural approximation of freedom of function, estimating how many further input–output commitments remain possible while preserving correctness on training data. In each experiment, training data is used to calibrate an instrument with which freedom is approximated. Then, neural networks are trained and undergo evolutionary selection for freedom measured using that instrument. On all five benchmarks tested, KMNIST, binary Fashion-MNIST, Rotten Tomatoes, binary AG News and binary 20 Newsgroups, this intervention achieves higher mean test accuracy than evolution using either approximate marginal likelihood or gradient-based GdScore. It wins 86 of 120 pairwise comparisons, by up to 14.8%. As predicted, freedom is more consistent. Freedom's final pick beats a random on every benchmark, by as much as 9.6%. On the text benchmarks in particular, a single generation of freedom selection improves test accuracy by 4.48–6.09%. These results offer causal evidence that selecting for freedom improves neural generalisation, connecting freedom of function to learned representations and adaptive behaviour.

1 INTRODUCTION

More than one neural representation can fit a set of training data. Though these representations might fit the training data equally well, they can differ with respect to everything else. Beyond the training data, each choice of representation determines which further behaviours a network can express. I investigate whether selecting representations for *freedom of function* improves generalisation. Here, freedom concerns additional input–output commitments[1] that remain possible while existing commitments stay satisfied.

Freedom of function is a matter of behaviour, rather than how a network is described. For example, a *reparameterisation* changes the coordinates used to express a network. When such a reparameterisation preserves behaviour, freedom remains unchanged. Any measure which is affected by such reparameterisation is a measure of form, rather than function. For instance a parameter-based score would vary with such a change in coordinates (Dinh et al., 2017). As freedom is a measure of function, not form, this motivates a prediction that selection by freedom will improve generalisation more consistently across architectures and datasets[2]. I test this predicted consistency by comparing each selector with a random choice from initial trained populations.

Freedom of function is typically measured in the context of an abstraction layer[3], mediating between form and function. Representation learning deals in learning such abstraction layers, changing which

[1] A *commitment* is a requirement on a network's behaviour, such as assigning a given class to a given input. Section 2 represents each commitment as a program, and a set of commitments that can hold together as a statement.

[2] It may also complement measures of form, rather than just compete with them.

[3] An *abstraction layer* is simplified language built over something more complex, like Python over C. This paper draws on Stack Theory's formalisation of abstraction layers, which involves a vocabulary of conditions, and an induced language which is a set of every set of those conditions that can hold at once (Bennett, 2025).

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.

functions *subsequent* learning can express (Bengio et al., 2013). Earlier approximations of freedom worked by testing hypothetical classifications within frozen learned representations (Bennett, 2026). Here I also freeze the representation. First, I train neural networks to classify every training example correctly, then freeze the representation, then test which additional label assignments their learned representations permit, using *linear-programming feasibility tests*. Each linear-programming feasibility test keeps a network's representation fixed and looks for a hypothetical linear final layer to replace its own. This replacement must classify every training example correctly and assign chosen labels to additional sampled inputs. It must also satisfy *output constraints*. These are tolerances, bounding how far its outputs may differ from original outputs. With a representation fixed, each requirement is a linear inequality. A linear-programming feasibility test passes if at least one final layer satisfies every inequality together. I estimate freedom as a fraction of linear-programming feasibility tests that pass. I refer to passing linear-programming feasibility tests as *feasible*, and those that aren't as *infeasible*.

A mere association between freedom and accuracy, as past work has demonstrated, leaves its practical significance unresolved. Hence I intervene through evolutionary selection. I use an approximation of freedom to decide which trained networks reproduce, to create the next generation. Their architecture and training specifications mutate, and offspring learn new representations. Repeating this process tests whether selection directed by freedom improves unseen-data performance relative to alternative selection criteria.

I compare freedom with two established measures of form, namely last-layer Laplace approximate marginal likelihood[4] and a final-layer adaptation of GdScore[5] (Immer et al., 2021; Xie et al., 2024). Freedom achieves higher mean test accuracy against both on all benchmarks tested, winning 86 of 120 pairwise comparisons[6]. Freedom also exceeds random initial selection on every benchmark, while each comparator falls below random on occasion[7]. Initially, freedom selected better-generalising networks than its comparator measures, before any offspring were generated. Then evolution tested consequences of repeatedly using freedom or its comparators to determine reproduction.

2 Freedom within learned representations

2.1 Measuring freedom

In the classification benchmarks used in this paper, the beginning layers of a neural network turn each input into a list of numbers. This learned conversion is its *representation*. Each number it produces is a *learned feature*, so a representation can be understood as a set of learned features. Then a final layer[8] follows those beginning layers, converting those representation numbers into a list of scores, with one score per class. Whichever class receives highest score is the predicted class for the input. An *assignment* gives a hypothetical class to each of several inputs, such as class three to one input and class eight to another. Freedom can be approximated by how many assignments a representation permits while existing commitments stay satisfied. A representation permits an assignment when at least one final layer realises that assignment on it. I estimate what fraction of sampled assignments each representation permits in six steps.

1. I train a network to classify every training example correctly. I then freeze its representation, while considering hypothetical replacements for its final layer.

Section 2 defines both. In this paper each condition assigns a class to an input, and holds for every final layer that realises that assignment on one fixed representation.

[4]Marginal likelihood is a probability of training labels, averaged over a prior distribution of parameters. A Laplace approximation replaces a posterior distribution with a Gaussian centred on trained parameters. *Last-layer* means I apply it to final-layer parameters only, with representation fixed.

[5]GdScore estimates accuracy on unlabelled inputs from gradient magnitudes of a network's loss with respect to final-layer weights, using its own predictions as labels. Larger gradients predict lower accuracy.

[6]A *pairwise comparison* sets freedom against one comparator in one trial. Both methods use matching training data, initial architecture plans and training-seed schedules. Freedom wins when its final selected network has higher official-test accuracy.

[7]This isn't to say freedom cannot underperform random. Indeed, I expected it would in at least some circumstances. However in these experiments at least, it consistently outperformed everything.

[8]Also called a *readout* or *head*.

2. I draw unlabelled inputs called *anchors* and record this network's predictions on them. Anchors specify which aspects of current behaviour a replacement final layer must preserve. For example, under label anchoring[9], a replacement must classify an anchor as class seven whenever its trained network predicted seven there, even when that prediction was mistaken. Each network has its own anchor predictions.
3. I draw further unlabelled inputs called *probes*, disjoint from those anchors, and randomly assign a hypothetical class to each. One linear-programming feasibility test might require class three on one probe and class eight on another. These test how flexible the representation is. Note that the real dataset labels for anchors and probes are not used at all.
4. I combine training labels, anchor requirements and hypothetical probe labels. For example, requiring class number three means requiring number three's score to exceed every other class's score. I also impose output constraints on class-score differences for reference inputs[10], restricting how far a hypothetical final layer may alter current behaviour.
5. I use a linear-programming feasibility test to check whether one replacement final layer could satisfy every requirement simultaneously. It must preserve training correctness, anchor requirements and output constraints while satisfying all requested probe labels. Different linear-programming feasibility tests may pass with different final layers. Every network keeps its trained final layer after measurement.
6. I repeat with newly sampled anchors and probes, using shared sampling rules across networks. A higher passing fraction means more tested combinations of further classifications remain possible. This fraction estimates freedom under those requirements.

Anchors connect possible further behaviours to what each trained network currently does. I use two anchor modes. *Label anchors* preserve predicted classes. *Score anchors* bound changes in differences between predicted and other class scores on anchors. When two scores are close, this bound can permit their order to reverse. Numerical bounds appear in Appendix A.

Freezing representations during each measurement lets me compare what different representations can express. Hypothetical final layers are used to approximate the freedom of a representation. Evolution subsequently changes the representations, breeding those with the highest freedom.

2.2 Statements and possible commitments

I use Stack Theory's definitions of language and extension (Bennett, 2025). Let Φ be a nonempty set of environmental states. A declarative program $p \subseteq \Phi$ specifies states in which a condition holds. A finite vocabulary $\mathfrak{v} \subseteq 2^{\Phi}$ determines a language

$$L_{\mathfrak{v}} = \left\{ l \subseteq \mathfrak{v} : \bigcap_{p \in l} p \neq \varnothing \right\}. \tag{1}$$

A statement l consists of conditions that can hold together. I take an empty intersection to be Φ. For a statement $l \in L_{\mathfrak{v}}$, its extension and freedom of function are

$$\mathrm{Ext}(l) = \{y \in L_{\mathfrak{v}} : l \subseteq y\}, \qquad w(l) = |\mathrm{Ext}(l)\,|. \tag{2}$$

An element of $\mathrm{Ext}(l)$ is a statement l' compatible with commitments in l. Due to the measure's origins in logical induction[11], freedom of function is also referred to as weakness of constraints on function. Freedom is constrained by the commitments an embodied vocabulary[12] can express.

[9] Under *label anchoring*, a replacement final layer must keep a network's predicted class on every anchor. Score anchoring, defined below, bounds changes in class scores instead.

[10] *Reference inputs* form a set drawn from training inputs and unlabelled measurement inputs. Output constraints apply to them. Section 2.3 defines this set.

[11] *Program induction* infers a program from examples of its behaviour. In *logical* program induction, candidate programs are sets of logical constraints, and a weaker set of constraints rules out fewer behaviours.

[12] A vocabulary is *embodied* when its programs are conditions one system can realise. Its language then contains every combination of commitments that system can meet together (Bennett, 2025). Here that system is a final layer acting on one fixed representation.

A *policy* is a statement used to specify permitted task[13] outputs. Generality concerns remaining correct as future demands[14] make themselves known. Among those policies known to be correct for one task, observed outputs are fixed. Greater freedom permits more combinations of *unseen* commitments. Under uniform unseen demands[15], generalisation probability increases strictly with freedom, so maximal freedom is necessary and sufficient for maximal generalisation probability (Bennett, 2025, pp. 94–95). Appendix A gives this counting argument.

Within a fixed behavioural vocabulary, function-preserving reparameterisations satisfy identical programs, giving identical statements and extensions. Freedom is therefore unchanged. Neural measurement further specifies which final-layer behaviours and probe assignments are tested.

2.3 Neural feasibility

Let $h(x) \in \mathbb{R}^d$ be a network's representation of input x, with d learned features and K classes. A trained network has class scores, called *logits*, $z_0(x) = W_0 h(x) + b_0$, where $W_0 \in \mathbb{R}^{K \times d}$ and $b_0 \in \mathbb{R}^K$. Each row of W_0 weights features for one class, and its entry in b_0 adds an offset. During measurement I fix h and vary a hypothetical final layer $z(x) = Wh(x) + b$, a weighted sum plus offsets.

For a fixed representation h, possible final layers form $\Theta_h = \mathbb{R}^{K \times d} \times \mathbb{R}^K$. Each state at this abstraction layer is one choice of weights and offsets (W, b). Let X be a finite set of measurement inputs and $\mathcal{C} = \{0, \ldots, K-1\}$ their possible classes. Write W_c for row c of W and b_c for its offset. For input x and requested class c, define

$$p^{(h)}_{x,c} = \left\{ (W, b) \in \Theta_h : \begin{array}{l} W_c h(x) + b_c > W_k h(x) + b_k \\ \text{for every } k \in \mathcal{C} \setminus \{c\} \end{array} \right\}. \tag{3}$$

This program contains every hypothetical final layer assigning class c a strictly winning score on input x. Collecting these programs gives a *feature-classifier vocabulary*[16] (Bennett, 2026),

$$\mathfrak{v}_h = \{p^{(h)}_{x,c} : x \in X,\ c \in \mathcal{C}\}. \tag{4}$$

The vocabulary is used to construct a language

$$L_{\mathfrak{v}_h} = \left\{ l \subseteq \mathfrak{v}_h \;\middle|\; \bigcap_{p \in l} p \neq \varnothing \right\}. \tag{5}$$

A member of $L_{\mathfrak{v}_h}$ is called a **statement**. Statements combine classifications that one shared final layer can satisfy. For example, inputs with identical representations receive identical scores, so a demand assigning different classes to those inputs is *infeasible*. A representation is what determines these programs and their compatible combinations. **Selecting a trained network therefore selects its induced vocabulary.** Across networks, an input–class request keeps its external meaning[17] while its representing program[18] depends on that network's representation.

Let $D \subseteq X \times \mathcal{C}$ be labelled training data, $A \subseteq X$ anchor inputs for 1 linear-programming feasibility test, and $y_0(x)$ a trained network's predicted class. Under label anchoring, its *empirical policy* is

$$\pi = \{p^{(h)}_{x,y} : (x, y) \in D\} \cup \{p^{(h)}_{a,y_0(a)} : a \in A\}. \tag{6}$$

[13] In Stack Theory, a *task* is a pair of sets of statements. One holds admissible inputs and another holds correct outputs (Bennett, 2025). A training set is one task. Generalising means remaining correct on a larger task that adds unseen inputs.

[14] A *demand* is a correct output that a larger task adds beyond those observed, such as a label for an unseen input.

[15] Unseen demands are *uniform* when every set of unseen outputs is equally likely to be added as correct outputs.

[16] A feature-classifier vocabulary has one program per input–class pair. Each program holds every final layer that realises that pair on representation h.

[17] A request's *external meaning* is which input receives which class, a description independent of any network.

[18] A request's *representing program* is $p^{(h)}_{x,c}$, a set of final layers realising that request on one network's representation h.

This is a **statement** in $L_{\mathfrak{v}_h}$ records training labels and network-specific anchor predictions. Probe classifications are further commitments tested against it. Measurement requires a chosen positive classification margin $\gamma > 0$. Training labels must satisfy

$$z_y(x) - z_k(x) \geq \gamma \quad \text{for every } (x, y) \in D \text{ and } k \neq y. \tag{7}$$

Each correct label must score at least γ above every alternative. I impose corresponding inequalities to describe the sampled anchor and probe labels. With a representation fixed, these inequalities are linear in W and b. A *linear-programming feasibility test* determines whether one choice of those weights and offsets satisfies all requirements together.

I draw a reference set R from labelled fitting inputs and additional unlabelled inputs used for anchors and probes. Output constraints apply on R. Let $\varepsilon > 0$ bound each permitted change. In centred, scale-normalised logit coordinates[19], I impose output constraints

$$\left|\{z_{y_0(x)}(x) - z_k(x)\} - \{z_{0,y_0(x)}(x) - z_{0,k}(x)\}\right| \leq \varepsilon \quad (x \in R,\ k \neq y_0(x)). \tag{8}$$

These output constraints bound changes in current class preferences on reference inputs. With unrestricted final layers, a finite strictly feasible collection[20] can be rescaled to meet any fixed positive margin. Here that rescaling can violate output constraints, so γ is part of measurement specification. An invertible affine recoding of a representation can still be undone by a corresponding change of final layer, preserving every logit. Appendix B proves this invariance and describes numerical normalisation.

I augment $\mathfrak{v}_h$ with programs expressing margin requirements, score-anchor conditions and output constraints. Each program is a set of final layers satisfying its corresponding inequality. For one linear-programming feasibility test, let l collect training requirements, anchor requirements and output constraints, and let q collect requested probe classifications with their margins. Under label anchoring, $\pi \subseteq l$. Under score anchoring, score constraints replace anchor-label programs. In this augmented vocabulary[21] $\mathfrak{v}$,

$$l \cup q \in L_{\mathfrak{v}} \quad \Longleftrightarrow \quad \bigcap_{p \in l \cup q} p \neq \varnothing. \tag{9}$$

A passing linear-programming feasibility test identifies a completion[22] $l \cup q \in \mathrm{Ext}(l)$. One outcome is recorded however many final layers satisfy its requirements. I redraw anchors between linear-programming feasibility tests, so existing commitments can change.

Freedom $w(l)$ measures extension size, as defined in 2. Because each passing feasibility test identifies a completion of l, and the extension is the set of completions of l, freedom can be approximated by the fraction of tests which pass. Appendix A specifies sampling of linear-programming feasibility tests and unresolved outcomes[23].

2.4 Calibration

To get around the problem of intractability, I calibrate a simplifying instrument for which freedom of function is both tractable, and correlates with training accuracy. Note that this instrument is calibrated using the same *labelled* training data as I use for training. On top of those I also use a small number of additional *unlabelled* inputs from the training set, for probe and anchor inputs. These are detailed in appendix. This results in an imperfect but tractable approximation of weakness. Notably, the instrument performs much better on the text benchmarks than the image benchmarks, and future work should address this discrepancy.

[19] Logits are shifted to remove a common offset and divided by a common scale, so that one value of ε permits a comparable change for every network. Appendix B specifies both steps.

[20] A collection of inequalities is *strictly feasible* when some final layer satisfies every one of them with strict inequality.

[21] Augmenting a vocabulary adds programs to it. Its language then also contains statements built from those programs.

[22] A *completion* of statement l is any statement $y \in L_{\mathfrak{v}}$ with $l \subseteq y$. It keeps every commitment in l and may add others (Bennett, 2025).

[23] An outcome is *unresolved* when a linear-programming solver returns neither a feasible solution nor a proof of infeasibility.

For each trained network, I keep its representation fixed and sample anchor inputs and probe inputs. I preserve requirements on anchor outputs and assign hypothetical labels to probe inputs. I express these requirements, training labels and output constraints as inequalities on a replacement final layer. **Each linear-programming feasibility test checks whether at least one replacement final layer could satisfy every inequality in this collection simultaneously.** A network passes when such a final layer exists. This is why anchors are important. A difficult set of anchors can stop the network passing. It separates the free from the constrained. I repeat this procedure with different sampled inputs and hypothetical labels, estimating freedom as a fraction of these linear-programming feasibility tests that pass. Approximating freedom this way requires I choose what to preserve and what to discard. **Too few constraints can allow almost every sampled combination of probe labels for every network, giving indistinguishable scores near one. Too many can forbid every sampled combination, giving indistinguishable scores near zero. The point of calibration is to find a balance, choosing measurement rules by how well their selected networks generalise to held-out training examples.**

1. I divide each trial's training data into four parts (folds), keeping class proportions similar. I train a population of networks on three parts and reserve the last for testing calibration.
2. I measure each trained network's freedom under several settings. I vary how many anchors preserve its current predictions and how many hypothetical probe labels must hold together. On text benchmarks, I also compare preserving predicted labels with preserving gaps between class scores. Each combination gives a passing fraction for that network.
3. I repeat this with each part reserved in turn, training fresh networks for every split. I do this across several trial training sets. I can then compare freedom measurements with accuracy on examples each calibration network was not trained on.
4. I construct proposed selection rules from these measurements. Some use one passing fraction directly. Others raise or lower it through a bounded adjustment based on further freedom measurements. I fit adjustment weights using accuracy on reserved training examples. I limit each adjustment so a passing fraction below one stays below one.
5. I apply every proposed rule to each trained population. Each rule chooses four parents by freedom score. I assess accuracy on reserved examples for both its first-ranked network and its four selected parents. I also penalise rules that give maximal scores to all four parents or tie selected parents with one another or with an excluded network. A fixed calibration score combines accuracy, consistency of parent choices and these penalties. I shortlist rules for confirmation.
6. I test shortlisted rules on other trial training sets, again reserving one part out of four. Within each split, I reuse one initial population across all shortlisted rules. Each chooses parents whose architecture and training specifications are mutated to generate offspring. I train offspring on their parents' three training parts, then repeat selection. Reserved examples let me check whether a rule continues selecting networks that generalise as evolution changes their representations.
7. I choose one rule per benchmark from confirmation results using that fixed scoring formula. Its measurement settings, adjustment weights and tie-breaking order stay unchanged during final experiments. I train final-run networks on complete trial training sets and use this rule for evolutionary selection. I evaluate their generalisation on official test data.

For all three text benchmarks, calibration chose a single passing fraction. KMNIST and binary Fashion-MNIST additionally use bounded adjustments from two other freedom measurements. Appendix A.3 specifies search budgets, adjustment coefficients and calibration scoring.

3 INTERVENING THROUGH EVOLUTION

In each trial I first train 48 neural networks on one labelled training set (the same one I used for calibration). Admission[24] requires complete training accuracy and positive classification margin.

[24] *Admission* means the network meets the criteria to move on to the next stage. A network has to be admitted before it is considered for selection.

Thus, admitted networks all have training-label accuracy of 1. Very little training data is used, so we can get a large observable effect for the various selection methods. The point is to get scientifically useful comparison between selection methods. Generation zero denotes this trained population. Generation one selects four parents using freedom, marginal likelihood or GdScore.

Each parent generates 12 offspring through a common mutation schedule. Offspring inherit architecture and training specifications, then train afresh. Surviving parents preserve their trained parameters. From generation two onward, selection ranks four incumbents[25] together with 48 trained offspring. Incumbents win tied scores. I evaluate whichever network ranks first at a method's final selection generation.

Pairwise methods use matching training-set allocations, initial architecture plans and training-seed schedules. Mutation rules and maximum evolutionary budgets are shared. Once parent choices diverge, offspring specifications and learned representations can diverge. This is how selection pressure intervenes on subsequent representation learning.

Marginal likelihood ranks trained final layers using a Kronecker-factored[26] Laplace approximation and a fixed prior-precision grid[27]. GdScore ranks final layers by a pseudo-label gradient score[28] on unlabelled official test inputs. Freedom measures disjoint unlabelled inputs from a training partition[29]. I draw these additional *unlabelled* inputs from each dataset's training split, excluding that trial's labelled fitting examples. These information differences[30] are part of their implemented procedures. Appendix D states both comparator formulas and their adaptation from prior work.

4 Experiments

4.1 Tasks and endpoints[31]

I tested KMNIST character recognition, binary Fashion-MNIST clothing classification, Rotten Tomatoes sentiment classification, binary AG News topic classification and binary science-versus-recreation classification in 20 Newsgroups (20NG) (Clanuwat et al., 2018; Xiao et al., 2017a; Pang & Lee, 2005; Zhang et al., 2015; Lang, 1995). The two image tasks use convolutional or multilayer perceptron networks. Text inputs are hashed unigram and bigram features, followed by learned multilayer perceptrons. Architecture and training grids are in Appendix C.

I compared twelve matched trials on each of five benchmarks, with at most twenty selection generations per trial. Every comparison pairs methods on identical training samples. Each trial uses 40 labelled examples for KMNIST, 32 for binary Fashion-MNIST, 128 for Rotten Tomatoes, 32 for binary AG News and 256 for binary 20NG. For freedom measurement, I additionally reserve one fixed unlabelled pool per trial, with 160 inputs for KMNIST, 128 for binary Fashion-MNIST, 512 for Rotten Tomatoes, 128 for binary AG News and 1,024 for binary 20NG. Table 1 reports mean test accuracy and standard deviation across trials. Each trial trains and evolves a separate network population. Trials within a benchmark share its official test set. *Random* is expected accuracy when choosing uniformly among 48 initial trained networks, averaged over twelve trials per benchmark. Table 1 gives freedom's initial-population baseline; Appendix E retains each branch's baseline where trained networks differ numerically across hardware.

[25] *Incumbents* are four networks selected as parents in a previous generation. They keep their trained parameters and compete again with offspring.

[26] A *Kronecker-factored* approximation writes a layer's curvature matrix as a Kronecker product of two smaller matrices, one over its inputs and one over its outputs. This reduces computation.

[27] *Prior precision* is inverse variance of a Gaussian prior over final-layer parameters. A *prior-precision grid* is a fixed list of candidate values for it.

[28] A *pseudo-label* is a network's own prediction, used in place of an unknown label. A *pseudo-label gradient score* measures gradients of a network's loss when pseudo-labels serve as labels.

[29] A *training partition* is a portion of a dataset's official training split. Official test inputs lie outside it.

[30] *Information differences* are differences in which data each criterion reads. Marginal likelihood reads just the labelled training data. On top of that, GdScore reads all of the unlabelled official test inputs, while Freedom only needs some unlabelled inputs from the training partition.

[31] An *endpoint* is a quantity chosen in advance to judge an experiment. Here it is official-test accuracy of each method's final selected network.

Table 1: Official-test accuracy in percent over twelve matched trials per benchmark. Parentheses contain sample standard deviations. Random averages freedom's initial-population accuracy. MargLik denotes Laplace approximate marginal likelihood.

Benchmark	Random	Freedom	MargLik	GdScore
KMNIST	44.28 (2.49)	**47.02 (3.60)**	46.86 (3.20)	32.24 (2.95)
Fashion-MNIST binary	78.23 (1.64)	**78.80 (1.57)**	78.13 (2.37)	78.14 (1.91)
Rotten Tomatoes	51.85 (0.50)	**55.99 (1.54)**	53.82 (2.05)	55.46 (1.88)
AG News binary	50.24 (0.19)	**59.84 (2.98)**	50.00 (0.01)	58.23 (3.34)
20NG binary	60.65 (1.07)	**67.22 (2.39)**	66.56 (3.83)	66.72 (2.85)

4.2 Selection before evolution

The first generation of selection just uses the learned representations, before I spawn any offspring. So generation zero is the basic networks with no selection, and generation 1 is selection applied to those networks. On generation one, freedom's mean accuracy is 56.32% on Rotten Tomatoes, 56.33% on binary AG News and 66.13% on binary 20NG, exceeding random initial selection by 4.48, 6.09 and 5.47 percentage points. Initial selection also favours freedom against both comparators on all five tasks (Appendix E). Training-data identities, initial architecture plans and training-seed schedules match across pairwise comparisons[32]. Freedom therefore identifies better-generalising networks within initial learned representations before evolutionary search changes subsequent populations.

4.3 Comparative generalisation

After evolution, freedom has higher mean test accuracy than both comparators on all five benchmarks tested. It wins 86 of 120 pairwise comparisons, with one tie and 33 losses. Each trial is compared with both controls. Comparator-specific outcomes are available in Appendix E. In any case, freedom exceeds random initial selection on every benchmark. In comparison, marginal likelihood falls below its initial-population baseline on binary AG News and binary Fashion-MNIST, and GdScore falls below its baseline on KMNIST and binary Fashion-MNIST. Freedom leads marginal likelihood on binary AG News by 9.84 percentage points and GdScore on KMNIST by 14.78 points. Thus freedom improves on arbitrary initial selection across both image and text tasks, while each comparator reverses that advantage on some tasks. I expect future work may find exceptions to this as the calibrated, approximation of freedom is far from perfect, but in these tests at least its performance was consistent. See Appendix E for generation-matched[33] and within-run[34] results, and Appendix C for early stopping rules and computation budgets.

5 Interpretation

5.1 Causal contribution

By naming this paper *freedom causes neural generalisation*, I'm making an interventional claim. I deliberately change reproductive success according to measured freedom, then compare generalisation with populations evolved using alternative criteria. Freedom determines which networks reproduce before their offspring train. Its measurement therefore participates in generating subsequent representations, rather than being computed only to explain test accuracy afterward.

Random initial selection establishes how much each criterion improves on an arbitrary choice among trained networks. Repeated reproductive selection tests what happens when that criterion also determines how subsequent networks are generated. These comparisons examine selection among learned representations and intervention on subsequent representation learning.

[32] Initial-population mean accuracies do differ slightly between freedom and controls on KMNIST and binary 20NG.

[33] *Generation matching* compares two methods at one shared generation, here each control's stopping generation.

[34] *Within-run* results compare accuracy across generations of one evolutionary run.

1. Trained representations determine which additional classifications remain jointly possible under existing commitments.
2. Approximate freedom is then used to rank those representations and select parents. Mutations of parental architecture and training specifications determine how offspring learn their representations.
3. Repeated selection changes subsequent populations. Held-out accuracy measures consequences of replacing freedom-guided evolution with marginal-likelihood or GdScore selection.

Architecture and representation change along this causal path because reproductive choices change. Holding them fixed throughout evolution would remove this route of influence. Freedom-guided evolution yields higher mean test accuracy than both alternatives on all five benchmarks tested. The mean freedom selection score also increases through evolution on all five tasks. Generation matching preserves positive accuracy differences, so differing stopping generations do not explain their sign. This evidence supports a causal contribution of selecting for operationalised[35], approximate freedom. I must emphasise that this is far from an ideal approximation, and future work can no doubt improve on its performance.

5.2 Relation to existing approaches

Representation learning concerns representations through which subsequent functions can be expressed (Bengio et al., 2013). I evaluate representations by testing further behaviours while preserving training set fit. This places freedom at a point where observed training accuracy has ceased to distinguish between the trained networks. They are all equally accurate with respect to observed requirements, but differ with respect to unseen requirements. Evolutionary neural architecture search is then used to evolve genotypes[36] through performance-directed selection (Real et al., 2019). Freedom, GdScore and MargLik are all means of predicting which networks will perform best, in order to guide evolution. Freedom measures possible behavioural commitments within a specified vocabulary (Bennett, 2025). It is a measure of function, while GdScore and MargLik are measures of form. GdScore measures parameter gradients, whose values change when a representation is recoded. MargLik uses a prior over final-layer parameters. Consistently transforming that prior preserves marginal likelihood under a coordinate change, whereas this implementation fixes an isotropic prior family in native final-layer coordinates[37]. Appendix D explains these distinctions from behavioural feasibility. They motivate cross-benchmark consistency as an empirical prediction.

6 Conclusion

Here I have demonstrated one possible neural approximation freedom of function can effectively direct evolutionary selection. Freedom exceeds random initial selection on every benchmark and achieves higher mean test accuracy than approximate marginal likelihood and GdScore on all five benchmarks tested, winning 86 of 120 pairwise comparisons. Future work should improve on this approximation, particularly in the case of vision calibration, which is where it was weakest.

AI use

I used AI to assist with coding, packaging and edits.

[35] Freedom is *operationalised* when a procedure measures it. Here that procedure computes passing fractions from linear-programming feasibility tests.

[36] A *genotype* is an inherited specification from which a network is built and trained. Here genotypes are architecture and training specifications.

[37] An *isotropic* prior has equal variance in every parameter direction. A prior *family* varies one precision value over a grid. *Native final-layer coordinates* are weights and offsets as a network stores them. Recoding a representation changes these coordinates, and an isotropic prior fixed in them does not change with it.

REPRODUCIBILITY STATEMENT

Appendices specify measurement constraints, calibration, comparators, data processing and trial protocols. Supplementary code includes training and measurement implementations, recorded results and scripts that recompute every result table.

REFERENCES

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

Michael Timothy Bennett. *How To Build Conscious Machines*. PhD thesis, The Australian National University, 2025. URL https://hdl.handle.net/1885/733782452.

Michael Timothy Bennett. Are flat minima an illusion?, 2026. URL https://arxiv.org/abs/2605.05209.

Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical japanese literature, 2018. URL https://arxiv.org/abs/1812.01718.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1019–1028. PMLR, 2017. URL https://proceedings.mlr.press/v70/dinh17b.html.

Alexander Immer, Matthias Bauer, Vincent Fortuin, Gunnar Rätsch, and Mohammad Emtiyaz Khan. Scalable marginal likelihood estimation for model selection in deep learning. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4563–4573. PMLR, 2021. URL https://proceedings.mlr.press/v139/immer21a.html.

Ken Lang. NewsWeeder: Learning to filter Netnews. In Armand Prieditis and Stuart Russell (eds.), *Machine Learning Proceedings 1995*, pp. 331–339. Morgan Kaufmann, San Francisco, CA, 1995. ISBN 978-1-55860-377-6. doi: 10.1016/B978-1-55860-377-6.50048-7. URL https://www.sciencedirect.com/science/article/pii/B9781558603776500487.

Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pp. 115–124. Association for Computational Linguistics, 2005. doi: 10.3115/1219840.1219855. URL https://aclanthology.org/P05-1015/.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4780–4789, 2019. doi: 10.1609/aaai.v33i01.33014780. URL https://ojs.aaai.org/index.php/AAAI/article/view/4405.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms, 2017a. URL https://arxiv.org/abs/1708.07747. arXiv:1708.07747.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017b. URL https://arxiv.org/abs/1708.07747.

Renchunzi Xie, Ambroise Odonnat, Vasilii Feofanov, Ievgen Redko, Jianfeng Zhang, and Bo An. Leveraging gradients for unsupervised accuracy estimation under distribution shift, 2024. URL https://arxiv.org/abs/2401.08909. arXiv:2401.08909.

Renchunzi Xie, Ambroise Odonnat, Vasilii Feofanov, Ievgen Redko, Jianfeng Zhang, and Bo An. Leveraging gradients for unsupervised accuracy estimation under distribution shift. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=FIWHRSuoos.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28, pp. 649–657. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/hash/250cf8b51c773f3f8dc8b4be867a9a02-Abstract.html.

A NEURAL MEASUREMENT

A.1 OPERATIONAL SCOPE

Equations (1)–(2) preserve Stack Theory's language and extension definitions. For a task α, let I_α be its input statements and $O_\alpha \subseteq \mathrm{Ext}(I_\alpha)$ its correct output statements, where extension of a set denotes a union of extensions. Correct policies are

$$\Pi_\alpha = \{\pi \in L_{\mathfrak{v}} : \mathrm{Ext}(I_\alpha) \cap \mathrm{Ext}(\pi) = O_\alpha\}. \tag{10}$$

A correct policy permits all correct outputs and excludes incorrect outputs for those inputs. For any correct policy π, finiteness and this equality give

$$w(\pi) = |O_\alpha| + |\mathrm{Ext}(\pi) \setminus \mathrm{Ext}(I_\alpha)\,|. \tag{11}$$

Correct policies permit identical known outputs. Freedom can differ through further commitments outside known inputs (Bennett, 2025).

Let $U = L_{\mathfrak{v}} \setminus \mathrm{Ext}(I_\alpha)$ be unseen statements and $B_\pi = \mathrm{Ext}(\pi) \cap U$ those permitted by a correct policy π. A later demand $S \subseteq U$ is satisfied when $S \subseteq B_\pi$. If S is uniform over all subsets of U, then

$$\Pr(S \subseteq B_\pi) = \frac{2^{|B_\pi|}}{2^{|U|}} = 2^{w(\pi)-|O_\alpha|-|U|}. \tag{12}$$

There are $2^{|B_\pi|}$ compatible demand sets among $2^{|U|}$ possibilities. Each additional permitted unseen statement doubles how many demand sets remain possible. Since $|O_\alpha|$ and $|U|$ are fixed, maximal generalisation probability requires and follows from maximal freedom. This finite-demand calculation expresses generality as compatibility with future requirements. Classification experiments test whether a neural approximation guides selection toward greater accuracy.

Section 2.3 defines classification programs using strictly winning scores, then states positive margins and output restrictions separately. For finite measurement inputs, these additional inequality programs extend a finite feature-classifier vocabulary. Each passing query witnesses a completion of its specified requirements. Sampling fixes assignment sizes, a distribution over demands and permitted output changes, so passing fractions estimate feasibility under that query design.

Equation (12) assumes policies correct for one common task. Neural comparisons use common external meanings for requested input–class pairs, interpreted in each learned feature-classifier vocabulary. Each network's anchors impose its own predictions or score constraints. Applying that common-task argument across these readings additionally requires compatible anchor commitments and a task construction establishing policy correctness.

A.2 QUERIES AND UNRESOLVED FEASIBILITY

Source probes are unlabelled training-partition inputs disjoint from a trial's labelled set and validation-control panel. KMNIST reserves 160 probes, binary Fashion-MNIST and binary AG News each reserve 128, Rotten Tomatoes 512, and binary 20NG 1,024. Each query samples anchor inputs and further input–label assignments. Labels in those assignments are artificial commitments, not inspected ground truth. Label-anchor mode preserves current predicted classes. Score-anchor mode, available on all three text tasks, bounds predicted-versus-other logit differences with tolerance $10^{-6} + 0.025|g_0|$, where g_0 is a current logit difference. Both image tasks use label anchors only. Queries use positive classification margin 10^{-5}.

Geometry is derived from nominal feature dimension plus one affine-bias coordinate. Anchor fractions are $1/8$, $1/4$, and $1/2$ of that dimension, rounded upward. Assignment-size fractions are $1/64$, $1/32$, and $1/16$, rounded upward and capped at eight. Both image tasks consequently use anchor numbers $5, 9, 17$ and assignment sizes $1, 2, 3$. Binary AG News uses anchor numbers $9, 17, 33$ and assignment sizes $2, 3, 5$. Rotten Tomatoes and binary 20NG use anchor numbers $17, 33, 65$ and assignment sizes $3, 5, 8$. Radius $\varepsilon = 0.35$ is shared across benchmarks. Geometry remains common across networks within a benchmark.

A cell is one combination of anchor mode, anchor number and probe-label group size. Each cell uses 256 queries. Let F, J and U denote feasible, infeasible and unresolved outcomes from $N =$

$F + J + U$ queries. When $F + J > 0$, a cell score and numerical-resolution interval are

$$p = \frac{F}{F+J}, \qquad p_{\text{low}} = \frac{F}{N}, \qquad p_{\text{high}} = \frac{F+U}{N}. \tag{13}$$

Unresolved queries are excluded from a cell's point estimate and included in its resolution bounds. These bounds concern unresolved outcomes in sampled queries, rather than sampling confidence. The implementation records F, J, U, N and numerical diagnostics during execution; the supplied result extracts retain calibration summaries and selected-network scores and signatures.

Implementation is in `wam/surf.py`, `wam/fspace.py`, `wam/fgeom.py`, and `wam127_partc_weakm/full.py` and `lp.py`. Linear-programming shortcuts reuse feasible witnesses and certified infeasibility where supported.

A.3 Frozen calibration

Four-fold cross-fitting uses trial training data. Stage one uses trials zero through seven, giving 32 cross-fit sources. It evaluates 24 measurement rules and advances four rules to confirmation. Stage two uses trials eight through eleven, giving 16 sources. Within each split, four shortlisted rules start from one shared trained population. Confirmation comprises initial parent selection followed by two rounds of offspring training and selection, ending at generation three. Final networks subsequently train on complete trial training sets. Cross-fitting adds no distinct labelled calibration panel, but calibration reuses labels across final trials. Per-trial sample sizes therefore describe final-network training, not total label exposure of a benchmark-wide calibrated procedure.

Let $p \in [0, 1]$ be a primary cell's passing fraction. Let f_j be further measurement features, μ_j their calibration centres, $\sigma_j > 0$ their scales, a_j fixed weights and c a correction cap. Selection uses

$$s = p + c \min(p, 1-p) \tanh\left(\sum_j a_j \frac{f_j - \mu_j}{\sigma_j} \right). \tag{14}$$

A bounded adjustment combines evidence from other feasible fractions. With $c = 0$, selection uses p directly. Rules use at most two adjustment features and caps $c \in \{0, 0.25, 0.5, 0.75\}$. I fit adjustment weights using reserved-fold accuracies in stage one and freeze them before confirmation. Calibration fixes every coefficient before evolutionary evaluation.

Table 2: Frozen primary measurement settings. Anchor numbers and assignment sizes identify cells defined above. Both image tasks use a residual correction in Eq. (14).

Benchmark	Anchors	Anchor inputs	Assignments	Cap
KMNIST	Label	9	3	0.75
Fashion-MNIST binary	Label	9	2	0.5
Rotten Tomatoes	Label	65	8	0
AG News binary	Score	9	5	0
20NG binary	Label	33	5	0

KMNIST's adjustment uses its label-anchor cell with nine anchors and assignment size one, and its aggregate label-anchor feature at nine anchors. Binary Fashion-MNIST uses the aggregate lower-bound feature `label_bound` and the assignment-size-three feature `label_o3`. Table 3 gives the adjustment coefficients. Complete coefficients, centres, scales and tie settings appear under `calibration.f_b` in each benchmark's `f_b.json` result file. Raw feasible totals resolve primary-score ties from larger assignment sizes toward smaller sizes, using frozen anchor and mode ordering. `wam/partf_core.py` implements Eq. (14) and these rules.

Calibration ranks rules using a fixed weighted score. I add mean first-ranked accuracy, 0.25 times mean parent accuracy, 0.10 times its tenth percentile across splits, 0.05 times best-network inclusion rate, and 0.05 times selection stability. Best-network inclusion means selecting at least one network with maximal reserved-example accuracy within its comparison population. Stability measures overlap in parent choices across grouped evaluations during stage one, defaulting to zero when unavailable. During confirmation, stability is mean surviving-parent fraction between generations.

I subtract 0.15 times how often all four parents reach primary score one, an additional 0.35 times how often all their tie-breaking measurements also reach maximum, and 0.10 times how often rankings contain ties within selected parents or across selection's cutoff. These ties are assessed after applying measurement-based tie-breaks. Confirmation accuracy and inclusion use final selections, while ceiling and tie rates average across selection generations. `calibration_utility_values` in `wam/partf_core.py` implements this score. Search budgets and scoring weights are shared across benchmarks.

For binary AG News and binary Fashion-MNIST only, calibration fitting allows two additional 140-epoch finishing phases at learning-rate multipliers 0.1 and 0.01. A network enters calibration selection only after classifying every fitted training example correctly with minimum margin above 10^{-8}. An exhausted slot remains ineligible without replacement. The ordinary attempts remain unchanged, and this policy applies only to calibration, leaving production training and both controls unchanged. The other three freedom profiles use ordinary calibration fitting.

Table 3: Frozen adjustment parameters in Eq. (14). Displayed values are rounded; the supplied result files retain full precision.

Benchmark	Feature	a_j	μ_j	σ_j
KMNIST	`label_r9o1`	0.61598866	0.89843750	0.02316562
KMNIST	`label_r9`	-0.78775502	0.82421875	0.02371077
Fashion-MNIST binary	`label_bound`	0.73652309	0.47020449	0.12973043
Fashion-MNIST binary	`label_o3`	-0.67641240	0.47716503	0.13272961

B AFFINE RECODING

Proposition 1. *Fix queried inputs, additional label assignments, training labels and behavioural constraints expressed through logits or logit differences. Let $A \in \mathbb{R}^{d\times d}$ be invertible and $a \in \mathbb{R}^d$. Replacing features $h(x)$ by $\widetilde{h}(x) = Ah(x)+a$, with unrestricted affine readouts in both coordinates, preserves feasibility of every such query.*

Proof. For any readout (W, b), define

$$\widetilde{W} = WA^{-1}, \qquad \widetilde{b} = b - WA^{-1}a. \tag{15}$$

Then $\widetilde{W}\widetilde{h}(x)+\widetilde{b} = Wh(x)+b$ for every input x. Conversely, set $W = \widetilde{W}A$ and $b = \widetilde{W}a+\widetilde{b}$. These transformations are inverse maps between affine readouts and preserve all queried logits. Every training, anchor, neighbourhood and assignment inequality has its previous truth value. Hence a satisfying readout exists in one coordinate system if and only if a satisfying readout exists in its recoding. □

An invertible change of feature coordinates changes how a readout is written while preserving possible behaviours. Equal query outcomes follow without imposing a prior or measuring parameter-space volume.

Numerically, `canonical_function_arrays` removes class-common logit offsets, normalises centred logit scale, and constructs affine feature coordinates using a singular-value decomposition and pivoted factorisation. It rounds canonical arrays to twelve decimal places and checks logit reconstruction within 5×10^{-9}. Finite rank thresholds and solver tolerances can affect borderline cases. Proposition 1 concerns mathematical feasibility for fixed behavioural constraints, rather than bitwise equality of all numerical implementations. Reference-input identities and query rules must stay fixed when comparing recodings.

C TRAINING AND EVOLUTIONARY PROTOCOL

C.1 DATASETS

KMNIST uses torchvision's training and test partitions, with 60,000 training images and 10,000 test images (Clanuwat et al., 2018). Each trial uses 40 labelled training images. Its input representation resizes to 32×32 and uses 32 learned feature coordinates.

Binary Fashion-MNIST uses original classes zero (T-shirt/top) and six (Shirt), mapped to zero and one (Xiao et al., 2017b). Its official binary splits contain 12,000 training images and 2,000 test images. Each trial uses 32 labelled training images. Inputs retain their native 28×28 dimensions, with pixel values divided by 255, and networks learn 32 feature coordinates. The class-defined test cohort is fixed independently of model performance.

Rotten Tomatoes uses `cornell-movie-review-data/rotten_tomatoes`, revision `aa13bc287fa6fcab6daf52f0dfb9994269ffea28`, associated with Pang & Lee (2005). Each trial uses 128 labels, balanced 64 per class. Training-text deduplication normalises case and whitespace, excludes empty texts, and preserves first occurrences. Official test evaluation uses all 1,066 rows. Official validation data are unused. Dataset revision and source-allocation rules are in `wam/rotten_tomatoes.py` and `wam/tasks.py`.

Binary 20 Newsgroups maps `sci.crypt`, `sci.electronics`, `sci.med` and `sci.space` to class zero, and `rec.autos`, `rec.motorcycles`, `rec.sport.baseball` and `rec.sport.hockey` to class one (Lang, 1995). Loading uses scikit-learn's training/test splits, removes headers, footers and quotes, and excludes empty documents. Recorded data contain 4,586 training documents and 3,032 test documents. Each trial uses 256 labels. This study tests binary science-versus-recreation classification.

Binary AG News uses original classes zero (World) and one (Sports), with 60,000 training articles and 3,800 test articles (Zhang et al., 2015). Each trial uses 32 labels. The dataset revision is `eb185aade064a813bc0b7f42de02595523103ca4`. Its loader uses original categories solely to form the predeclared binary cohort and discards test labels before returning inputs to a scorer.

All three text tasks use hashed unigram and bigram inputs with lowercase text, nonnegative hashing and Euclidean normalisation. Rotten Tomatoes and binary 20NG use 20,000 input coordinates and 128 learned feature coordinates; binary AG News uses 12,000 and 64. A learned multilayer perceptron maps the hashed inputs through two width-sized hidden layers to these feature coordinates, followed by a linear classifier. Rectified linear units follow hidden layers. Hashing is fixed preprocessing. Feature extraction beyond hashing is learned.

C.2 NETWORK TRAINING

Image genotypes include a convolutional network, an image multilayer perceptron and a shallow convolutional network. Text genotypes use `text_mlp`. Width, learning rate, weight decay, batch size and dropout mutate within Table 4. Source functions in `wam127_partc_weakm/net.py` specify every layer. Matrix parameters use Muon and auxiliary parameters use momentum stochastic gradient descent. These optimizer choices are fixed across selectors. Admission requires full training accuracy and positive margin. Source specifies restart and stopping behaviour for unsuccessful training attempts.

C.3 EVOLUTION AND BUDGETS

Each trial starts with 48 independently trained networks. First selection chooses four parents. Each produces 12 offspring. Mutation slots comprise an unchanged specification with a new training seed, increases and decreases in width, learning rate, weight decay, batch size and dropout, and an architecture change. Adjacent grid mutations reflect at boundaries. Where architecture has only one choice, its mutation increases width and learning rate. Fixed coordinates remain unchanged.

Four incumbents and 48 offspring compete after first selection. At most 20 selection generations correspond to 960 network trainings per trial, excluding calibration. Calibration's maximum is

Table 4: Training-specification grids. Image settings apply to both image tasks. RT denotes Rotten Tomatoes. Epoch and attempt limits apply to ordinary network-training procedures; calibration finishing is specified in Appendix A.3.

Setting	Images	RT and binary 20NG	Binary AG News
Width	192, 256, 384, 512	256, 384, 576	192, 256, 288
Learning rate	$0.0003, 0.001, 0.002$	$0.001, 0.002, 0.003$	$0.0005, 0.001, 0.0015$
Weight decay	$0, 10^{-5}, 10^{-4}, 10^{-3}$	$0, 10^{-5}, 10^{-4}$	$0, 10^{-5}, 10^{-4}$
Batch size	32, 64, 128	16, 32, 64	32, 64, 128
Dropout	$0, 0.05, 0.10$	0	0
Epoch limit	280	320	240
Attempt limit	8	10	10

8,448 trainings per benchmark, compared with 11,520 final-run trainings across twelve trials. These are protocol maxima, rather than measured wall-clock equivalences. Saturation and comparator stopping can reduce executed work.

Trial task seeds are $91001 + 1000003t$ for trial index $t \in \{0, \ldots, 11\}$. Initial plans, training seeds and mutation-slot schedules match across methods. Parent choices determine subsequent genotypes, so matching seeds does not imply identical later networks.

Binary 20 Newsgroups control stopping generations are

$$\begin{aligned} &\text{MargLik} \quad 20, 14, 12, 8, 11, 16, 14, 6, 16, 16, 12, 15, \\ &\text{GdScore} \quad 14, 7, 10, 20, 10, 7, 12, 8, 8, 11, 7, 5. \end{aligned}$$

Binary 20NG controls stop after three generated generations without strict improvement in their ranked scores. On the other four tasks their patience is twenty, and every control reaches generation 20. Freedom reaches generation 20 in all sixty trials; its stopping rule permits earlier termination only at full measurement saturation. Generation-matched comparisons read freedom's recorded endpoint at each corresponding control stop. This aligns evolutionary depth without altering calibration, scorer inputs or training costs.

D Comparator implementations

D.1 Approximate marginal likelihood

MargLik uses a last-layer, Kronecker-factored Laplace calculation motivated by neural marginal-likelihood selection (Immer et al., 2021). Let $X \in \mathbb{R}^{n\times(d+1)}$ contain training features augmented with a bias coordinate, $P \in \mathbb{R}^{n\times K}$ their softmax probabilities, and $\theta \in \mathbb{R}^{K(d+1)}$ flattened trained head parameters. Define

$$A = \frac{X^\top X}{n}, \qquad B = \frac{\operatorname{diag}(\sum_i P_{i,:}) - P^\top P}{n}, \qquad H = n(B \otimes A). \tag{16}$$

Here $\otimes$ is a Kronecker product, and H approximates head curvature. Let λ_j be its eigenvalues, let $q = K(d+1)$, and let $\ell(\theta)$ be training negative log likelihood. For precision $\alpha > 0$, scoring uses

$$M(\alpha) = -\ell(\theta) - \frac{\alpha}{2}\|\theta\|^2 + \frac{q}{2}\log\alpha - \frac{1}{2}\sum_{j=1}^{q}\log(\lambda_j + \alpha). \tag{17}$$

I rank by maximum $M(\alpha)$ over 49 logarithmically spaced precisions from 10^{-6} to 10^6. Each score uses a fitted representation and fitted head. Head parameters are not refitted separately for every precision. This is a plug-in approximate-evidence selector.

Marginal likelihood integrates likelihood against a prior measure. A coordinate change preserves that integral when its prior measure is transformed consistently. Here α specifies an isotropic Gaussian prior in native head coordinates. General affine recodings can transform an isotropic prior into an anisotropic prior outside this family. Freedom's feasibility test preserves behavioural requirements directly.

D.2 GdScore

GdScore adapts gradient-based unsupervised accuracy estimation (Xie et al., 2025) to a fitted final linear layer. On unlabelled official test inputs, compute softmax probabilities P and pseudo-label matrix Y. A prediction with maximum probability at least 0.5 receives its predicted class. Lower-confidence predictions receive a seeded random class. On binary tasks, maximum probability is at least 0.5.

For m feature vectors, form a batch matrix X without a bias coordinate. Compute classifier-weight gradient

$$G = \frac{(P - Y)^\top X}{m}, \qquad q(G) = \left(\sum_{j,k} |G_{jk}|^{0.3} \right)^{1/0.3}. \tag{18}$$

This is an entrywise quasi-norm. Smaller average $q(G)$ is preferred. Batches have size 128, with equal averaging over batches including a final partial batch. Random seed one is reset for each network. Bias gradients are excluded. Scoring performs no network update. `wam/partf_comparator_scores.py` implements both comparators.

For any $c > 0$, replacing features X by cX and classifier weights W by W/c preserves logits and pseudo-labels. However, G becomes cG and $q(G)$ becomes $c\,q(G)$. Thus function-preserving feature scaling changes this gradient score, while Appendix B establishes unchanged freedom feasibility under corresponding recodings.

E Trajectory evidence

Random initial-pick accuracy is population-mean test accuracy averaged across twelve initial populations per benchmark. All evolved comparisons use these same twelve trial allocations. Branch-specific initial means are retained: freedom and control means are 44.2798% and 44.2107% on KMNIST, and 60.6526% and 60.6629% on binary 20NG. The other three benchmarks have identical initial means across methods. Matching data, architecture plans and training seeds does not ensure numerically identical trained networks across accelerator and library combinations. Supplementary calculations reproduce these means and first-pick gains from unrounded measurements.

Table 5: Mean paired accuracy differences in percentage points over 12 trials. Initial selection uses generation zero's trained population, before offspring. Generation matching uses each control's stopping generation.

Benchmark	Comparator	First pick	Final	Gen.-matched
KMNIST	MargLik	+1.23	+0.16	+0.16
KMNIST	GdScore	+9.60	+14.78	+14.78
Fashion-MNIST binary	MargLik	+0.80	+0.67	+0.67
Fashion-MNIST binary	GdScore	+1.05	+0.65	+0.65
Rotten Tomatoes	MargLik	+2.46	+2.17	+2.17
Rotten Tomatoes	GdScore	+0.77	+0.53	+0.53
AG News binary	MargLik	+6.33	+9.84	+9.84
AG News binary	GdScore	+3.33	+1.61	+1.61
20NG binary	MargLik	+0.45	+0.66	+0.41
20NG binary	GdScore	+0.86	+0.51	+0.48

Generation matching compares freedom with each control at that control's final selection generation. Generation choice depends on a recorded stopping rule rather than test accuracy. Positive mean differences persist against both controls on all five benchmarks. This aligns selection generations while leaving calibration costs and scorer inputs unchanged.

Freedom's selected mean score rises on all five tasks. Mean accuracy rises on KMNIST, binary AG News and binary 20NG, and declines slightly on Rotten Tomatoes and binary Fashion-MNIST. Higher measured freedom and higher accuracy therefore need not move together between every pair of generations. Comparative endpoints test selection procedures over their recorded runs.

Table 6: Freedom-selected trajectories on all five benchmarks. Values are means across twelve trials. Subscripts denote selection generations. Text scores are feasible fractions; image scores include their frozen bounded adjustments. Accuracy changes are percentage points.

Benchmark	s_1	s_{20}	Acc.$_1$	Acc.$_{20}$	Δ acc.
KMNIST	0.8883	0.9000	45.56	47.02	+1.47
Fashion-MNIST binary	0.7837	0.8417	78.90	78.80	-0.10
Rotten Tomatoes	0.6386	0.6979	56.32	55.99	-0.34
AG News binary	0.3138	0.4836	56.33	59.84	+3.51
20NG binary	0.8446	0.8634	66.13	67.22	+1.10

Table 7: Paired wins, ties and losses for freedom relative to each comparator at final recorded endpoints. Every row includes twelve matched trials.

Benchmark	Comparator	Wins	Ties	Losses
KMNIST	MargLik	5	0	7
KMNIST	GdScore	12	0	0
Fashion-MNIST binary	MargLik	7	0	5
Fashion-MNIST binary	GdScore	8	1	3
Rotten Tomatoes	MargLik	11	0	1
Rotten Tomatoes	GdScore	9	0	3
AG News binary	MargLik	12	0	0
AG News binary	GdScore	7	0	5
20NG binary	MargLik	7	0	5
20NG binary	GdScore	8	0	4

Paired differences reuse each trial's training-set allocation. A common calibration is shared across freedom trials, and official test sets are reused. I report descriptive paired results without asserting independence of all trajectories or converting sample standard deviations into inferential claims.

E.1 Validation-selected reference

A separate labelled panel of 512 examples selects once from each freedom branch's initial population, then stops. Table 8 reports the selected network's official-test accuracy. This reference uses additional labelled information; freedom calibration instead cross-fits the trial training sets. It is a descriptive first-selection reference, while MargLik and GdScore are the evolutionary controls.

Table 8: Official-test accuracy of the validation-selected initial network, in percent over twelve trials. Parentheses contain sample standard deviations.

Benchmark	Validation-selected test accuracy
KMNIST	48.69 (2.52)
Fashion-MNIST binary	79.25 (1.51)
Rotten Tomatoes	55.42 (1.85)
AG News binary	56.13 (4.06)
20NG binary	67.77 (2.11)

F Reproduction

Supplementary `code.zip` contains source, all fifteen twelve-trial result files, and table-reproduction and verification scripts. Its `README.md` lists dataset preparation, numerical environments and commands for rerunning each benchmark and method. `report_results.py` recomputes the descriptive summaries using only Python's standard library. The result extracts provide trial-level accuracies, trajectories, selection scores and frozen calibration; trained checkpoints, datasets and per-example predictions are generated by rerunning the supplied implementation.

Recorded numerical environments are retained in each result file. Freedom's binary 20NG branch used MPS while its controls used CUDA. KMNIST's freedom branch also has a different recorded numerical environment from GdScore, and KMNIST MargLik's library versions were not recorded. These comparisons share data and training plans, rather than a guarantee of bitwise-identical initial networks. The supplementary requirements distinguish the recorded environments where available.

G BEYOND CLASSIFICATION

Behavioural possibilities connect current correctness with future adaptation. A system that satisfies present commitments while permitting further commitments has options for responding to additional requirements. Stack Theory formalises this relationship through extensions of statements (Bennett, 2025). Linear-programming feasibility tests give this account an experimental implementation. Selection then tests whether prioritising those possibilities improves performance on unseen observations.

Freedom now has an executable role[38] in selecting neural systems that generalise. This connects a theory of compatible future commitments to changes in learned behaviour. Applying that connection beyond classification requires identifying each system's existing commitments, its possible further behaviours and a process that selects among them.

[38] Freedom has an *executable role* when a procedure computes it and acts on it, here by choosing which networks reproduce.