Isotropic Activation Functions Enable Deindividuated Neurons and Adaptive Topologies

George Bird
Department of Computer Science
University of Manchester
Manchester, UK
george.bird@postgrad.manchester.ac.uk

Abstract

Varying the number of neurons while preserving function is difficult because neural networks' dense connectivity creates strong computational interdependence between units. Elementwise activation functions, particularly their limited equivariance properties, are identified as a central hurdle to achieving this. We propose activation functions designed around chosen equivariances, enabling dense neural networks to adapt via singular value decomposition of their affine maps. This diagonalises layers into one-to-one, ordered connections, making it easier to assess the impact of individual connections on function. Low-impact neurons can be removed (neurodegeneration), and a thresholded buffer of largely inactive 'scaffold' neurons is maintained (neurogenesis). These symmetry-led diagonalisation and structural changes prioritise function-invariance: neurogenesis exactly preserves function, neurodegeneration is asymptotically function-preserving with the pruned singular-value threshold. Moreover, this construction enables asymptotic 50% parameter sparsification of dense networks while preserving function analytically. Thus, the approach enables real-time restructuring of the architecture in response to task demands, task appending, removal or changes. Conceptually, this motivates broader applications of symmetry prescriptions at the small scale, even when no data-driven model-scale symmetry adherence is required. Following this, we build activations around symmetry, not from classical assumptions about individual neurons. Unlike typical elementwise forms, these alternatives feature explicit basis independence and do not individuate neurons. We leverage this to decompose and reinterpret neurons across sequential layers, simplifying structure and providing a natural route to adaptive topologies. Additional developments include a new tunable parameter, the 'intrinsic length', new possibilities for interpretability and real-time monitoring, and a parallelised precomputation of matrix-vector products.

1 Introduction

Neural plasticity in the quantity and connectivity of neurons is biologically advantageous [1–3] and routinely occurs during mammalian brain development [4–7]. Shown to improve neural efficiency through controlled pruning [8, 9], with a benefit to initial overabundance [9], whilst enabling the accumulation of knowledge [3, 7], improving robustness [10] and function through growth [11, 7]. Hence, it is hypothesised that analogous behaviour within artificial neural networks will be similarly beneficial. The practical consequences of a network that can reliably adapt its architecture and connectivity in a real-time controlled manner may be transformative — and this constitutes a broad goal of architecture-based continual learning approaches [12].

Contemporary primitives, e.g. activation functions, many initialisations, normalisers, optimisers, and more, are typically constructed using elementwise functional forms. Historically, forms were inspired

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

by biological neural systems [13]: "*[...] the conformity of their relations to those of the logic of propositions, insure that the relations of psychons are those of the two-valued logic of propositions*", an initial framing helping to naturalise the eventual pervasive implementation of activation vectors being decomposed in a basis-dependent manner, interpreted as individual artificial 'neuron'-like units composed with scalar operations. Modern elementwise formulae apply univariate maps independently to components; a construction generalising from this original individuated neuron-derived premise.

This duality of individuated yet interdependent neuron construction makes adapting architectures while closely preserving function challenging—especially for neurodegeneration. This is one reason connectivity pruning [14–20], rather than highly-connected whole-neuron removal [21–26], is typical.

This 'individuation' limitation is inherent to most contemporary primitives, with few exceptions. In particular, those classified by their equivalence class under reparameterisation [27, 28] admitting permutation-structured groups, e.g., permutation, hyperoctahedral, scaled-permutations, etc. [29]. We posit that generalising primitive construction instead through a broader symmetry-*prescribed*[1] approach is beneficial [30–32]. Exploring this allows novel functional forms and generally new behaviours, specifically via intentionally introduced reparameterisation invariances in the present work. These symmetry-prescribed primitive reformulations are central to the adaptive topology approach presented: replacing the typical, derived permutation symmetry with a chosen alternative.

Using basis-independent formulations as the goal, one can replace the discrete permutation symmetry with a continuous one. Namely, *orthogonal equivariance* symmetry naturally serves as a foundation for such altered activation functions. This is because this particular choice of continuous group prescription still permits non-linear functions on the magnitude, making it well-suited for activation functions; meanwhile, the orthogonal group is a supergroup of the permutation group, $\mathrm{O}(n) \supset S_n$, all while it ensures basis-independence by being direction-agnostic ('isotropic') in effect. Hence the orthogonal groups offer a suitably minimal and natural generalisation for such reformulations. *Emergent* from this symmetry redefinition is a more generalised concept of neurons, which are 'deindividuated', since no function-distinguished basis exists for decomposition [33, 30, 32].

Enabled by this is our methodology for dynamic topologies. It builds on the observation that these orthogonal-equivariant, symmetry-derived reformulations, termed "isotropic primitives", commute by definition with orthogonal-group actions. Considering this in tandem with singular value decomposition (SVD) for dynamical networks, or independent component analysis (ICA) [34–36] for interpretable networks, both of which yield standard orthogonal matrices that only now commute due to the symmetry-led design. Combining SVD's properties thus enables a diagonalised topology by reparameterising the networks with its orthogonal matrices into a simpler one-to-one connectivity. This simplifies the complexity of full neurodegeneration by *making it equivalent to connectivity pruning*. Similarly, an ICA representation derived from dataset activations may better represent a network for interpretability and real-time monitoring, up to ICA's own concept resolvability.

This diagonalisation procedure incurs *no functional degradation* analytically, owing to the primitive's basis independence. In practice, only machine precision, and propagation thereof, cause negligible alteration. Adaptive topologies are then achieved by thresholding these diagonalised entries to trigger real-time adaptations in response to task demand and learning. These are exactly function-preserving for neurogenesis and, given by the thresholding, arbitrarily well approximated for neurodegeneration. Hence, equivalence classes expand to full architectural adaptations, not just static reparameterisations.

Overall, this is unique to continuous-symmetry-prescribed primitives, and it is not possible for classical elementwise functions constrained to discrete parameter relabelling. Hence, we demonstrate flexible use cases for these alternative isotropic formulations, whilst motivating a broader exploration of primitive symmetry prescription for benefits in *general* practice. The primary contributions are:

1. **Conceptual Contribution**: Functional forms are historically derived from individual neurons as fundamental units to compose; from this, symmetries are *since deduced*. Instead, we propose an ontological reversal: treating symmetry as fundamental and neurons as emergent. Adopting symmetry-prescriptive foundations offers an expanded class of admissible functional forms that underpin *neural* learning systems, unconstrained by biological cell-like conceptions and yielding generalised notions of neurons induced by group representations. *Eqns.* 1 to 3 formalise this in an organising taxonomy for current and novel functional forms

[1] 'Prescribed' indicates intentional introduction of a target symmetry for a purpose, contrasting to studying those 'deduced' in a pre-existing form. We motivate this utility even at primitive-scale, without requiring data-derived model-scale symmetries.

by their descriptive and prescriptive symmetries. Induced effects may be assigned to and compared across categories or their compositions, offering utility for general applications by understanding symmetry-driven properties and biases. This constitutes the broader objective that the adaptive architectures begin to pursue and evidence.

2. **Reformulation Contribution**: With this enriched design space, isotropic primitives are designed to exhibit a crucial basis independence absent in elementwise formulae.
3. **Practical Contribution**: Leveraging this allows affine maps to be represented diagonally, with singular-value thresholding triggering architecture changes, alongside asymptotic 50% perfect sparsification, ICA representations and parallelised matrix-vector products.

Related Work: Various approaches to architecture adaptation exist, including neuroadaptive-like methods [21–26] and also connectivity-pruning [17–20, 37]. The former represent the closest related work, varying the number of neurons, not solely connectivity; examples include Net2Net [22], Network Morphism [24] and Neuron Merging [26]. Net2Net considers neurogenesis only, through weight-column duplication in elementwise networks, showing function preservation when outgoing weights sum to the original; it also considers Net2DeeperNet as a method to append additional layers invariantly—Network Morphism extends this work, generalising both approaches. Neuron Merging considers an analogous case for neurodegeneration only and is restricted to solely ReLU activations. Moreover, due to distinctive primitives, the present approach bears limited similarity to these prior implementations. Works such as SNIP [19] and RigL [20] diverge more considerably by addressing connectivity sparsity instead, showing network performance can be maintained with significantly fewer connections rather than adapting width through neuron count. Separately, prior work [38] arrived at a related use of orthogonal equivariant activations, studied for the development of a special-case Step-ReLU universal approximation theorem. They also identified a markedly different exploitation of basis change via QR decomposition and preactivation span to obtain a distinct form of model compression [38]. This enriches the potential utility of such altered activations, and their compression appears to compound with our distinct connectivity-sparsification capabilities, and may aid in streamlining successive layers in neuroadaptation. *App.* J compares all these works in depth.

2 Theoretical Background

An overview of the symmetry-prescribed framework is provided, and it is then used to define isotropic activation functions that commute with orthogonal-group actions. Finally, layerwise diagonalisation and sparsification are detailed, culminating in the adaptive topology procedure. This is detailed for multilayer perceptrons, whilst *App* E.1 presents a conceptual generalisation for convolution[39, 40].

2.1 From Symmetry Taxonomy to Primitive Reformulation: Isotropic Activation Functions

Proposed is a broader organising taxonomy of function (including up to model-scale) symmetries. This is formalised via groups, $\mathcal{G}$, with their elements, $g \in \mathcal{G}$, acting on functions, f of a (potentially singleton) functional class $\mathcal{F}$ through various matrix representations $\mathbf{g} = \varrho^{(i)}(g)$, in categorisable ways. These are expressed as three non-exhaustive relations *Eqns.* $1 \Rightarrow 2 \Rightarrow 3$. These detail different 'strength' symmetry adherence over the functional class, termed "algebraic", "probabilistic" and "closure". Algebraic is more standard [29, 41], but is unconventional in its application over a *functional class*. *Eqn.* 2's probabilistic category is unnecessary for adaptive networks but provided for completeness, with intentionally flexible probability measure $\mathbb{P}$, capable of classifying initialisations.

$$\text{Algebraic:} \qquad \forall g \in \mathcal{G},\ \forall f \in \mathcal{F}: \quad \varrho^{(1)}(g) \circ f \circ \varrho^{(2)}(g^{-1}) = f \tag{1}$$

$$\text{Probabilistic:} \qquad \forall g \in \mathcal{G},\ \forall f \in \mathcal{F}: \quad \mathbb{P}\Big(\varrho^{(1)}(g) \circ f \circ \varrho^{(2)}(g^{-1})\Big) = \mathbb{P}(f) \tag{2}$$

$$\text{Closure:} \qquad \forall g \in \mathcal{G},\ \forall f \in \mathcal{F}: \quad \varrho^{(1)}(g) \circ f \circ \varrho^{(2)}(g^{-1}) \in \mathcal{F} \tag{3}$$

These taxonomise and structure the design space, assisting comparisons of symmetry-induced properties, utilisable behaviours and structural biases, both individually and in composition. Through careful representation choices, relations can serve as a foundation for activation functions, normalisers, initialisers, optimisers and further primitives, categorised by underlying symmetry. This facilitates novel prescriptive use and use as composable building blocks. In isolation, these do *not* describe

reparameterisation invariances (parameter symmetries), which typically require composition: usually pairing one-sided closures through algebraic intertwiners — a setup utilised in the subsequent section.

Defining isotropic functions as specifically algebraic-function constraints of standard representations of the orthogonal group family *maximally* (no supergroup), $\varrho : \mathrm{O}(n) \to \mathrm{GL}_n(\mathbb{R})$. These orthogonal matrices can be denoted in shorthand as $\mathbf{R} \in \mathrm{O}(n)$ for n-width layers, and the algebraic constraint of *Eqn.* 1 can then be simplified to *Eqn.* 4, for a constrained subset of the functional class $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$.

$$\forall \mathbf{R} \in \mathrm{O}(n), \forall \vec{x} \in \mathbb{R}^n : \qquad [\mathbf{R}, \mathbf{f}] = \mathbf{R} \circ \mathbf{f} - \mathbf{f} \circ \mathbf{R} = 0 \quad \text{equally} \quad \mathbf{f}(\mathbf{R}\vec{x}) = \mathbf{R}\mathbf{f}(\vec{x}) \tag{4}$$

This is the desired standard orthogonal equivariant constraint for activation functions to *maximally* adhere to, yielding rotationally symmetric (isotropic) properties. These can be maximally satisfied by the general multivariate functional form shown in *Eqn.* 5, for general non-linear univariate functions $\sigma, \tilde{\sigma}, \bar{\sigma}$, and unit-vector $\hat{x}$. Any supergroup is also feasible for the subsequent diagonalisation, as it preserves this equivariance; however, consider the presence and nature of the respective non-linearity. Several placeholders may be suggested for σ, e.g. $\tanh$; however, superficially replicating existing activation functions should not be assumed to convey any continued optimality in this isotropic form.

$$\mathbf{f}(\vec{x}) = \sigma(\|\vec{x}\|_2)\,\hat{x} \qquad \text{or} \qquad \mathbf{f}(\vec{x}) = \tilde{\sigma}(\|\vec{x}\|_2)\,\vec{x} \qquad \text{or} \qquad \mathbf{f}(\vec{x}) = \bar{\sigma}\left(\vec{x}^T\vec{x}\right)\vec{x} \tag{5}$$

Recovering Neurons: Adopting this ontology entails treating neuron individuation not as an intrinsic property of the vector space or network but as an induced artefact of the *chosen* group representations used to construct primitives, under *Eqns.* 1 through 3. Matrix representations of the permutation group are generally not fixed under standard orthogonal, $\mathbf{R} \in \mathrm{O}(n) \setminus P$, conjugation: $\{\varrho(\pi) \,|\pi \in S_n\} = P \neq \left\{\mathbf{R}\mathbf{P}_\pi\mathbf{R}^T | \mathbf{P}_\pi \in P\right\}$. Thus, constraining a functional form to be *maximally* equivariant to one such matrix representation introduces a *representation-specific* basis dependence in the admitted form. For a conjugated standard permutation representation, $\mathbf{R}\mathbf{P}_\pi\mathbf{R}^T$, this anisotropy distinguishes various vectors, particularly the unordered columns of $\mathbf{R}$, selecting an *absolute frame* in the function's codomain. These distinguished columns of $\mathbf{R}$ correspond to, and make special, up to relabelling, the classical historical notion of artificial neurons[2]. Such individuation ceases for orthogonal-prescribed isotropic functions: $\{\varrho'(Q) \,|Q \in \mathrm{O}(n)\} = O = \left\{\mathbf{R}\mathbf{Q}\mathbf{R}^T | \mathbf{Q} \in O\right\}$, where no direction is inherently distinguished by the group representation's admitted functional forms. Thus, group representations precede the individuated, characteristic [44, 45, 30] nature of emergent neurons, and are sufficient to recover them. Generalising to deindividuated neurons is crucial for this paper.

2.2 Layerwise Affine Diagonalisation

Full singular value decomposition of weights $\mathbf{W}$ produces $\mathbf{W} = \mathbf{R}\mathbf{\Lambda}\mathbf{Q}^T$, with diagonal $\mathbf{\Lambda} = \mathbf{\Lambda} \odot \mathbf{I}$, and orthogonal matrices $\mathbf{R}, \mathbf{Q} \in \mathrm{O}(n)$, which can be used to transform an affine map in *Eqn.* 6.

$$\mathbf{W}\vec{x} + \vec{b} \quad = \quad \mathbf{R}\mathbf{\Lambda}\mathbf{Q}^T\vec{x} + \vec{b} \tag{6}$$

Then, generalising this to performing the decomposition on the middle affine map of a three-layer perceptron and using the designed orthogonal equivariance of isotropic functions enables an overall function-invariant reparameterisation. This is unusual in that it leverages this *continuous* symmetry definition across *three* layers, acting twice concurrently on both sides of the layer. With the choice of SVD, this forms a diagonalised representation of the middle layer (or *App.* F, an ICA representation).

For a multilayer perceptron (MLP), this minimally necessitates a standard orthogonal algebraic (*Eqn.* 1) constraint on the activation function, f_{Iso}, composed alongside two closure (*Eqn.* 3) constraints on each sandwiching affine layer, f_{Aff}, such that a standard-representation acts inwardly and a trivial-representation outwardly, forming the appropriately required closure-invariances. The latter constraint is already satisfied by the existing $\mathrm{O}(n) \subset \mathrm{GL}_n(\mathbb{R})$ two-sided closure of affine maps, enabling the desired equivalence class without modification. One may also *choose*, but not require, an orthogonally symmetric initialiser via *Eqn.* 2. This construction permits two, potentially differing, isotropic activation functions. The three-layer transformations are described in *Eqns.* 7 through 11. Layer widths are not assumed and may vary layer-wise, given this procedure's generality.

$$\mathbf{f} = f_{\text{Aff.3}} \circ f_{\text{Iso.2}} \circ f_{\text{Aff.2}} \circ f_{\text{Iso.1}} \circ f_{\text{Aff.1}} \tag{7}$$

$$\mathbf{f}(\vec{x}) = \mathbf{W}_3 f_{\text{Iso.2}}\left(\mathbf{W}_2 f_{\text{Iso.1}}\left(\mathbf{W}_1\vec{x} + \vec{b}_1\right) + \vec{b}_2\right) + \vec{b}_3 \tag{8}$$

[2]Curiously columns of $\mathbf{C} = \alpha\mathbf{I} + \beta\vec{1}\vec{1}^T$ when similarly unit-scaled provide representation-induced, function-distinguished arcs, with two important cases: non-interfering [33] classical neurons [30, 32] and an equiangularly separated simplex [42, 43].

Then one can apply the SVD to the weight $\mathbf{W}_2$ whilst inserting an identity matrix, $\mathrm{I}_n = \mathbf{R}\mathbf{R}^T$.

$$\mathbf{f}(\vec{x}) = \mathbf{W}_3 f_{\text{Iso.2}}\Big((\underbrace{\mathbf{R}\mathbf{R}^T}_{\mathrm{I}_n})\Big(\mathbf{R}\mathbf{\Lambda}\mathbf{Q}^T f_{\text{Iso.1}}\left(\mathbf{W}_1\vec{x} + \vec{b}_1\right) + \vec{b}_2\Big)\Big) + \vec{b}_3 \tag{9}$$

$$\mathbf{f}(\vec{x}) = \underbrace{\mathbf{W}_3\mathbf{R}}_{\mathbf{W}'_3} f_{\text{Iso.2}}\Big(\mathbf{\Lambda} f_{\text{Iso.1}}\Big(\underbrace{\mathbf{Q}^T\mathbf{W}_1}_{\mathbf{W}'_1}\vec{x} + \underbrace{\mathbf{Q}^T\vec{b}_1}_{\vec{b}'_1}\Big) + \underbrace{\mathbf{R}^T\vec{b}_2}_{\vec{b}'_2}\Big) + \vec{b}_3 \tag{10}$$

Finally, reparameterising the affine layers with modified weights $\mathbf{W}'_i$, $\vec{W}$ and biases $\vec{b}'_i$.

$$\mathbf{f}(\vec{x}) = \mathbf{W}'_3 f_{\text{Iso.2}}\left(\mathbf{\Lambda} f_{\text{Iso.1}}\left(\mathbf{W}'_1\vec{x} + \vec{b}'_1\right) + \vec{b}'_2\right) + \vec{b}_3 \tag{11}$$

$$\mathbf{f}(\vec{x}) = \mathbf{W}'_3 f_{\text{Iso.2}}\left(\vec{W} \odot f_{\text{Iso.1}}\left(\mathbf{W}'_1\vec{x} + \vec{b}'_1\right) + \vec{b}'_2\right) + \vec{b}_3 \tag{12}$$

Hence, the central layer is diagonalised, whilst the MLP remains function-invariant overall to such a transformation. This layer is simplified to one-to-one connectivity, as depicted in *Fig.* 1. The singular values can be ordered, enabling a perturbative-like understanding of the layer's now simplified action — especially when incorporating normalisation or a magnitude-bounded $\mathbf{f}$. Similarly this can be expressed with a Hadamard product, $\odot$, with parameterisation $\vec{W} = \mathrm{diag}(\mathbf{\Lambda})$, as shown in *Eqn.* 12.

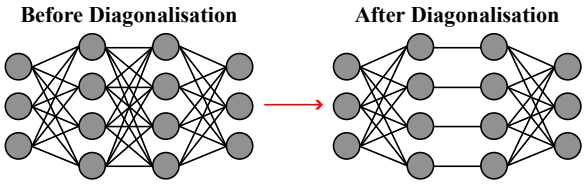


Figure 1: Illustrates the qualitative effects on a network undergoing full diagonalisation. The connectivity is simplified to a one-to-one correspondence between neurons on the chosen layer.

Practically, complete diagonalisation is not necessary for adaptive topologies or interpretability; instead, a left-sided partial diagonalisation, as shown in *Eqn.* 13, is sufficient and marginally more computationally efficient to enact. However, at inference time, full diagonalisation may provide substantial benefit by its significant sparsity whilst remaining analytically function-invariant.

$$\mathbf{f}(\vec{x}) = \mathbf{W}'_3 f_{\text{Iso.2}}\left(\mathbf{\Lambda}\mathbf{Q}^T f_{\text{Iso.1}}\left(\mathbf{W}_1\vec{x} + \vec{b}_1\right) + \vec{b}'_2\right) + \vec{b}_3 \tag{13}$$

Finally, unlike traditional permutation reparameterisations, orthogonal transforms exhibit typical forward-pass function invariance but, novelly, display *variance* in their coupling to most second-order gradient-descent algorithms after the transform. This produces, sequentially backwards from the transform, a training-time-only breaking of the reparameterisation symmetry — absent at inference time. This optimiser connection may offer an insightful direction for future study, explored in *App.* A.

2.2.1 Sparsification

A significant quality arising in isotropic-MLPs stems from this function-invariant diagonalisation. Beyond the adaptive or interpretability considerations, it affords considerable connectivity sparsity.

Two directions may be taken, which may be performed just prior to inference time: Joint sparsity optimisation (JSO), which optimises over the multi-layer orthogonal direct-sum Lie group to minimise an objective function of parameter count via a continuous proxy — many such forms are applicable, including L_1-norm-based approaches. Simpler still is an alternating layerwise diagonalisation, which asymptotically achieves an overall 50% parameter reduction, despite the model function being identical up to floating-point precision. For an odd-depth MLP, *Eqn.* 14 shows the sparsity factor, while *Eqn.* 15 is for the even-depth case, given $2D+1$ and $2D$ affine layers respectively (for fixed N-width, D-depth, and $\dim\mathbf{W} + \dim\vec{b} = N^2 + N$ parameter counting per layer).

$$S_{\text{odd}} = \frac{2DN + (D+1)\left(N^2+N\right)}{(2D+1)\left(N^2+N\right)} = \frac{1+D^{-1}}{2+D^{-1}} + \frac{2D}{(2D+1)(N+1)} \tag{14}$$

$$S_{\text{even}} = \frac{2(D-1)N + (D+1)\left(N^2+N\right)}{2D\left(N^2+N\right)} = \frac{1+D^{-1}}{2} + \frac{D-1}{D(N+1)} \tag{15}$$

Both sparsify to 50% asymptotically as $N, D \to \infty$, considerably reducing memory footprint and computation. This is not a sparse approximation, but an analytical equivalence under this orthogonal basis transform, offering significant overhead improvements. The formulas serve as upper bounds that may be improved with JSO and/or acceptable function approximation. This is considered in conjunction with QR-compressions [38] in *App.* J, clarifying how sparsification achieves considerable sparsity regardless of the MLP's layer-to-layer width structure, unlike QR-compression.

2.3 Left-Sided Neuroadaptation

Neuroadaptation requires less: only the left-sided partial diagonal reparameterisation (*Eqn.* 13). This section covers both function-invariant 'neurogenesis' involving the maintenance of 'scaffold' neurons, and 'neurodegeneration': neuron removal that closely approximates function invariance, especially through the introduction of a new parameter: the 'intrinsic length', capable of absorbing residual bias.

Beginning with the addition of scaffold neurons, this neurogenesis process is achieved by embedding the existing $\mathbb{R}^n$ activation space into $\mathbb{R}^{n+1}$. Under the conventional individuated picture, this corresponds to appending additional neurons to the layer, connected so that the model is functionally independent of them. Here, we term these 'scaffold neurons', indicating neurons present structurally that do not yet, until specialisation, contribute any function. Alternatively, this is comparable to further increasing the dimensionality of the generalised notion of a neuron, since no basis distinguishes them.

Such embedding appears largely arbitrary, yet the initialisation may, in practice, involve nuances ranging from explicit to spontaneous symmetry breaking in the new neuron's associated parameters. This raises implications for learning and is related to the aforementioned optimiser-based symmetry violation under orthogonal transforms. In any case, the new singular values should be zeroed, despite a freedom in the corresponding new row of $\mathbf{W}_3'$ denoted in *Eqn.* 13 for forward growth.

Neurodegeneration is the reverse process: producing a well-chosen subspace with appropriate reparameterisations to closely approximate function-invariance. Given the diagonally ordered, positive singular-value parameters, as one tends to zero, the 'diagonal-neuron' becomes increasingly independent of the preceding layer. Pruning is achieved by removing corresponding entries and reparameterising surroundings. Moreover, this outgoing connectivity pruning now has exact correspondence to neuron removal, due to the now one-to-one diagonal dependence. The thresholding of the zero-tending singular value can be selected to produce a suitably minimal degradation to model functionality. *Isotropic* normalisers (*App.* B) may help prevent activation-magnitude compensations to shrinking singular values. However, a singular-value-independent bias parameter persists.

A novel parameter, the scalar "intrinsic length" denoted o, is a crucial insight restoring a full neurodegeneration symmetry in the $\mathbf{\Lambda}_{ii} \to 0$ limit. It transforms under pruning to gauge away the troublesome 'residual bias', $\vec{b}_i$. The diagonalised layer, with this novel parameter, can be expressed as *Eqn.* 16, while *Eqn.* 17 displays the restored symmetry in the $\mathbf{\Lambda}_{00} \to 0$ limit, provided the bias in the linear term is absorbed into the next layer's biases appropriately. Intrinsic length's transform also preserves any existing positivity; if desired, it may be parameterised as such, e.g. $o = \exp \lambda$.

$$\mathbf{f}\left(\vec{x}\right) = \bar{\sigma}\left(\left\|\mathbf{\Lambda}\vec{x} + \vec{b}\right\|_2^2 + o\right)\left(\mathbf{\Lambda}\vec{x} + \vec{b}\right) \tag{16}$$

$$\left\|\mathbf{\Lambda}\vec{x} + \vec{b}\right\|_2^2 + o = \sum_{j\neq 0}\left(\mathbf{\Lambda}_{jj}x_j + b_j\right)^2 + \underbrace{\left(\mathbf{\Lambda}_{00}x_0}_{\to 0} + b_0\right)^2 + o = \left\|\mathbf{\Lambda}'\vec{x} + \vec{b}'\right\|_2^2 + \underbrace{b_0^2 + o}_{=o'} \tag{17}$$

This neurodegenerates the architecture, entering the functional class of narrower-width multilayer perceptrons, with the reparameterisation symmetry restored in the limit, and arbitrarily well approximated by chosen thresholding. Furthermore, to counteract the persistent growth of the intrinsic length, it can be made trainable. This may offer broader benefits, such as optimising the non-linearity's action or, if negative, allowing a non-linear sign change to representations, thereby permitting antipodal arrangements that may be desirable [33]. Numerous interpretations arise for it, such as a parameterised isotropic activation function preserving orthogonal equivariance, or a new affine translation, embedding the subspace of activations away from the origin, by a rotationally-degenerate bias-like vector, $\vec{o}$, in the orthogonal complement space, constrained by $\vec{o}^T\vec{o} = o$ and $\vec{o} \perp \mathbf{\Lambda}\vec{x} + \vec{b}$.

Additionally, the remaining residual weights ($\mathbf{\Lambda}_{ii} = \epsilon$) in $\mathbf{f}$'s linear term can be better mitigated via the Moore-Penrose pseudo-inverse. Given diagonalised weights $\mathbf{\Lambda}$ and the subsequent pruned form

$\mathbf{\Lambda}'$ alongside the next layer's weight matrices before $\mathbf{W}$ and after pruning $\mathbf{W}'$, then the following map should be closely preserved: $\mathbf{W}'\mathbf{\Lambda}'\mathbf{Q}^T\vec{x} \approx \mathbf{W}\mathbf{\Lambda}\mathbf{Q}^T\vec{x}$, by minimising the least-squares difference in these maps. $\mathbf{W}'$ can then be determined by the pseudo-inverse, where rank permits, shown in *Eqn.* 18. Otherwise, continue using classically pruned $\mathbf{W}'$. If the threshold is particularly small, the inverse may be poorly conditioned; thus, this method necessitates care. Regardless, the standard smallest singular value deletion reduces this to simple corresponding column deletion of $\mathbf{W}'$.

$$\mathbf{W}' = \mathbf{W}\mathbf{\Lambda}\mathbf{\Lambda}'^T \left(\mathbf{\Lambda}'\mathbf{\Lambda}'^T\right)^{-1} \tag{18}$$

2.3.1 Neurogenesis and Jacobians of Isotropic Activation Functions

Concerns regarding the trainability of scaffold neurons are mitigated by analysing the Jacobian, $J_{\mathbf{f}}$, of isotropic activation functions and *App.* A.3's discussion. Alongside standard connectivity mixing, isotropic Jacobians distribute learning gradients globally even in the case of an effective activation subspace. Hence, scaffold neurons may train rapidly, even if independent of the model's function and zero-valued associated parameters. The term proportional to the identity, I, in *Eqn.* 19's Jacobian is generally non-zero, increasing gradient flow through scaffold neurons, hastening differentiation and operationalisation. However, canonical (Kronecker) basis vector directions $\hat{e}_i$ of scaffold neurons are zeroed in *Eqn.* 20's Jacobian. With a suitable choice of $\tilde{\sigma}$, merely a coordinate singularity exists at $\vec{0}$.

$$\text{Isotropic:} \qquad \mathbf{f}\left(\vec{x}\right) = \tilde{\sigma}\left(\left\|\vec{x}\right\|_2\right)\vec{x} \qquad \Rightarrow \qquad J_{\mathbf{f}} = \tilde{\sigma}\left(\left\|\vec{x}\right\|_2\right)\mathrm{I} + \frac{\tilde{\sigma}'\left(\left\|\vec{x}\right\|_2\right)}{\left\|\vec{x}\right\|_2}\vec{x}\vec{x}^T \tag{19}$$

$$\text{Elementwise:} \qquad \mathbf{f}\left(\vec{x}\right) = \sum_{i=1}^{n}\sigma\left(\vec{x}\cdot\hat{e}_i\right)\hat{e}_i \qquad \Rightarrow \qquad J_{\mathbf{f}} = \sum_{i=1}^{n}\sigma'\left(\vec{x}\cdot\hat{e}_i\right)\hat{e}_i\hat{e}_i^T \tag{20}$$

2.3.2 Practical Considerations of Scheduling and Thresholding

Singular value decomposition itself features an $\mathcal{O}\left(m^2 n\right)$ computational scaling dependence for $\mathbb{R}^{n\times m}$, $m < n$, matrices. Due to this cost, it is appropriate to consider only intermittent scheduling for neuroadaptation. This scheduling aligns with existing literature [20], but here it stems from the computational cost of the SVD rather than dense gradient computation. For straightforward implementation, this work considers layerwise decomposition only once per training epoch.

Although many methods may be feasible and compatible with the neuroadaptation described, as stated, we focus on the following distinct approach: maintaining a buffer of 'scaffold neurons' in the diagonalised representation. The size of this buffer corresponds to a new hyperparameter, $\Xi \in \mathbb{Z}_+$, representing the desired number of singular values specified by a thresholding method. Because isotropic activation functions are not elementwise, different values of Ξ are inequivalent, leading to tangible differences in training and inference. A learning-induced increase in the number of neurons in the thresholded 'scaffold buffer' then triggers neurodegeneration, removing the smallest singular-value neuron with the least functional impact; an insufficient number in the buffer triggers neurogenesis—offering principled, real-time network adaptability. We note that many thresholding methods could indicate when neuroadaptation should occur; we compare a non-exhaustive list of methods in detail in *App.* C. However, we propose and test a minimalistic approach using $0 < \vartheta < 1$ to demarcate when singular values have a negligible, or rarely contributing, effect. Singular values below 1 cause multiplicative decay in activations, suggesting their insignificance in a model's function. Minimalistic thresholding is attractive in the subsequent verification experiments, since its simplicity mitigates additional failure modes while performing well with contextually appropriate adaptations.

2.3.3 Algorithmic Implementation

This section lists the exact transforms for forward neuroadaptation of an isotropic layer, for varying m: $(\mathbb{R}^n \to)\mathbb{R}^m \to \mathbb{R}^p$. For efficiency, several steps during and after adaptation (reparameterisation-*contractions* $\mathbf{W}_1' = \mathbf{\Lambda}'\mathbf{Q}^T$) are done, meaningfully altering optimisation trajectories (*App.* A).

The chosen scheduler determines the timing of these actions, and they are repeated per layer specified. First, perform left-sided partial diagonalisation (*Eqn.* 21) — such diagonalisation need not be later undone, but $\mathbf{W}_1' = \mathbf{\Lambda}'\mathbf{Q}^T$ contraction is done. From the diagonal singular value matrix, $\mathbf{\Lambda}$, determine the number of scaffold neurons from singular values below the specified threshold $\mathbf{\Lambda}_{ii} < \vartheta$.

$$\mathbf{f}\left(\vec{x}\right) = \mathbf{W}_2 f_{\text{Iso.}}\left(\mathbf{\Lambda}\mathbf{Q}^T\vec{x} + \vec{b}_1; o\right) + \vec{b}_2 \tag{21}$$

If the number of scaffold neurons falls below the desired count $\omega = |\{\mathbf{\Lambda}_{ii} | \mathbf{\Lambda}_{ii} < \vartheta\}| < \Xi$, iteratively enact neurogenesis for $(\Xi - \omega)$ neurons, described as follows:

1. Append a row of zeros to the bottom of $\mathbf{\Lambda} \in \mathbb{R}_{\geq 0}^{m \times n}$, to form a matrix $\mathbf{\Lambda}' \in \mathbb{R}_{\geq 0}^{(m+1) \times n}$.
2. For $\vec{b}_1 \in \mathbb{R}^m$ this is expanded to include a new last entry, b_+ to form $\vec{b}_1' \in \mathbb{R}^{(m+1)}$. For o-positivity, b_+ must be chosen such that $b_+^2 + o' = o$, while $b_+ = 0$ is needed for $\vec{b}_2$'s step.
3. Hence, with $o \in \mathbb{R}_{\geq 0}$, an adjustment follows accordingly from the prior step: $o' = o - b_+^2$.
4. The weight matrix, $\mathbf{W}_2 \in \mathbb{R}^{p \times m}$, acquires a new column, $\vec{W}_+$, forming $\mathbf{W}_2 \in \mathbb{R}^{p \times (m+1)}$. The column $\vec{W}_+$ exhibits freedom in its initialisation only during the forward pass, due to zero premultiplication; however, it does couple with the optimisation discussed below.
5. For vector, $\vec{b}_2 \in \mathbb{R}^p$, no adjustment needs to be made, if $b_+ = 0$, due to identical impact.

Initialising $\vec{W}_+$ as the (near-)zero vector can result in spontaneous symmetry breaking in the next optimisation step, whilst a non-zero vector acts as an explicit symmetry breaking, biasing learning accordingly, discussed in *App.* A.3. Hence, a range of options is available for empirical exploration: a linear combination of existing columns to achieve greater fidelity around the existing span, or various choices for semi-orthogonality, etc. Choices about symmetry-breaking initialisations may result in scaffold neurons diverging in specialisation, with their number having significant consequences.

Iterative pruning occurs when the number of scaffold neurons exceeds the desired count, $\omega > \Xi$.

1. The row of $\mathbf{\Lambda} \in \mathbb{R}_{\geq 0}^{m \times n}$ with smallest norm is deleted, forming $\mathbf{\Lambda}' \in \mathbb{R}_{\geq 0}^{(m-1) \times n}$.
2. Similarly, delete the corresponding entry, $b_\times$, of $\vec{b}_1 \in \mathbb{R}^m$ to form $\vec{b}_1' \in \mathbb{R}^{(m-1)}$.
3. Adjust $o \in \mathbb{R}_{\geq 0}$ accordingly by $o' = o + b_\times^2$.
4. Deletion of $\mathbf{W}_2 \in \mathbb{R}^{p \times m}$ corresponding column, $\vec{W}_\times$, then yields $\mathbf{W}_2 \in \mathbb{R}^{p \times (m-1)}$.
5. For vector, $\vec{b}_2 \in \mathbb{R}^p$, an adjustment arising from the prior bias term can be considered: $\vec{b}_2' = \vec{b}_2 + \mathbb{E}\left[\bar{\sigma}\left(\cdot\right)\right] \vec{W}_\times b_\times$. This better approximates the functionality, where the expectation may be computed batch-wise or test-set-wise, or if isotropic normalisation ensures it is one.

Better still, a further non-linear additive term and learnable parameter, $+\sigma\left(\cdot\right) \vec{\psi}$, can be used to absorb such terms identically, $\vec{\psi}' = \vec{\psi} + \vec{W}_\times b_\times$ — essential is its regularised decay $\vec{\psi} \to \vec{0}$. This is for two reasons: it must be negligible before next-layer pruning, so it doesn't require further correction, and this also suggests an output-to-input ordering for neuroadaptation. Secondly, translation inconsistency due to the non-linear behaviour of ψ may be troublesome, so it should decay over a short yet sufficient time for the network to compensate appropriately. Across both procedures, $\mathbf{Q}^T \in \mathbb{R}^{n \times n}$ need not be changed and can then be contracted with $\mathbf{\Lambda}'$ once concluded. Repeating this procedure over all specified layers then produces the automatic adaptive architecture. One can choose the final-layer representations, but alternating diagonalisation may help via the aforementioned sparsity benefit.

3 Empirical Verifications of Theory and Mechanisms

Naturally, the present derivations apply to multilayer perceptrons; hence, the claims must be assessed in reference to various instances of this architecture. We use $\sigma(a) = \tanh a$ as a working placeholder, in analogy to standard tanh, but it is not assumed to be optimal[3]. Consequently, we empirically verify the theoretical claims about function invariance and neuroadaptation, rather than broader competitiveness due to function-optimality confounds. Beginning with *Tab.* 1 as the key demonstration of function preservation and invariance when neuroadapting a Caltech256 [46] MLP architecture, with invariances quantified through test-set mean relative change measured for the output before and after adaptation.

Table 1 shows all variations for intrinsic length and ψ-correction worked, with the best invariances observed for constant or trainable intrinsic lengths, but varying for $\vec{\psi}$. Tiny relative changes ($\approx 10^{-4}\%$)

[3]'Isotropic-tanh' also saturates, $\lim_{\alpha \to \infty} \sigma\left(\alpha\right) = 1$, which alleviates concerns regarding magnitude growth in activations compensating for small singular values. Thus, zero-tending singular values do suggest a negligible contribution to the function.

Table 1: Experiment One (*App.* K.1): measures function invariance after neuroadaptation: appending $+k$, removing $-k$, or diagonalisation 0 the $\mathbb{R}^{100} \to \mathbb{R}^{100}$ layer of a $[12288, 100, 100, 256]$ Caltech256 trained classifier. Mean $\pm$ standard error (SE) are reported over 20 independent repeats. Columns specify intrinsic length o and the ψ-vector approach, yielding 7×20 total networks. Σ_0 and Σ_1 denote the before- and after-sums of singular values, respectively (with their SE omitted for legibility; all ≈ 0.4). ϵ_{rel} is the test-set mean L_2 logit change divided by the initial value, reported as a percentage.

Neuroadaptation	$o_{\text{none}},\ \vec{\psi}_{\text{decay}}$ $\Sigma_0 = 240$		$o_{\text{constant}},\ \vec{\psi}_{\text{decay}}$ $\Sigma_0 = 240$		$o_{\text{trainable}},\ \vec{\psi}_{\text{decay}}$ $\Sigma_0 = 241$		$o_{\text{trainable}},\ \vec{\psi}_{\text{none}}$ $\Sigma_0 = 240$		$o_{\text{trainable}},\ \vec{\psi}_{\text{constant}}$ $\Sigma_0 = 237$		$o_{\text{trainable}},\ \vec{\psi}_{\text{trainable}}$ $\Sigma_0 = 237$		$o_{\text{trainable}},\ \vec{b}_{\text{adjusted}}$ $\Sigma_0 = 240$	
	Σ_1	ϵ_{rel} (%)	Σ_1	ϵ_{rel} (%)	Σ_1	ϵ_{rel} (%)	Σ_1	ϵ_{rel} (%)	Σ_1	ϵ_{rel} (%)	Σ_1	ϵ_{rel} (%)	Σ_1	ϵ_{rel} (%)
$+75$	240	$(8.4 \pm 0.2) \times 10^{-4}$	240	$(8.5 \pm 0.2) \times 10^{-4}$	241	$\mathbf{(8.2 \pm 0.2) \times 10^{-4}}$	240	$(8.6 \pm 0.1) \times 10^{-4}$	237	$(8.5 \pm 0.2) \times 10^{-4}$	237	$(8.3 \pm 0.2) \times 10^{-4}$	240	$(8.3 \pm 0.1) \times 10^{-4}$
$+50$	240	$(8.4 \pm 0.2) \times 10^{-4}$	240	$(8.5 \pm 0.2) \times 10^{-4}$	241	$\mathbf{(8.2 \pm 0.2) \times 10^{-4}}$	240	$(8.6 \pm 0.1) \times 10^{-4}$	237	$(8.5 \pm 0.2) \times 10^{-4}$	237	$(8.3 \pm 0.2) \times 10^{-4}$	240	$(8.3 \pm 0.1) \times 10^{-4}$
$+25$	240	$(8.4 \pm 0.2) \times 10^{-4}$	240	$(8.5 \pm 0.2) \times 10^{-4}$	241	$\mathbf{(8.2 \pm 0.2) \times 10^{-4}}$	240	$(8.6 \pm 0.1) \times 10^{-4}$	237	$(8.5 \pm 0.2) \times 10^{-4}$	237	$(8.3 \pm 0.2) \times 10^{-4}$	240	$(8.3 \pm 0.1) \times 10^{-4}$
$+5$	240	$(8.4 \pm 0.2) \times 10^{-4}$	240	$(8.5 \pm 0.2) \times 10^{-4}$	241	$\mathbf{(8.2 \pm 0.2) \times 10^{-4}}$	240	$(8.6 \pm 0.1) \times 10^{-4}$	237	$(8.5 \pm 0.2) \times 10^{-4}$	237	$(8.3 \pm 0.2) \times 10^{-4}$	240	$(8.3 \pm 0.1) \times 10^{-4}$
$+1$	240	$(8.4 \pm 0.2) \times 10^{-4}$	240	$(8.5 \pm 0.2) \times 10^{-4}$	241	$\mathbf{(8.2 \pm 0.2) \times 10^{-4}}$	240	$(8.6 \pm 0.1) \times 10^{-4}$	237	$(8.5 \pm 0.2) \times 10^{-4}$	237	$(8.3 \pm 0.2) \times 10^{-4}$	240	$(8.3 \pm 0.1) \times 10^{-4}$
0	240	$(8.4 \pm 0.2) \times 10^{-4}$	240	$(8.5 \pm 0.2) \times 10^{-4}$	241	$\mathbf{(8.2 \pm 0.2) \times 10^{-4}}$	240	$(8.6 \pm 0.1) \times 10^{-4}$	237	$(8.5 \pm 0.2) \times 10^{-4}$	237	$(8.3 \pm 0.2) \times 10^{-4}$	240	$(8.3 \pm 0.1) \times 10^{-4}$
-1	240	$(2.4 \pm 0.1) \times 10^{-1}$	240	$(2.4 \pm 0.1) \times 10^{-1}$	240	$(2.6 \pm 0.2) \times 10^{-1}$	240	$\mathbf{(2.2 \pm 0.1) \times 10^{-1}}$	237	$(2.7 \pm 0.1) \times 10^{-1}$	236	$(2.6 \pm 0.1) \times 10^{-1}$	240	$(2.3 \pm 0.2) \times 10^{-1}$
-5	236	$(3.5 \pm 0.2) \times 10^{-1}$	236	$(3.7 \pm 0.2) \times 10^{-1}$	236	$(3.8 \pm 0.3) \times 10^{-1}$	236	$\mathbf{(3.4 \pm 0.2) \times 10^{-1}}$	233	$(6.7 \pm 0.2) \times 10^{-1}$	233	$(6.4 \pm 0.2) \times 10^{-1}$	236	$(3.7 \pm 0.3) \times 10^{-1}$
-25	212	$(7.2 \pm 0.2) \times 10^{-1}$	212	$(7.3 \pm 0.2) \times 10^{-1}$	212	$(7.5 \pm 0.3) \times 10^{-1}$	212	$(7.1 \pm 0.2) \times 10^{-1}$	209	$(1.0 \pm 0.0) \times 10^{0}$	209	$(9.9 \pm 0.2) \times 10^{-1}$	212	$\mathbf{(7.1 \pm 0.3) \times 10^{-1}}$
-50	170	$(3.7 \pm 0.1) \times 10^{0}$	170	$(3.6 \pm 0.1) \times 10^{0}$	170	$(3.9 \pm 0.2) \times 10^{0}$	170	$\mathbf{(3.6 \pm 0.2) \times 10^{0}}$	167	$(4.2 \pm 0.1) \times 10^{0}$	167	$(4.1 \pm 0.2) \times 10^{0}$	170	$(3.7 \pm 0.1) \times 10^{0}$
-75	99	$(6.3 \pm 0.0) \times 10^{+1}$	99	$(6.3 \pm 0.0) \times 10^{+1}$	99	$(6.2 \pm 0.1) \times 10^{+1}$	99	$(6.3 \pm 0.0) \times 10^{+1}$	98	$(6.3 \pm 0.0) \times 10^{+1}$	97	$\mathbf{(6.2 \pm 0.0) \times 10^{+1}}$	99	$(6.3 \pm 0.0) \times 10^{+1}$

for neurogenesis and diagonalisation indicate negligible function alteration, confirming theoretical predictions and may arise from machine-error amplification in forward propagation. This rises modestly ($\approx 3\%$) when removing the lowest 50% of neurons, and increases to ($\approx 60\%$, width $100 \to 25$) when rank decreases severely and non-negligible singular values are removed. In practice, however, thresholded neuroadaptation should not remove such large singular values. *App.* K.1 supports this conclusion with extended results across KL-divergence, accuracy and relative-changes for all CIFAR-10, CIFAR-100 and Caltech256. Overall, this shows that functional degradation is mitigated despite *no retraining* after adaptation—also explaining columnwise consistency. Training-set optimisation, at 19.0 ± 0.2s, and test-set evaluation, at 2.04 ± 0.01s, made the epoch-wise diagonalisation and neurogenesis, 29 ± 1ms, or neurodegeneration, 39 ± 2ms, comparatively negligible.

Fig. 2 shows the collated evolution of isotropic parameters, which evolve very similarly despite random (orthogonal) initialisation, perhaps because the symmetry induces a flat loss curve across the group's continuous orbit. These differ strongly from elementwise forms (*App.* K.2), showing their incomparability despite superficial similarities. Layers 0 and 2 show strong specialisation, with distinct separating bands—decay of small singular value tails suggests utility in their neurodegeneration.

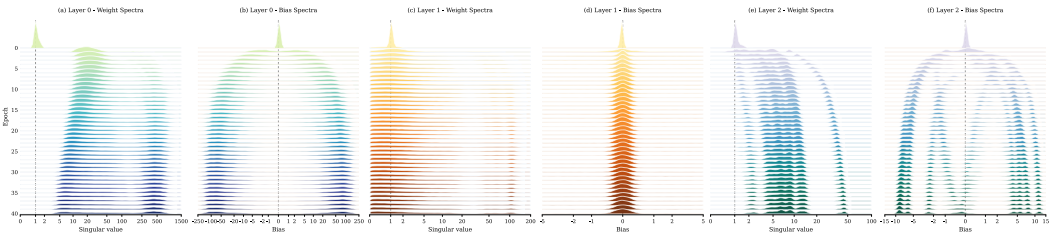


Figure 2: Experiment Two (*App.* K.2) displays the evolving layerwise singular value and bias spectra for a $[3072, 100, 100, 10]$ fixed isotropic MLP trained on CIFAR-10 [47] classification. The spectra are obtained from 40 independent repeats, showing strong consistency. Singular values begin at one, indicated by the dashed line, due to orthogonal initialisation, and biases at zero due to initialisation.

Measuring neuroadaptation under a changing task schedule is essential, with *Fig.* 3 demonstrating how added learning pressures tend to cause a bounded neurogenesis until stabilising at a new width, whilst removing datasets causes neurodegeneration of unneeded excess to a new stable width — reducing EMNIST and MNIST test accuracies correspondingly to demand. This indicates *Fig.* 2's intended decay of low-singular-value bands is robust (if desirable one could preserve task-corresponding singular values [51]). Overall, it demonstrates clearly that thresholded scaffold neurons, and their buffer, operate as intended. Adaptive networks have matched or improved accuracy over static baselines[4]. The first layer, H1, shows remarkable consistency across repeats at all times,

[4]At T3, epoch 450, the (control)/(adaptive) mean and standard error test accuracy, was recorded at $(94.2 \pm 0.1)\%$/$(94.2 \pm 0.1)\%$ for MNIST, $(85.6 \pm 0.2)\%$/$(85.4 \pm 0.1)\%$ for FMNIST and $(87.3 \pm 0.2)\%$/$(87.3 \pm 0.2)\%$ for A-J EMNIST

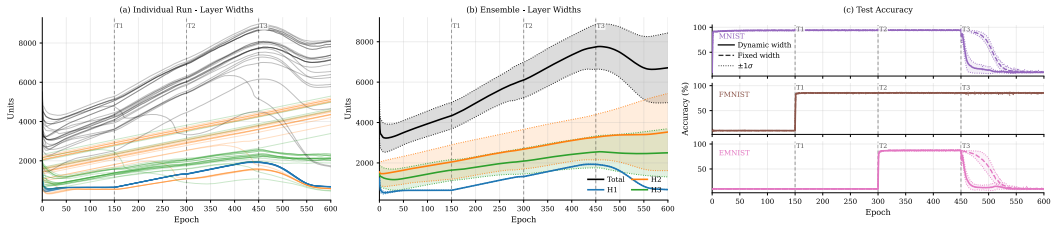


Figure 3: Experiment Three (*App.* K.3) demonstrates thresholded neural adaptation when an architecture of $[2352, \text{H1}, \text{H2}, \text{H3}, 30]$ adapts to a changing task schedule, across 25 independent repeats for random hidden widths $\text{H1}, \text{H2}, \text{H3} \sim \text{U}(500, 2500)$. From epoch 0 through 150, only MNIST [48] cross-entropy terms contribute to the loss; at T1, the total loss is equally averaged between both MNIST and FMNIST [49]. At T2, the total loss is equally averaged between MNIST, FMNIST, and A-J letters from EMNIST [50], and finally, at T3, *only* FMNIST's loss contributes.

suggestive of an optimal width. Curiously, distinct width groupings appear and bifurcate in H2, and clearly for H3, beginning from the start to the T3 boundary. The linearly increasing H2 group appears metastable, often converging rapidly to a stable, narrower width. We speculate this reflects function consolidation, in which inter-task learning transfer merges separately acquired task-specific behaviours into a shared approach that supports multiple tasks—reducing redundancy in both function and neurons. Adding datasets appears to trigger this, perhaps by bridging intermediate internal representations, strengthened by its occurrence in deeper layers, suggestive of redundancy exploited in more generalised representations shareable across tasks. Individual runs indicate bifurcation may involve layer couplings, typically splitting between standard contracting $H1 \geq H2 \geq H3$, which stabilises, and $H2/3 \geq H3/2 \geq H1$, which grow. Work identified subsequently on QR-compression [38] may force width contraction but alters learning. Overall, adaptation remains effective and responsive to task demands: growing and shrinking with task addition and removal.

4 Conclusion

We develop a symmetry-led approach to adaptive topologies. Crucially, through this we demonstrate the utility of prescribed primitive-scale symmetries in their own right, which need not correspond to symmetries of the data and model, but instead are used solely within internal representations to enable otherwise inaccessible structural transformations. Activation functions are made to exhibit orthogonal equivariance that permits function-invariant diagonalisation and neuroadaptation. Verification across various datasets confirms negligible functional degradation under substantial neurogenesis, while mitigating drift under neurodegeneration. Together, this suggests the taxonomy and reversed ontology underlying the method offer novel behaviours and opportunities: from neuroadaptation to sparsification and ICA-interpretable representations, this sweep offers concrete examples of unusual capabilities enabled through symmetry-prescribed primitives. Moreover, the reversal reveals neuron individuation can be an emergent consequence rather than a fundamental assumption. This motivates a broader exploration of prescribed primitives and the expanded functional class of neural systems they permit for general applications. Appendices cover optimiser and normaliser considerations, sparsification verification, nested functional classes, ICA representations, a sketch of convolution and transformer neuroadaptation and the *parallelised precomputation of all matrix-vector products*.

Scope and Limitations. This approach to neuroadaptation has a defined scope, as expanded in *App.* I. The empirics focus on mechanistic verification of the approach. This is for several reasons, including that the methodology depends on isotropic primitives, which have not benefited from the longstanding refinement of modern elementwise functions. This offers a substantial avenue for further exploration. Additionally, isotropic forms still lack a general UAT [52, 53]; yet the intervening decades between Rosenblatt [54] and elementwise-UATs did not stall this form's practical adoption. Separately, the primary methodology centres on MLP architectures for clarity. Accordingly, we prioritise mechanistic verification over competitive SOTA performance. Further implementation considerations include explicit symmetry breaking during neurogenesis while retaining function-invariance.

Acknowledgments and Disclosure of Funding

Work supported by the Engineering and Physical Sciences Research Council (EPSRC), grant number [EP/W524347/1]. I thank Dr Omar Rivasplata and Dr Mohammad Saeed Bahramy for their continued supervision and support throughout my PhD.

References

[1] Caleb J. Axelrod, Swanne P. Gordon, and Bruce A. Carlson. Integrating neuroplasticity and evolution. *Current Biology*, 33(8):R288–R293, Apr 2023. ISSN 0960-9822. doi: 10.1016/j.cub.2023.03.002. URL https://doi.org/10.1016/j.cub.2023.03.002.

[2] Wei Wen and Gina G Turrigiano. Keeping your brain in balance: homeostatic regulation of network function. *Annual review of neuroscience*, 47, 2024.

[3] Pico Caroni, Flavio Donato, and Dominique Muller. Structural plasticity upon learning: regulation and functions. *Nature Reviews Neuroscience*, 13(7):478–490, 2012.

[4] Jyoti Mishra and Adam Gazzaley. Cross-species approaches to cognitive neuroplasticity research. *NeuroImage*, 131:4–12, 2016. doi: 10.1016/j.neuroimage.2015.10.013. URL https://doi.org/10.1016/j.neuroimage.2015.10.013.

[5] Chiara La Rosa and Luca Bonfanti. Brain plasticity in mammals: An example for the role of comparative medicine in the neurosciences. *Frontiers in Veterinary Science*, 5:274, 2018. doi: 10.3389/fvets.2018.00274. URL https://doi.org/10.3389/fvets.2018.00274.

[6] Ioanna Dori, Chrysanthi Bekiari, Ioannis Grivas, Anastasia Tsingotjidou, and Georgios C Papadopoulos. Birth and death of neurons in the developing and mature mammalian brain. *The International Journal of Developmental Biology*, 66(1-2-3):9–22, 2021.

[7] Anthony Holtmaat and Karel Svoboda. Experience-dependent structural synaptic plasticity in the mammalian brain. *Nature Reviews Neuroscience*, 10(9):647–658, 2009.

[8] Martin M Riccomagno and Alex L Kolodkin. Sculpting neural circuits by axon and dendrite pruning. *Annual review of cell and developmental biology*, 31(1):779–805, 2015.

[9] Carolin Scholl, Michael E Rule, and Matthias H Hennig. The information theory of developmental pruning: Optimizing global network architectures using local synaptic rules. *PLoS computational biology*, 17(10):e1009458, 2021.

[10] Gina G Turrigiano and Sacha B Nelson. Hebb and homeostasis in neuronal plasticity. *Current opinion in neurobiology*, 10(3):358–364, 2000.

[11] Dmitri B Chklovskii, BW Mel, and K Svoboda. Cortical rewiring and information storage. *Nature*, 431(7010):782–788, 2004.

[12] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application, 2024. URL https://arxiv.org/abs/2302.00487.

[13] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[14] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

[15] Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, 5, 1992.

[16] Babak Hassibi, David G Stork, and Gregory J Wolff. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pages 293–299. IEEE, 1993.

[17] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks, 2015. URL https://arxiv.org/abs/1506.02626.

[18] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns, 2016. URL https://arxiv.org/abs/1608.04493.

[19] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Snip: Single-shot network pruning based on connection sensitivity, 2019. URL https://arxiv.org/abs/1810.02340.

[20] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners, 2021. URL https://arxiv.org/abs/1911.11134.

[21] Michael C Mozer and Paul Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. *Advances in neural information processing systems*, 1, 1988.

[22] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer, 2016. URL https://arxiv.org/abs/1511.05641.

[23] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks, 2016. URL https://arxiv.org/abs/1606.04671.

[24] Tao Wei, Changhu Wang, Yong Rui, and Chang Wen Chen. Network morphism. In *International conference on machine learning*, pages 564–572. PMLR, 2016.

[25] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks, 2018. URL https://arxiv.org/abs/1708.01547.

[26] Woojeong Kim, Suhyun Kim, Mincheol Park, and Geunseok Jeon. Neuron merging: Compensating for pruned neurons. *Advances in Neural Information Processing Systems*, 33:585–595, 2020.

[27] Héctor J. Sussmann. Uniqueness of the weights for minimal feedforward nets with a given input-output map. *Neural Networks*, 5(4):589–593, 1992. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(05)80037-1. URL https://www.sciencedirect.com/science/article/pii/S0893608005800371.

[28] Vijay Badrinarayanan, Bamdev Mishra, and Roberto Cipolla. Understanding symmetries in deep networks, 2015. URL https://arxiv.org/abs/1511.01029.

[29] Charles Godfrey, Davis Brown, Tegan Emerson, and Henry Kvinge. On the symmetries of deep learning models and their internal representations, 2023. URL https://arxiv.org/abs/2205.14258.

[30] George Bird. The spotlight resonance method: Resolving the alignment of embedded activations. In *Second Workshop on Representational Alignment at ICLR 2025*, 2025. URL https://openreview.net/forum?id=alxPpqVRzX.

[31] George Bird. Isotropic deep learning: You should consider your (foundational) biases, May 2025. URL https://doi.org/10.5281/zenodo.15476947.

[32] George Bird. Emergence of quantised representations isolated to anisotropic functions, 2025. URL https://arxiv.org/abs/2507.12070.

[33] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

[34] J. Hérault and B. Ans. Réseau de neurones à synapses modifiables: Décodage de messages sensoriels composites par apprentissage non supervisé et permanent. *Comptes Rendus de l'Académie des Sciences, Série III*, 299:525–528, 1984.

[35] Bernard Ans, Jeanny Hérault, Christian Jutten, et al. Architectures neuromimétiques adaptatives: Détection de primitives. *Proceedings of Cognitiva*, 85(2):593–597, 1985.

[36] Pierre Comon. Independent component analysis, a new concept? *Signal processing*, 36(3): 287–314, 1994.

[37] Utku Evci, Yani A. Ioannou, Cem Keskin, and Yann Dauphin. Gradient flow in sparse neural networks and how lottery tickets win, 2022. URL https://arxiv.org/abs/2010.03533.

[38] Iordan Ganev, Twan van Laarhoven, and Robin Walters. Universal approximation and model compression for radial neural networks, 2022. URL https://openreview.net/forum?id=ePgJfxYxl7m.

[39] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[40] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[41] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges, 2021. URL https://arxiv.org/abs/2104.13478.

[42] Vardan Papyan, X. Y. Han, and David L. Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117 (40):24652–24663, September 2020. ISSN 1091-6490. doi: 10.1073/pnas.2015509117. URL http://dx.doi.org/10.1073/pnas.2015509117.

[43] Vignesh Kothapalli. Neural collapse: A review on modelling principles and generalization, 2023. URL https://arxiv.org/abs/2206.04041.

[44] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations, 2017. URL https://arxiv.org/abs/1704.05796.

[45] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. doi: 10.23915/distill.00007. https://distill.pub/2017/feature-visualization.

[46] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. Technical report, California Institute of Technology, 2007.

[47] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[48] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[49] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[50] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

[51] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, March 2017. ISSN 1091-6490. doi: 10.1073/pnas.1611835114. URL http://dx.doi.org/10.1073/pnas.1611835114.

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

[53] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[54] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[55] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. pmlr, 2013.

[56] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

[57] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

[58] Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. *URL https://kellerjordan. github. io/posts/muon*, 6(3):4, 2024.

[59] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

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

[61] Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

[62] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[63] George Bird. Parallelising matrix-products in isotropic multilayer perceptrons. In *OPT 2026: Optimization for Machine Learning (NeurIPS 2026 Workshop)*, 2026. URL https://openreview.net/forum?id=z3N0ccKOi0.

[64] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding, 2017. URL https://arxiv.org/abs/1703.03130.

[65] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

[66] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules, 2017. URL https://arxiv.org/abs/1710.09829.

[67] D.S. Broomhead and D Lowe. Radial basis functions, multi-variable functional interpolation, and adaptive networks. *Royal Signals & Radar Establishment*, 1988.

[68] Derek Lim, Theo Moe Putterman, Robin Walters, Haggai Maron, and Stefanie Jegelka. The empirical impact of neural parameter symmetries, or lack thereof, 2024. URL https://arxiv.org/abs/2405.20231.

[69] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks, 2019. URL https://arxiv.org/abs/1803.03635.

[70] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[71] Roger Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285, 1990.

[72] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[73] Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *International conference on machine learning*, pages 1120–1128. PMLR, 2016.

[74] Daniel E. Worrall, Stephan J. Garbin, Daniyar Turmukhambetov, and Gabriel J. Brostow. Harmonic networks: Deep translation and rotation equivariance, 2017. URL https://arxiv.org/abs/1612.04642.

[75] George Bird. Isotropic activation functions: Early works (orthogonal equivariant activation functions), 2026. URL https://doi.org/10.5281/zenodo.20208386. Earlier work dating to 7 October 2020.

[76] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[77] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. URL https://arxiv.org/abs/1912.01703.

[78] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

Appendix Contents

The appendices present a breadth of additional considerations, network constructions, notations, experiments and real-time interpretability approaches. The sections are structured as follows.

A. Considering Gradient Non-Invariance Examines how function-preserving forward-pass reparameterisations can alter optimisation trajectories.

- **A.1 Transforming Optimiser States** Analyses how optimiser statistics transform under orthogonal reparameterisation.
- **A.2 Tensor Contraction Inequivalence** Explains why equivalent parameterisations can produce different gradient updates.
- **A.3 Null-Like Reparameterisations** Discusses gradient flow and symmetry breaking during scaffold-neuron initialisation.

B. Considerations of Normalisation Examines normalisation compatibility with isotropic activations and proposes symmetry-preserving alternatives.

C. Alternative Methodologies to Thresholding Compares alternative criteria for identifying negligible singular values (scaffold neurons) and triggering neuroadaptation.

D. Parallelising Matrix-Vector Products and Nested Functional Classes Derives parallelisable computations and asymptotically nested functional classes for isotropic networks.

E. Architectural Generalisations Explores extensions of the neuroadaptive framework beyond multilayer perceptrons.

- **E.1 Sketching Adaptive-Channel Convolutional Networks** Outlines function preserving channel adaptation in convolutional architectures.
- **E.2 Attention Mechanisms** Explores potential applications to attention mechanisms.

F. Independent Component Analysis (ICA) Network Representation Develops function preserving ICA representations for real-time inference interpretability and monitoring.

G. Notation Defines the mathematical notation and conventions used throughout the paper.

H. Plain-Language Glossary Explains key mathematical and conceptual terminology.

I. Limitations, Assumptions, and Applicability Clarifies the framework's assumptions, limitations, and scope of applicability.

J. Related Works Compares the proposed approach with existing architecture adaptation and pruning methods.

K. Experimental Details Provides experimental configurations, extended results, and supporting analyses.

- **K.1 Experiment One: Function-Invariance to Neuroadaptation** Quantifies functional changes under neurogenesis, neurodegeneration, and diagonalisation.
- **K.2 Experiment Two: Singular Value and Bias Spectra Evolutions** Characterises the evolution of singular-value and bias spectra during training.
  - **K.2.1 Comparison to Identical Architecture with Standard Tanh** Compares spectral evolution between isotropic and elementwise tanh.
  - **K.2.2 Comparison Over Hidden-Layer Widths** Examines how hidden-layer width affects spectral evolution.
  - **K.2.3 Comparison to Caltech256 Across Various Hidden-Layer Widths** Extends spectral comparisons across widths on Caltech256.
- **K.3 Experiment Three: Thresholded Neuroadaptation to Changing Task Schedule** Evaluates automatic architectural growth and contraction under changing task demands.
- **K.4 Experiment Four: Initial Under-to-Over Neuron Abundance** Investigates the effects of initial neuron abundance on adaptive training.
- **K.5 Experiment Five: Sparsification** Verifies function-preserving parameter sparsification through layerwise diagonalisation.
- **K.6 Experiment Six: Isotropic-Tanh Against Standard-Tanh Trends** Compares classification performance trends between isotropic and conventional tanh.

A Considering Gradient Non-Invariance

Despite the forward-pass invariance of various reparameterisation approaches, some reparameterisations diverge in their optimisation trajectories under these transforms. This section will overview these considerations and demonstrate how neuroadaptive transforms, optimiser states and contractions introduce greater nuance than straightforward forward-pass parameter symmetries.

A.1 Transforming Optimiser States

Practically, the core consideration is that of the various optimisers' running statistics. These include the momentum of momentum gradient descent [55], the first- and second-order moment moving-average estimates of the ADAM [56] and ADAM-W [57] optimisers, among many others. Several can be exactly transformed under a basis change to ensure invariance; however, some must be approximated. This requires consideration of each mathematical formulation and its motivation to determine which tensor rank the state should transform under. This section will consider derivations of the momentum and ADAM state transforms.

For simplicity, consider a vector parameter, $\vec{a}$, being updated by an optimiser of form $\vec{a}_{n+1} = \vec{a}_n - \eta\vec{U}(\vec{a}_n)$. Assume the network of interest has a construction such that it exhibits forward-pass invariance to orthogonal transforms on the parameter $\vec{a}$, denoted: $\vec{a}' = \mathbf{R}\vec{a}$. For the isotropic MLP, this transform is compensated for elsewhere, such as in the next and previous layers; for example, the diagonalisation of *Sec.* 2.2. Due to the designed function invariance, the loss is similarly invariant, $\mathcal{L}'(\vec{a}') = \mathcal{L}(\vec{a})$ as long as any additional terms are assumed also invariant, such as regularisers, e.g. $\mathcal{L}_2 \propto \vec{a}^T\vec{a}$. Naturally, we may also want the optimisation trajectories to be invariant under basis changes, as this acts like a new coordinate system and representation for the matrices.

Therefore, to ensure continued invariance under this reparameterisation, $\mathcal{L}'(\vec{a}'_n) = \mathcal{L}(\vec{a}_n)$, for all n, requires the trajectories to be equivariant to the reparameterisation transform. This is described in *Eqn.* 22, and effectively uses *Eqn.* 1 applied to a general optimiser.

$$\vec{a}'_{n+1} = \left(\vec{a}'_n - \eta\vec{U}'(\vec{a}'_n)\right) = \left(\mathbf{R}\vec{a}_n - \eta\mathbf{R}\vec{U}\left(\mathbf{R}^T\vec{a}'_n\right)\right) = \mathbf{R}\vec{a}_{n+1} \quad \Rightarrow \quad \vec{U}'(\vec{a}') = \mathbf{R}\vec{U}\left(\mathbf{R}^T\vec{a}'\right) = \mathbf{R}\vec{U}(\vec{a}) \tag{22}$$

Hence, network and loss invariance under an orthogonal transform motivates an optimiser algebraic-equivariance condition on the optimiser dynamics under reparameterisation $\vec{U}'(\vec{a}') = \mathbf{R}\vec{U}(\vec{a})$. This is an intertwining relation between the original and transformed optimiser states, that reduces to a standard algebraic equivariance when $\vec{U}' = \vec{U}$. Standard gradient descent satisfies this trivially by the chain rule.

$$\vec{U}'(\vec{a}') \equiv \nabla_{\vec{a}'}\mathcal{L}'(\vec{a}') = \left(\frac{\partial\vec{a}'}{\partial\vec{a}}\right)^{-T}\nabla_{\vec{a}}\mathcal{L}(\vec{a}) = \mathbf{R}\nabla_{\vec{a}}\mathcal{L}(\vec{a}) = \mathbf{R}\vec{U}(\vec{a}) \tag{23}$$

Overall, this can be understood intuitively: if $\nabla_\varrho\mathcal{L}$ is locally the direction of steepest descent in parameter space, for parameter ϱ, then transforming to a new orthonormal basis in parameter space, $\varrho' = \mathbf{R}\varrho$, does not change the vector of steepest descent, only its representation in the chosen basis — the underlying loss surface is invariant due to function invariance.

However, this must be reevaluated for momentum, $\vec{U}_{\text{mome.}}$, and ADAM optimisers, $\vec{U}_{\text{ADAM}}$. For momentum, with factor $\alpha \in [0, 1]$, this requires a transform to the update vector as $\vec{M}'_n = \mathbf{R}\vec{M}_n$. Hence, this transforms the same as the underlying parameter, $\vec{a}$, as may be expected.

$$\vec{U}_{\text{mome.}} = \vec{M}_n \quad , \quad \vec{M}_{n+1} = \alpha\vec{M}_n + (1-\alpha)\nabla_{\vec{a}}\mathcal{L}(\vec{a}) \tag{24}$$

$$\text{requiring:} \quad \vec{M}'_{n+1} = \mathbf{R}\vec{M}_{n+1} = \alpha\mathbf{R}\vec{M}_n + (1-\alpha)\underbrace{\mathbf{R}\nabla_{\vec{a}}\mathcal{L}(\vec{a})}_{\nabla_{\vec{a}'}\mathcal{L}'(\vec{a}')} \tag{25}$$

For ADAM, the first moment transforms, $\vec{M}$ identically, $\vec{M}'_n = \mathbf{R}\vec{M}_n$; however, the second moment itself is a diagonal approximation of a true second moment due to elementwise approximations. *Eqn.* 26 displays ADAM in elementwise form, with the ε-positivity term neglected for simplicity.

$$\vec{U}_{\text{ADAM}} = \sum_i \frac{\vec{M}_n \cdot \hat{e}_i}{\sqrt{\vec{V}_n \cdot \hat{e}_i}} \hat{e}_i \qquad \begin{cases} \vec{M}_{n+1} = \beta_1 \vec{M}_n + (1-\beta_1)\, \nabla_{\vec{a}} \mathcal{L}\left(\vec{a}\right) \\ \vec{V}_{n+1} = \beta_2 \vec{V}_n + (1-\beta_2)\left(\nabla_{\vec{a}} \mathcal{L}\left(\vec{a}\right) \odot \nabla_{\vec{a}} \mathcal{L}\left(\vec{a}\right)\right) \end{cases} \tag{26}$$

This product of gradients term in $\vec{V}_n$ behaves as the diagonal of a rank-2 tensor quantity in terms of transform: $\tilde{V} = \left(\nabla_{\vec{a}} \mathcal{L}\left(\vec{a}\right) \odot \nabla_{\vec{a}} \mathcal{L}\left(\vec{a}\right)\right) = \operatorname{diag}\left(\nabla_{\vec{a}} \mathcal{L}\left(\vec{a}\right) \left(\nabla_{\vec{a}} \mathcal{L}\left(\vec{a}\right)\right)^T\right)$, considering that it is a projection of the outer product of two rank-1 tensors. Thus, the elementwise squaring of the gradient induces basis dependence by removing off-diagonal information, which is irrecoverable. Hence it is proposed to transform it as a rank-2 matrix but preserve the off-diagonal zero projection, as shown in *Eqn.* 27, where $\mathbf{R}^{\odot 2} = \mathbf{R} \odot \mathbf{R}$ is the an elementwise square of the matrix, while $\operatorname{diag} : \mathbb{R}^{n \times n} \to \mathbb{R}^n$ is the canonical diagonal function which extracts the matrix diagonal into a vector, and the reverse process $\operatorname{diag}^\dagger : \mathbb{R}^n \to \mathbb{R}^{n \times n}$ which embeds a vector along a matrix diagonal.

$$\tilde{V}' = \operatorname{diag}\left(\mathbf{R}\left(\operatorname{diag}^\dagger \tilde{V}\right)\mathbf{R}^T\right) = \mathbf{R}^{\odot 2}\tilde{V} = \left(\mathbf{R} \odot \mathbf{R}\right)\tilde{V} \tag{27}$$

For hyperoctahedral-prescribed networks, including the more typical permutation-prescribed networks, all these optimiser transforms are exact and consistent with a (signed) relabelling. This includes ADAM's second moment. Hence, typically, networks do not exhibit a distinction between the forward pass reparameterisation symmetry contrasted with a backward pass broken reparameterisation symmetry; their optimisation trajectories remain equivariant and exhibit a training-time function invariance, $\vec{a}'_n = \mathbf{P}\vec{a}_n$, for all n and constant permutation matrix $\mathbf{P}$. Yet this is not the case for orthogonal transforms; the second-moment transform is only approximate. The network is symmetric under orthogonal reparameterisations during the forward pass, but it couples with certain optimisers in a non-invariant manner, such that reparameterisations yield different optimisation trajectories.

Nevertheless, one should still apply the approximate transform to ADAM's second moment to avoid inappropriate scaling of the effective loss surface, and it is proposed that this approximated rank-2 transform is more consistent with the quadratic gradient term than treating it as a rank-1 quantity $\vec{V}' = \mathbf{R}\vec{V}$. Hence, an inference-time reparameterisation orthogonal symmetry exists, but concurrently, a training-time equivalent symmetry does not generally exist.

Briefly, Muon [58] with its natural orthogonal construction appears well-suited to such orthogonal basis changes. It takes a first-order moment $\mathbf{M} \in \mathbb{R}^{n \times m}$ on the weight's gradient and adapts it via Newton–Schulz — approximately equivalent to setting the moment's singular values to one via SVD. Hence, as long as this first-order moment transforms appropriately under our orthogonal basis changes (analogous to the weights' own transformation), this operator should remain unaffected in the trajectories it produces.

Overall, the practical takeaway is that, provided an optimiser is equivariant under the symmetry and the function and loss are invariant under it, an appropriate reparameterisation of the model and optimiser yields an equivariant trajectory.

A.2 Tensor Contraction Inequivalence

A separate, classical route in which reparameterisation can result in function invariance but optimiser non-invariance is considering the presence and absence of various tensor contractions, such as whether the SVD decomposition preserves $\mathbf{\Lambda Q}^T$ as separate or contracted $\mathbf{W}' = \mathbf{\Lambda Q}^T$. This consideration is more broadly pertinent, including implications for the normaliser's own rescaling parameterisations (typically diagonally-constrained affine maps). Generally, this arises when optimisation is performed on the parameter space, with these non-contracted maps increasing dimensionality, thereby altering the loss surface and steepest descent, resulting in different optimiser trajectories.

Considering the map $\vec{y}_m = \mathbf{W}_{mn}\vec{x}_n$ with decomposition $\mathbf{AB} = \mathbf{W}$, one can see two different functions after a gradient step when assuming standard gradient descent and notation $\vec{g} = \nabla_{\varrho}\mathcal{L}$ for appropriate parameter ϱ.

$$\vec{y}'_m = (\mathbf{W}_{mn} - \eta\vec{g}_m\vec{x}_n)\,\vec{x}_n \qquad \Rightarrow \qquad \Delta\vec{y} = \vec{y}' - \vec{y} = -\eta\,\|\vec{x}\|_2^2\,\vec{g} \tag{28}$$

$$\vec{y}'_m = (\mathbf{A}_{mk} - \eta\vec{g}_m\mathbf{B}_{ki}\vec{x}_i)\,(\mathbf{B}_{kn} - \eta\vec{g}_j\mathbf{A}_{jk}\vec{x}_n)\,\vec{x}_n \tag{29}$$

$$\Rightarrow \qquad \Delta\vec{y} = \vec{y}' - \vec{y} = -\eta\left(\vec{g}\,\|\mathbf{B}\vec{x}\|_2^2 + \|\vec{x}\|_2^2\,\mathbf{A}\mathbf{A}^T\vec{g}\right) + \eta^2\,\|\vec{x}\|_2^2\,(\vec{x}^T\mathbf{B}^T\mathbf{A}^T\vec{g})\,\vec{g} \tag{30}$$

This result shows that leaving a layer in a *decomposed* diagonalised parameterisation changes optimisation.

The approach used in this paper is to decompose the specified weight matrix into three temporary matrices every epoch, take appropriate neuroadaptive/sparsification actions, and then immediately recombine the temporaries into the appropriate contracted matrices before continuing training. Consequently, one does not optimise over $\mathbf{R}$, $\mathbf{Q}$ and $\mathbf{\Lambda}$ but over $\mathbf{W}_i$ and $\mathbf{W}'_i$ instead for before and after the neuroadaptive actions respectively. We do this contraction because optimisation will likely change the subsequent SVD rotation factors and thus require contraction and re-decomposition anyway, while also improving efficiency by reducing the trainable parameter count. Thus, $\mathbf{R}$, $\mathbf{Q}$ and $\mathbf{\Lambda}$ never become persistent trainable parameters in this setup.

A.3 Null-Like Reparameterisations

Of greater significance for this paper are the null-like reparameterisations that arise when the singular values of a partially or fully diagonalised layer tend to zero, or are created at zero. Many such reparameterisations leave the network functionally identical; yet couple to the optimisation in non-trivial ways. This introduces a mode to steer the training of such networks by explicitly breaking symmetry through the chosen reparameterisations/initialisation. Considering the map of *Eqn.* 31, representing a two-layer MLP with a diagonalised first layer and isotropic activation $\mathbf{f}(\vec{x}) = \bar{\sigma}\left(\vec{x}^T\vec{x}\right)\vec{x}$.

$$\vec{y}_i = \mathbf{W}_{ij}\left[\mathbf{f}(\vec{z})\right]_j + \vec{b}_i^{(2)} \qquad\qquad \vec{z}_j = \vec{\Lambda}_j\vec{x}_j + \vec{b}_j^{(1)} \tag{31}$$

Computing the gradients of $\mathbf{W}$ and $\vec{\Lambda}$ yields *Eqn.* 32.

$$\frac{\partial\mathcal{L}}{\partial\mathbf{W}_{pq}} = \vec{g}_p\left[\mathbf{f}(\vec{z})\right]_q \qquad \frac{\partial\mathcal{L}}{\partial\vec{\Lambda}_p} = \vec{g}_i\mathbf{W}_{ij}\mathbf{J}^{\mathbf{f}}_{jp}\vec{x}_p \qquad \frac{\partial\mathcal{L}}{\partial\vec{b}_p^{(1)}} = \vec{g}_i\mathbf{W}_{ij}\mathbf{J}^{\mathbf{f}}_{jp} \tag{32}$$

$$\mathbf{J}^{\mathbf{f}}_{jp}(\vec{z}) = \bar{\sigma}\left(\vec{z}^T\vec{z}\right)\delta_{jp} + 2\bar{\sigma}'\left(\vec{z}^T\vec{z}\right)\vec{z}_j\vec{z}_p$$

These null-like reparameterisations emerge as a singular value tends to zero, $\vec{\Lambda}_0 \to 0$ with bias $\vec{b}_0^{(1)} \to 0$. Then as a consequence $\vec{z}_0 \to 0$ and $\left[\mathbf{f}(\vec{z})\right]_0 \to 0$ and from this a function-invariant unconstrained reparameterisation of $\mathbf{W}_{i0}$ is admitted.

By numerical substitution into *Eqn.* 32, one can determine that such a reparameterisation not only couples with the optimiser but produces a zero-valued fixed point for both $\mathbf{W}_{i0}$ and $\vec{\Lambda}_0$ — stalling training. Hence, careful consideration of such null-like reparameterisations is essential, especially for neurogenesis initialisations, and an explicit symmetry breaking of $\mathbf{W}_{i0} \in \mathbb{R}^n \setminus \{\vec{0}\}$ is necessary. However, any non-zero choice biases the optimisation trajectory accordingly. Hence, a 'no-free-lunch' result arises; one must bias the behaviour of the network's new neuron through a choice of $\vec{W}^+ \neq \vec{0}$ or $b^+ \neq 0$. To name several approaches, one may choose $\vec{W}^+ \neq \vec{0}$ randomly, or a vector in the existing column-vector span of $\mathbf{W}$, or create a signed-bias in $b^\star$ or take it to equal the current intrinsic length, $b^\star = \pm\sqrt{o}$ — many valid initialisations can help avoid this vanishing gradient problem, all with optimisation consequences.

B Considerations of Normalisation

Within isotropic multilayer perceptrons, one may wish to implement a form of normalisation, especially post-composed with the activation function. Isotropic normalisers then mitigate activations growing and compensating for small singular values, which would affect the negligible impact assumption of $\mathbf{\Lambda}_{ii} \to 0$. Isotropic-tanh, $\sigma(a) = \tanh a$, similarly mitigates this by saturating to 1. However, BatchNorm [59], LayerNorm [60], among others, do not exhibit standard orthogonal algebraic (*Eqn.* 1) equivariance re-individuating neurons despite isotropic activation functions. RMSNorm [61] does display this orthogonal equivariance, but pre- and post-composition with an isotropic activation presents difficulties because the vector magnitude is constant.

This is displayed in *Eqn.* 33 and *Eqn.* 34, for pre- and post-composition of L_2-Norm (scaled RMSnorm), denoted as $\mathbf{N}(\vec{x}) = \vec{x}/\|\vec{x}\|_2$ (neglecting ε) and generic isotropic activation function, $\mathbf{f} = \bar{\sigma}\left(\vec{x}^T\vec{x}\right)\vec{x}$, for $\bar{\sigma} : \mathbb{R}^+ \to \mathbb{R}^+$ and constant $C \in \mathbb{R}$. This section assumes parameterless normalisers with a generally neglected, negligible ε for notational simplicity.

$$\mathbf{f}(\mathbf{N}(\vec{x})) = \bar{\sigma}\left(\hat{x}^T\hat{x}\right)\hat{x} = \bar{\sigma}(1)\,\hat{x} = C\hat{x} = C\mathbf{N}(\vec{x}) \tag{33}$$

$$\mathbf{N}(\mathbf{f}(\vec{x})) = \frac{\bar{\sigma}\left(\vec{x}^T\vec{x}\right)\vec{x}}{\|\bar{\sigma}\left(\vec{x}^T\vec{x}\right)\vec{x}\|_2} = \frac{\bar{\sigma}\left(\vec{x}^T\vec{x}\right)\vec{x}}{|\bar{\sigma}\left(\vec{x}^T\vec{x}\right)|\,\|\vec{x}\|_2} = \frac{\vec{x}}{\|\vec{x}\|_2} = \hat{x} = \mathbf{N}(\vec{x}) \tag{34}$$

Consequently, $\mathbf{N} \circ \mathbf{f} = \mathbf{N}$, $\mathbf{f} \circ \mathbf{N} = C\mathbf{N}$, alongside $\mathbf{N}$'s idempotence $\mathbf{N} \circ \mathbf{N} = \mathbf{N}$. Thus, when implementing normalisation with adaptive isotropic networks, we encourage alternative radial normalisation, such as reintroducing batch statistics. This is described in *Eqn.* 35, which acts like RMSNorm/L_2-Norm using a batched estimate of the mean norm, $\mathbb{E}_{j\in B}\left[\|\vec{x}_j\|_2\right]$ and samples $\vec{x}_i$.

$$N_b(\{\vec{x}_i\}) = \frac{\vec{x}_i}{\mathbb{E}_{j\in B}\left[\|\vec{x}_j\|_2\right]} \tag{35}$$

Similarly, 'FrobNorm' over a matrix of batched samples $\mathbf{X} \in \mathbb{R}^{n\times b}$, using a Frobenius norm $\|\cdot\|_F$, could be used to similar effect, shown in *Eqn.* 36, with a specific instance for RMSNorm comparability.

$$N_f(\mathbf{X}) \propto \frac{\mathbf{X}}{\|\mathbf{X}\|_F} \qquad \left(\text{e.g.} = \frac{\mathbf{X}}{(b\sqrt{n})^{-1}\|\mathbf{X}\|_F}\right) \tag{36}$$

LayerNorm, $\mathrm{LN} : \mathbb{R}^n \to \mathbb{R}^n$, does exhibit an equivariance to $\mathrm{O}(n-1)$, with a $\varrho : g \to \mathrm{GL}_n(\mathbb{R})$ representation which leaves $\vec{1}$ fixed: $\left\{\varrho'(g)\,|g \in \mathrm{O}(n), \varrho'(g)'\,\vec{1} = \vec{1}\right\} \cong \{\varrho(g)\,|g \in \mathrm{O}(n-1)\}$. In general, this ordinarily prohibits the usage of SVD diagonalisation, which necessitates a full $\mathrm{O}(n)$ equivariance; however if one generalises the form of LayerNorm to parameterise mean as instead a weighted-mean, then a joint equivariance to $\mathrm{O}(n)$ is restored through a direct-sum. This generalised form is displayed in *Eqn.* 37, $\mathrm{LN}_{\hat{n}} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$, with equivariance displayed in *Eqn.* 38 for standard representation, notated in shorthand as $\forall \mathbf{R} \in \mathrm{O}(n)$. A further *Eqn.* 1 prescription for normalisers. We neglect rescaling parameters and small non-zero epsilon for notational clarity.

$$\mathrm{LN}_{\hat{n}}(\vec{x};\hat{n}) = \frac{\vec{x} - \hat{n}\hat{n}^T\vec{x}}{\|\vec{x} - \hat{n}\hat{n}^T\vec{x}\|_2} \tag{37}$$

$$\mathrm{LN}_{\mathbf{R}\hat{n}}(\mathbf{R}\vec{x};\mathbf{R}\hat{n}) = \mathbf{R}\mathrm{LN}_{\hat{n}}(\vec{x};\hat{n}) \tag{38}$$

Thus, considering *Eqn.* 38, one may use LayerNorm in the neuroadaptive isotropic multilayer perceptron, since the weighted mean transforms as *Eqn.* 38 under an orthogonal basis change.

Similar generalisations may be considered to reinterpret parameterless BatchNorm, which typically exhibits standard hyperoctahedral algebraic equivariance. In this formula, each vector element is divided by its standard deviation across the batch. This elementwise approximation to full covariance-whitening induces hyperoctahedral equivariance — analogous to the second-moment elementwise

approximation of the ADAM optimiser [56]. Typical composed parameterisations also break this symmetry down to a closure form of the symmetry (similarly, care is needed for parameterising *Eqn.* 37).

However, typically BatchNorm is used in sequence with surrounding weights. These weights are closed under orthogonal transformations too, separate from the elementwise activations. Hence, training may naturally explore rotated representations. Consequently, the reference basis used to compute diagonal (elementwise) variance may shift during training (also for the non-linearity). Thus, one may view BatchNorm as a batch estimate of the diagonal of the full covariance matrix. The full covariance matrix transforms as a rank-2 tensor $\mathbf{C}' = \mathbf{R}\mathbf{C}\mathbf{R}^T$, with its diagonal approximately transforming as $\vec{C}' = (\mathbf{R} \odot \mathbf{R})\,\vec{C}$ for Hadamard product $\odot$, as previously derived in *Eqn.* 27. These approximate transformations may be reasonable for three reasons: first, training may induce comparable rotations of the representation regardless; second, BatchNorm already relies on noisy batch-level estimates, so the approximate transformation may not introduce qualitatively different noise during training; and third, transformation to running statistics need not be undertaken during training, since each forward pass recomputes the relevant diagonal batch statistics in the current basis.

Thus, one may argue that since training-time evolution will likely continuously rotate the basis in which the covariance is diagonally approximated anyway, at least until the weights stabilise, this may justify an explicit basis transform used in the diagonalisation too. Hence, this treats BatchNorm with an approximate $\mathrm{O}(n)$ equivariance, rather than permutation S_n equivariance only — especially during training time.

Similarly, one may periodically take the post-activation function vectors over a chosen set of inputs and determine the true covariance. Singular value decomposition of this symmetric matrix yields orthogonal matrices, which, in an isotropic network, can then act on the weights to produce representations in a new basis. In this basis, BatchNorm no longer produces a diagonal approximation to the whitening transform, but the exact whitening transform in reference to the chosen set due to the diagonalised covariance matrix. This is maintained at least until further training. At late training/inference time, this explicit transform may be more problematic because weights may have settled into a consistent basis for the diagonalised covariance, which downstream parameters may have specialised to. Therefore, we propose regularly reparameterising weights to diagonalise the covariance during training to mitigate the risk of downstream parameters specialising on a specific approximation. At inference time, the transform can ensure diagonal whitening is equivalent to a full whitening transform.

At inference time, the estimated statistics must also be transformed appropriately, with mean, $\vec{\mu}$, transforming as a rank-1 quantity $\vec{\mu}' = \mathbf{R}\vec{\mu}$ and standard deviation, $\vec{\sigma}$ as a diagonal rank-2 approximation $\vec{\sigma}' \odot \vec{\sigma}' = (\mathbf{R} \odot \mathbf{R})\,(\vec{\sigma} \odot \vec{\sigma})$, then taking the elementwise square-root to recover $\vec{\sigma}'$ — or recompute this diagonal from scratch as a stored statistic. If one wishes to undertake this with neuroadaptation, then SVD weight-diagonalisation and adaptation must precede the transform that diagonalises the covariance matrix. This treatment of BatchNorm not as elementwise standardisation with permutation equivariance, but as diagonally approximated full-whitening with approximate orthogonal equivariance, is important for enabling the latter independent component analysis approach on pre-whitened vectors, as detailed further in *App.* F — especially in tandem with a network which is regularly reparameterised to produce layerwise diagonalised covariance during training to prevent basis-specialisation.

Separately, this indicates the taxonomy's utility even when compositional symmetries are absent, i.e. two surrounding weights are closed orthogonally, even though the intermediate activation breaks orthogonal algebraic equivariance symmetry, preventing the parameter symmetries or broader model symmetries. Yet, this unusual construction can thus be analysed with separate building blocks from the taxonomy, and composed together, revealing how one can handle BatchNorm's symmetry-breaking properties to behave favourably in this context. The continuous change of basis conducted in neuroadaptation and/or diagonalisation periodically during training also helps achieve a similar effect for elementwise approximations used within optimisers — they may better approximate full standardisation than what their diagonal approximations initially intended. The training-time orthogonal closure of weights likely also causes unaccounted for effective basis changes to optimiser running statistics which it is unaffected by, encouraging the approximate second-order transforms proposed. Whilst periodic changes of basis may reduce an optimiser's dependence on any single coordinate system.

C Alternative Methodologies to Thresholding

The specified symmetry construction determines the form of neuroadaptation described and its function-preserving properties, but it does not uniquely determine when to perform it. This paper's overall approach uses scheduled and conditional neuroadaptation, but these methods and the ones enclosed are changeable. Alternatives can be explored more broadly while still using the function-preserving neuroadaptation process. The clearest alternatives are demonstrable in the current method's thresholding formulations.

Here, thresholding is used to indicate when a neuron has an insignificant contribution to the overall model function, and therefore constitutes a so-called 'scaffold neuron' — a neuron which is available to specialise and learn if required, ultimately allowing the possibility for a network to expand its width/capacity if thresholding is surpassed, but until then is deemed otherwise negligible.

This is an important part of this particular approach to scheduled, conditional, neuroadaptation, but has many possible formulations. This section expands on several alternatives, comparing pros and cons, contextualised through specific isotropic activations and their spectral behaviours. The discussion starts by comparing different statistical strategies, then how these may be applied to activation or gradient information, or complement them with batch statistics.

Arguably the most straightforward approach is the one utilised in the main text and empirical section: setting a constant value against which singular values are compared. The chosen value was 0.95 — principally used to indicate whether a singular value amplifies or attenuates the prior activation in its map (assuming prior uniformity, aided somewhat by magnitude saturation). It is interpretable, given the use of orthogonal initialisation that begins the singular values at 1, so a small offset below this, such as the decrease to 0.95, is meaningful because it shows declining impact. The exact value of 0.95 is not inherently principled; any similar $1 - \epsilon$ consideration likely suffices. Overall, the benefits of this approach are that it is trivial to understand and is not elaborate, which minimises the risk of introducing additional complicated failure modes. Therefore, it remained an appropriate choice in the empirical section because its focus on mechanistic verification benefits from a minimalistic approach. However, this approach has downsides: a fixed cut is not adaptive to overall singular value spectral shifts in the data, which may occur during training, nor does it account for a magnitude-saturating activation function, which may make the relative proportions of singular values more meaningful than their absolute values. Hence, spectral information and statistics appear to be a favourable improvement, albeit less minimalistic.

Consequently, a natural extension is to use statistics such as the mean, median, standard deviation or maximum of singular values in the thresholding formulation. First to consider is a mean multiplier $c\mathbb{E}\left(\sigma_i\right)$, a simple method that deems a neuron unimportant if its singular value is substantially smaller than the mean. Its positives include its simplicity and, crucially, scalability to overall spectral shifts—particularly useful if a layer operates at a different scale. However, large outliers may be problematic; even a few can produce aggressive neurodegeneration in the network. This is addressed by using a median, which mitigates the risk of large outliers while retaining scalable thresholding behaviour; however, its multiplier is rather unintuitive and distribution-dependent in how many scaffold neurons it may classify. A similar point applies to a multiplier of the maximum, which is also separately highly susceptible to outliers.

Mean minus a multiplier of standard deviations, or comparably median minus a multiplier of median absolute deviation, may be more robust and interpretable as a threshold, selecting small tails of the singular value distribution. Setting the multiplier sufficiently high indicates statistically unusually low singular values, which could be interpreted as providing relatively little function (assuming roughly uniform-scale pre-activations). This also favours saturating isotropic activation functions, as it better measures relative contribution. Despite this, singular value distributions are non-negative, and the observed spectra in *Fig.* 2 may be significantly non-Gaussian. This could, in principle, cause negative thresholding and result in uncontrolled neurogenesis.

Briefly, quantiles are largely incompatible with the scaffold-neuron-triggered neuroadaptation described, as they would fix the number of neurons at all times and therefore never trigger such processes. Comparably constant, predefined growth and pruning should be avoided due to the repeated replacement of small singular values; thus, a reactive implementation is essential.

On relative methods: setting a threshold against a relative value such as $\lambda_i = N\sigma_i / \sum_j \sigma_j$ or $\lambda'_i = \sqrt{N}\sigma_i / \sqrt{\sum_j \sigma_j^2}$ may be practical, especially for the aforementioned case of saturating activation functions, and it is also scale-invariant. Here, N indicates a layer width $N = \min(m, n)$ for the weights $\mathbb{R}^{n \times m}$, which must be accounted for if consistent scaling is desired. Despite this, this thresholding approach remains strongly affected by large outliers.

Finally, the spectral gap approach appears promising, though more implementationally nuanced, for selecting scaffold neurons. This appears especially attractive given the larger spectral gaps observed across the singular value time-evolution plots, shown in *Fig.* 2 and the range of results of *App.* K.2. For example, one could threshold at the first spectral gap position from zero and deem all diagonalised neurons below it as 'scaffold-neurons'. Retaining a limited number would preserve the gap's presence, while also enabling these neurons to transition from the scaffold-band, across the band-gap, to the active-band—then triggering neurogenesis, and vice versa. This seems especially well suited to the observed spectra, and we propose it as the most promising practical implementation outside of mechanistic verification. Its complexity, especially in quantifying such spectral gaps, may introduce peculiar failure modes tied to this thresholding approach, although the nature and mitigation of these can be explored in subsequent work. Conversely, if no persistent gap forms in the spectrum, such as in a very dense case, this method may be unsuitable; however, gaps appeared persistent in the isotropic results of *App.* K.2.

This concludes a non-exhaustive list of thresholding approaches. The remaining paragraphs clarify how these may relate directly to singular values or their gradients and how batch statistics may enrich these categorising methods.

One canonical separation in judging importance is activations-based thresholding or thresholding on their gradients. Generally, such considerations are assumed to apply only to activations, but they may also generalise to gradients. Neuroadaptation is suggested to run at scheduled intervals alongside standard gradient-based parameter optimisation; this means gradient information is readily available to the thresholding stage. A nuance is that the activation-gradient may not be in the diagonalised-neuron basis, but it can be trivially transformed by performing SVD on the corresponding weights, then applying the corresponding orthogonal transforms to the available gradients. Composing this with the gradient of the outputs to their squared-sum, enables a measure of how much that diagonalised neuron impacts functionality. Alternatively, one could consider the magnitude of the gradient with respect to the loss [19], among others. Across these, many of the above thresholding methods can be employed to such gradients to determine outliers. For example, having a gradient within a small multiple of the standard deviation around zero would suggest little impact on the function's outputs — thus it may be considered a scaffold neuron. Various other permutations could be considered, offering practical avenues to improve when neuroadaptation should occur.

One could also use these methods to examine whether rarely significant connections tend to produce non-robust maladaptations in the output, rather than a simple lack of impact, when preceding activations are not assumed to be uniform.

Finally, batch information could represent an underexplored indicator that could potentially complement the discussed approaches. For example, batch statistics coupled with such methods could indicate variability in a neuron's significance: if a diagonalised neuron's resultant activation is consistently low, even across a batch, then it is arguably a better candidate for being a scaffold neuron. This also relates to removing the assumption of pre-activation uniformity.

Overall, this discussion indicates many alternatives to thresholding methods that may be productive to explore. The current minimalistic approach is preferable for the intended mechanistic verification and for reducing obscure failure modes, but alternatives may remain preferable if the intention is for competitive or production usage, and further exploration of these approaches and others not listed is encouraged.

D Parallelising Matrix-Vector Products and Nested Functional Classes

Generalising isotropic multilayer perceptrons yields an asymptotic nested functional class, much like residual networks, but arising from the functional form permitted by the chosen orthogonal symmetry. This is given by all possible isotropic networks of depth d being denoted as the set $\mathcal{F}_d$ such that $\mathcal{F}_d \subseteq \mathcal{F}_{d+1}$. This structure may be significant, offering an alternative to classical residual [62] connections, with an architectural constraint to maintain or monotonically increase width with depth, and a saturating isotropic function. Moreover, this approach clarifies a crucial decomposition showing that all matrix-vector computations can be parallelised up front for isotropic multilayer perceptrons.

Considering the two-layer isotropic affine map provided by *Eqn.* 39 and then re-expressing this equation using notation $\mathbf{f} = \bar{\sigma}\big(\|\vec{x}\|_2^2\big)\vec{x}$ for preactivation $\vec{x}$.

$$\vec{x}^{(d+1)} = \mathbf{W}^{(d+1)}\mathbf{f}\Big(\underbrace{\mathbf{W}^{(d)}\mathbf{f}\left(\vec{x}^{(d-1)}\right) + \vec{b}^{(d)}}_{\vec{x}^{(d)}}\Big) + \vec{b}^{(d+1)} \tag{39}$$

$$\vec{x}^{(d+1)} = \bar{\sigma}\left(\left\|\vec{x}^{(d)}\right\|_2^2\right)\mathbf{W}^{(d+1)}\vec{x}^{(d)} + \vec{b}^{(d+1)} \tag{40}$$

Then expanding this recurrence relation, yields *Eqn.* 41 for two layers, for new contracted variables $\mathbf{W}'$ and $\vec{b}^{(d)'}$.

$$\vec{x}^{(d+1)} = \bar{\sigma}\left(\left\|\vec{x}^{(d)}\right\|_2^2\right)\bar{\sigma}\left(\left\|\vec{x}^{(d-1)}\right\|_2^2\right)\underbrace{\mathbf{W}^{(d+1)}\mathbf{W}^{(d)}}_{=\mathbf{W}^{(d+1)'}}\vec{x}^{(d-1)} + \bar{\sigma}\left(\left\|\vec{x}^{(d)}\right\|_2^2\right)\underbrace{\mathbf{W}^{(d+1)}\vec{b}^{(d)}}_{=\vec{b}^{(d+1)'}} + \vec{b}^{(d+1)} \tag{41}$$

Which can subsequently be generalised again for d-layer isotropic perceptron, as shown in *Eqn.* 42. For notational simplicity a further reparameterisation to $\mathbf{W}_\circ^{(d)}$ and $\vec{b}_\circ^{(d,i)}$ is utilisThe input $\vec{x}^{(0)}$ does not undergo $\mathbf{f}$'s non-linear transform due to it being the input; hence, indexing starts at 1.

$$\vec{x}^{(d+1)} = \vec{b}^{(d+1)} + S_1^d\mathbf{W}_\circ^{(d)}\vec{x}^{(0)} + \sum_{i=1}^{d} S_i^d\vec{b}_\circ^{(i,d)} \qquad S_m^d = \prod_{j=m}^{d}\bar{\sigma}\left(\left\|\vec{x}^{(j)}\right\|_2^2\right) \tag{42}$$

In this case, the layerwise architecture still matters for the $\bar{\sigma}$ terms, with the directionality of intermediate $\vec{x}$ also affecting the various non-linear terms. This reparameterisation generally constrains the parameters $\mathbf{W}_\circ^{(d)}$ and $\vec{b}_\circ^{(i,d)}$; however, it is suggested to broaden the functional class by allowing both to be new, trainable, real matrices of appropriate shape. Hence, instead of interdependence between $\mathbf{W}_\circ^{(d)}$ depthwise, they are now treated as independent parameters, enabling up to full rank matrices, not limited to being less than or equal to the minimum rank of the original matrices in the contraction — the independent formulations are not subject to constraint $\operatorname{rank}\left(\mathbf{W}_\circ^{(d)}\right) \leq \min_k \operatorname{rank}\left(\mathbf{W}^{(k)}\right)$. Thus, unconstraining to general real matrices broadens the functional class and also enables further parallelisation at training time instead of requiring an initial sequential contraction of the intermediate weight matrices. They may also be trained as independent weight matrices.

Under the assumptions of a maintained or monotonically increasing width, with a non-linear term of an isotropic activation function, $\bar{\sigma}$, capable of saturating to a non-zero constant, with intrinsic length parameter able to ensure such general saturation — then in this saturation limit, the parameters of layer $d+1$ are able to embed the parameters of layer d, thus reproducing its function in that embedded (sub)space. In the uncontracted parameters, this is equivalent to a multiplication by a scaled identity, where the scaling is the inverse of the constant-saturated value of the non-linearity, which now acts as a constant linear scaling. Thus, given this architectural arrangement and asymptotic saturation limit, increasing depth retains the expressibility of all shallower networks while exhibiting a strictly equal-to-enlarged function class. This is not the case for typical multilayer perceptrons.

Furthermore, one may note that by this construction, *all* matrix-vector products, over all layers, $\vec{z}^{(d)} = \mathbf{W}_\circ^{(d)} \vec{x}^{(0)}$, may be precomputed *in parallel* by producing the $\vec{z}^{(d)}$ contraction. This is displayed in *Eqn.* 43, showing sequential computation only need occur for inner products, $a = \vec{x}^{(d)} \cdot \vec{x}^{(d)}$ in $\mathcal{O}(n)$ per-layer-scaling for n width, a single non-linear operation, $\bar{\sigma}(a)$ in $\mathcal{O}(1)$ (compared to elementwise $\mathcal{O}(n)$ per-layer-scaling), scalar-vector product $\mathcal{O}(n)$ and vector-vector addition, $\mathcal{O}(n)$. This remains valid for the general $\mathbf{W}_\circ^{(d)}$ or $\vec{b}_\circ^{(i,d)}$ the original constrained forms: $\mathbf{W}_\circ^{(d)} = \prod_{i=1}^{d+1} \mathbf{W}^{(i)}$ and $\vec{b}_\circ^{(i,d)} = \left(\prod_{k=i+1}^{d+1} \mathbf{W}^{(k)}\right) \vec{b}^{(i)}$.

$$\vec{x}^{(d+1)} = \vec{b}^{(d+1)} + S_1^d \vec{z}^{(d)} + \sum_{i=1}^{d} S_i^d \vec{b}_\circ^{(i,d)} \tag{43}$$

Hence, the computationally demanding matrix–vector products are *all* parallelised and precomputed, whilst comparatively trivial scalar-vector products, inner products, vector additions and simple $\mathcal{O}(1)$ activations are sequential, potentially offering significantly improved GPU utilisation.

This parallelisation applies only to matrix-vector products; all other operations remain sequential. Therefore, this one parallelised aspect does not inherently imply overall suboptimality, particularly since highly non-trivial, parameterised non-linear behaviour persists sequentially through the scalar. Thus, this construction is characterised by deep nonlinear dependence but simultaneously not a deep matrix-multiplication formulation. Therefore, these results do not show how this affects expressivity, positively or negatively, but this reorganisation of sequential structure may offer a significant benefit through its parallelisations — with indications from *App.* K.6 that isotropic MLPs remain expressive, and moreover competitive, for more complicated classification tasks.

Between submission and acceptance of this paper, we developed these ideas into a distinct standalone workshop paper now accepted for the OPT2026 workshop at NeurIPS [63] that explores these principles more substantively alongside empirics.

E Architectural Generalisations

This section discusses how the isotropic-neuroadaptive approach could be generalised to architectures beyond multilayer perceptrons.

E.1 Sketching Adaptive-Channel Convolutional Networks

The procedure for isotropic-adaptive networks can be generalised for convolutional networks [39, 48]. This section sketches these results and will develop them in future work.

A dynamical convolution architecture is enabled by implementing isotropic activation functions, $\mathbf{f}$, channel-wise — taking the norm over vectors in $\mathbf{R}^C$ for C channels. In analogy to the multilayer perceptron setup, a partial diagonalisation of each convolution's kernel parameters, $\mathbf{K}_1 \in \mathbb{R}^{C_{\text{out}} \times (C_{\text{in}} \times K_h \times K_w)}$, is possible as shown by *Eqns.* 44 through 47, for $\mathbf{R} \in \mathrm{O}(C_{\text{out}})$, $\mathbf{Q} \in \mathrm{O}(C_{\text{in}} K_h K_w)$ and diagonal $\mathbf{\Lambda}$.

$$\mathbf{Y} = \mathbf{K}_2 * \mathbf{f}\left(\mathbf{K}_1 * \mathbf{X} + \vec{b}_1\right) + \vec{b}_2 \tag{44}$$

$$\mathbf{Y} = \mathbf{K}_2 * \mathbf{f}\left(\left(\mathbf{R}\mathbf{\Lambda}\mathbf{Q}^T\right) * \mathbf{X} + \vec{b}_1\right) + \vec{b}_2 \tag{45}$$

$$\mathbf{Y} = \underbrace{\mathbf{K}_2\mathbf{R}}_{=\mathbf{K}_2'} * \mathbf{f}\left(\underbrace{\mathbf{\Lambda}\mathbf{Q}^T}_{=\mathbf{K}_1'} * \mathbf{X} + \underbrace{\mathbf{R}^T\vec{b}_1}_{=\vec{b}_1'}\right) + \vec{b}_2 \tag{46}$$

$$\mathbf{Y} = \mathbf{K}_2' * \mathbf{f}\left(\mathbf{K}_1' * \mathbf{X} + \vec{b}_1'\right) + \vec{b}_2 \tag{47}$$

In effect, the spatial indices of convolution must be kept consistent, but rotations can commute with isotropic functions, thereby partially diagonalising channels and enabling the described adaptation.

Generalising the approach is non-trivial when considering both padding and residual constructions, requiring further development; yet this notated approach presents a route to broader generalisation. Similarly, such a fully diagonalised, partially diagonalised or ICA-based approach may offer interpretable insights into representations, with the caveat that representations may change under isotropic functions [32].

E.2 Attention Mechanisms

The self-attention mechanism [64, 65] within a transformer [65] architecture is also a candidate where neuroadaptation appears feasible. Single-headed self-attention is characterised by the equation presented in its canonical form in *Eqn.* 48 and in a tensor-indexed form in *Eqn.* 49 (notation following from Vaswani et al. [65]).

$$\text{Attention}\left(Q, K, V\right) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{48}$$

The following uses notation $\mathbf{W}^{(\mathbf{K})}, \mathbf{W}^{(\mathbf{Q})} \in \mathbb{R}^{d\times d_k}$, and $\mathbf{W}^{(\mathbf{V})} \in \mathbb{R}^{d\times d_v}$ for key, query and value weight matrices respectively and $\mathbf{X} \in \mathbb{R}^{T\times d}$ for the input (and T tokens). The subscript m on softmax indicate it acts over columns indexed by m.

$$\text{Attention}\left(\mathbf{X}; \mathbf{W}^{(\mathbf{K})}, \mathbf{W}^{(\mathbf{Q})}, \mathbf{W}^{(\mathbf{V})}\right) = \text{softmax}_{(m)}\left(\frac{\mathbf{X}_{ij}\mathbf{W}^{(\mathbf{Q})}{}_{jq}\mathbf{W}^{(\mathbf{K})}{}_{lq}\mathbf{X}_{ml}}{\sqrt{d_k}}\right)_{im}\mathbf{X}_{mp}\mathbf{W}^{(\mathbf{V})}{}_{pn} \tag{49}$$

Due to the $\mathbf{W}^{(\mathbf{Q})}\mathbf{W}^{(\mathbf{K})^T}$ contraction, an identity, $\mathbf{I}$, can be inserted between them and separated into a pair of orthogonal actions, $\mathbf{R}$, as follows: $\mathbf{I} = \mathbf{R}^T\mathbf{R}$. Thus, these exhibit a form of orthogonal closure (*Eqn.* 3) in their parameterisation. Consequently, one could perform a right- or left-sided partial diagonalisation of the respective weight, exposing its singular values and rank.

Then one can act with the neuroadaptation approach to change the dimension d_k within this self-attention. However, care is needed because the full closure is for the general linear group, $\text{GL}\left(d_k, \mathbf{R}\right)$, and can thus be used to reparameterise the matrices in a way that affects singular values. One could construct an agreed special representation for the parameters to threshold, such as balancing singular values between terms, as shown in *Eqn.* 50 (and resizing $\mathbf{U}\mathbf{\Lambda}^{\frac{1}{2}}$ and $\mathbf{V}\mathbf{\Lambda}^{\frac{1}{2}}$ such that they are $\mathbb{R}^{d\times d_k}$ matrices).

$$\mathbf{M} = \mathbf{W}^{(\mathbf{Q})}\mathbf{W}^{(\mathbf{K})^T} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T = \underbrace{\left(\mathbf{U}\mathbf{\Lambda}^{\frac{1}{2}}\right)}_{=\bar{\mathbf{W}}^{(\mathbf{Q})}}\underbrace{\left(\mathbf{\Lambda}^{\frac{1}{2}}\mathbf{V}^T\right)}_{=\left(\bar{\mathbf{W}}^{(K)}\right)^T} \tag{50}$$

Additionally, the d_k denominator needs addressing, such as rescaling both matrices by a factor $\sqrt[4]{d_k'/d_k}$.

Similarly, one could consider $\mathbf{W}^{(\mathbf{V})}$, where one could cautiously neuroadapt d_v by considering a representation of the orthogonal group $\text{O}\left(d_v\right)$ acting on the weight matrix subsequent to the multi-head concatenation, $\mathbf{W}^{(\mathbf{H})}$. Consequently, one could adapt each head independently, but it would end up with potentially varying d_v per head. Similarly, depending on the construction, general-linear symmetries may apply, with analogous considerations as before for d_k.

Trivially, the MLP aspects of a transformer could utilise neuroadaptation if multilayered — in exact comparison to the MLP implementation presented in this paper. This could occur for the other axis of $\mathbf{W}^{(\mathbf{H})}$, if it has a subsequent layer before recombination into the residual term. The dimension of $\mathbf{X}$ corresponding to d may be varied, if constructed as a multilayer $\mathbf{X} = \mathbf{W}\mathbf{Y}$, since singular value decomposition of $\mathbf{W}$ could have its orthogonal transforms simultaneously absorbed into $\mathbf{W}^{(\mathbf{Q})}, \mathbf{W}^{(\mathbf{K})}, \mathbf{W}^{(\mathbf{V})}$, enabling neuroadaptation. But it is essential to consider whether the particular architecture being used allows this to affect the residual stream.

Token dimensions and layers that combine into residual streams pose problems for neuroadaptation. The former should not be transformed with an orthogonal symmetry, since it represents discrete, time-ordered tokens. It is therefore a conceptually and mathematically poor candidate for neuroadaptation in any straightforward way. Similarly, reparameterisation that requires orthogonal transforms to commute with residuals adds considerable complexity, as many parameters at different locations require simultaneous orthogonal closure with the transform. Therefore, this requires substantial nuance to implement, which is out of scope for the present work.

Overall, several promising directions for neuroadaptation in transformer models exist. These allow several internal dimensions to change neuron counts while preserving function. However, token counts at present appear unsuitable for such approaches. This presents an interesting line of enquiry for future generalisations of this neuroadaptive approach.

E.3 Hybrid Architectures

As discussed in the neuroadaptability of attention, a network need not be wholly isotropic to permit a layer's neuroadaptation — two affine layers on either side of an isotropic activation are sufficient for adaptation, enabling the designed reparameterisation invariance. Therefore, once such properties and biases are suitably understood, one could construct other layers with various symmetry-led properties to induce a repertoire of desirable behaviours coexisting throughout the network in differing locations. Hybrid constructions are therefore permitted, where various primitives can be used in network construction.

F Independent Component Analysis (ICA) Network Representation

Independent component analysis (ICA) is a standard statistical technique for linearly separating a mixture of signals into often meaningful, statistically independent channels for interpretation — typically relying on non-Gaussianity for separation. This could be used for interpretable isotropic networks at inference time, up to the inherent concept resolvability of ICA itself for hidden-layer representations. Given interpretable components produced by ICA, one may monitor the activation elements to determine a network's current expression of internal concepts.

This approach may be utilised only on the premise that the multilayer perceptron is constructed from sequential blocks of affine-map $\rightarrow$ isotropic activation function $\rightarrow$ BatchNorm. It must then be represented in its covariance-eigenbasis, diagonalising the post-activation function's covariance. The network should also be trained such that it is not contingent on any particular basis inherited from BatchNorm's diagonal-covariance approximation but largely invariant to orthogonal transforms of the affine layers (discussed in the BatchNorm section of *App.* B). Thus, this ensures a pre-whitened signal ready for the subsequent ICA step.

One may then use various approaches to determine the orthogonal matrix required by ICA to transform the whitened data — such as a loss that encourages non-gaussianity and optimising in the corresponding $\mathfrak{so}(n)$ Lie algebra space. Having determined the appropriate orthogonal transform[5], $\mathbf{Q}$, one may reparameterise the network in one of three distinct approaches, given that an identity is postcomposed with BatchNorm $\mathrm{I} = \mathbf{Q}^T\mathbf{Q}$ and that the $\mathbf{Q}^T$ matrix is used to reparameterise the subsequent affine-layer weights: $\mathbf{W}'_{(n+1)} = \mathbf{W}_{(n+1)}\mathbf{Q}^T$. These proceed identically in algebra to the partial-diagonalisation insertion of an identity, but with a different pair of rotations and corrections to BatchNorm's statistics, detailed below. Transforms are detailed in *Eqns.* 51 through 56, for parameterless BatchNorm, $BN\left(\vec{x};\vec{\mu},\vec{C}\right)$ and isotropic activation function $\mathbf{f}$.

[5] I am grateful to Dmitry Kangin for our discussion of isotropic functions and their orthogonal equivariance, during which he reminded me of the orthogonal transformations arising from independent component analysis and suggested their potential relevance. This section subsequently develops that connection further.

$$\vec{x}_{(n+1)} = \mathbf{W}_{(n+1)}\vec{z}_{\text{whi.}} + \vec{b}_{(n+1)} \qquad \vec{z}_{\text{whi.}} = \text{BN}\left(\mathbf{f}\left(\mathbf{W}_{(n)}\vec{x}_{(n)} + \vec{b}_{(n)}\right); \vec{\mu}, \vec{C}\right) \tag{51}$$

$$\vec{x}_{(n+1)} = \mathbf{W}_{(n+1)}\Big(\underbrace{\mathbf{Q}^T\mathbf{Q}}_{=\text{I}}\Big)\vec{z}_{\text{whi.}} + \vec{b}_{(n+1)} \qquad \vec{z}_{\text{whi.}} = \text{BN}\left(\mathbf{f}\left(\mathbf{W}_{(n)}\vec{x}_{(n)} + \vec{b}_{(n)}\right); \vec{\mu}, \vec{C}\right) \tag{52}$$

$$\vec{x}_{(n+1)} = \underbrace{\mathbf{W}_{(n+1)}\mathbf{Q}^T}_{=\mathbf{W}'_{(n+1)}}\underbrace{\mathbf{Q}\vec{z}_{\text{whi.}}}_{=\vec{z}_{\text{ICA}}} + \vec{b}_{(n+1)} \qquad \vec{z}_{\text{whi.}} = \text{BN}\left(\mathbf{f}\left(\mathbf{W}_{(n)}\vec{x}_{(n)} + \vec{b}_{(n)}\right); \vec{\mu}, \vec{C}\right) \tag{53}$$

From this point, there is an explicit $\mathbf{Q}$ matrix product retained, which may remain as is, necessitating an additional matrix-vector product per sample, or be absorbed into prior parameters to increase efficiency. However, the latter option is nuanced, requiring the assumption that the network has been trained to be approximately invariant to basis changes to BatchNorm described in *App.* B if an elementwise $\vec{C}$ is desired, or promoting this to a full rank-2 transform requiring a covariance matrix $\mathbf{C}$ (particularly its inverse $\mathbf{C}^{-1}$).

$$\vec{x}_{(n+1)} = \mathbf{W}'_{(n+1)}\underbrace{\mathbf{Q}\vec{z}_{\text{whi.}}}_{=\vec{z}_{\text{ICA}}} + \vec{b}_{(n+1)} \qquad \vec{z}_{\text{whi.}} = \text{BN}\left(\mathbf{f}\left(\mathbf{W}_{(n)}\vec{x}_{(n)} + \vec{b}_{(n)}\right); \vec{\mu}, \vec{C}\right) \tag{54}$$

$$\vec{x}_{(n+1)} = \mathbf{W}'_{(n+1)}\vec{z}_{\text{ICA}} + \vec{b}_{(n+1)} \qquad \vec{z}_{\text{ICA}} = \text{BN}\Big(\mathbf{f}\Big(\underbrace{\mathbf{Q}\Big(\mathbf{W}_{(n)}\vec{x}_{(n)} + \vec{b}_{(n)}\Big)}_{=\mathbf{W}'_{(n)}\vec{x}_{(n)}+\vec{b}'_{(n)}}\Big); \vec{\mu}', \vec{C}' \text{ or } \mathbf{C}'\Big) \tag{55}$$

$$\vec{x}_{(n+1)} = \mathbf{W}'_{(n+1)}\vec{z}_{\text{ICA}} + \vec{b}_{(n+1)} \qquad \vec{z}_{\text{ICA}} = \text{BN}\Big(\mathbf{f}\Big(\mathbf{W}'_{(n)}\vec{x}_{(n)} + \vec{b}'_{(n)}; \vec{\mu}', \vec{C}' \text{ or } \mathbf{C}'\Big) \tag{56}$$

These latter reparameterisation approaches temporarily assume the orthogonal equivariance of the full-whitening and isotropic activation function, commuting with $\mathbf{Q}$ such that $\mathbf{Q}$ may be absorbed into the preceding affine layer's weights, $\mathbf{W}'_{(n)} = \mathbf{Q}\mathbf{W}_{(n)}$, and biases $\vec{b}'_{(n)} = \mathbf{Q}\vec{b}_{(n)}$.

Assumed to be in a covariance-diagonalised representation ready for inference-time, $\mathbf{C} = \text{diag}^{\dagger}\left(\vec{C}\right)$, one may then choose to appropriately transform the mean, $\vec{\mu}' = \mathbf{Q}\vec{\mu}$, and covariance matrices $\mathbf{C}' = \mathbf{Q}\mathbf{C}\mathbf{Q}^T$. If full $\mathbf{C}'$ is retained, this necessitates a matrix-vector product per sample for full equivariance, rather than the diagonalised vector-vector Hadamard product. Alternatively, one may choose to retain the efficient vector-vector product nature of BatchNorm by approximating the original function with a diagonal-approximated covariance $\vec{C}' = \left(\mathbf{Q} \odot \mathbf{Q}\right)^T \vec{C}$ (inexact). Either of the latter two methods seems preferable to the first exact approach, which adds an unnecessary matrix-vector product.

Table 2: Two primary approaches for a network in the 'ICA-representation' with respective trade-offs.

Variant	**Functional Invariance**	**Computational Cost**
Q absorbed with full covariance	Exact, under assumptions	Loses diagonal BatchNorm efficiency
Q absorbed with diagonal covariance	Approximate	Retains BatchNorm's efficiency

In both exact cases, the 'neurons' associated with the vectors decomposed in the standard basis now represent independent components as specified by ICA and can be interpreted accordingly. The inexact approach approximates such components given the BatchNorm diagonal preweighting approximation. Thus, this ICA representation of the network may enable real-time monitoring of the network's hidden states and interpretability therein, at least up to the inherent concept resolvability of ICA.

Alongside diagonalising via singular value decomposition, this approach may enhance interpretability but does not concurrently permit sparsification for more efficient computation and memory, unless in the exceedingly unlikely case that both transforms coincide. Thus, this introduces an interpretability-sparsification tradeoff within isotropic networks. This approach still retains isotropic MLP 'big-$\mathcal{O}$' scalings detailed in *App.* D which may be preferable to standard MLP scalings. Corresponding approaches may also apply to the sketched convolutional form of *App.* E.1 for interpretability. Unlike

sparsification, which may be applied only to every other layer, the 'ICA-representation' of the network can be introduced for every hidden-layer map.

Overall, this demonstrates the flexibility of standard orthogonal prescribed primitives beyond neuroadaptation.

G Notation

- $\mathbb{R}_{\geq 0}$: Non-negative real numbers.
- $\mathbb{Z}_+$: Positive integers.
- $\vec{a}$: Vectors. $\mathbf{A}$: Matrices.
- $\circ$: Function composition.
- $\odot$: Hadamard, or elementwise, product.
- $\|\vec{x}\|_2$: Euclidean norm of $\vec{x}$.
- $\hat{x}$: Unit-normalised vector $\hat{x} = \vec{x}/\|\vec{x}\|_2$.
- $\hat{e}_i$: ith standard basis vector.
- δ_{ij}: Kronecker delta.
- I_n or I: Identity matrix. Rectangular identity in the case of $\mathbf{\Lambda} = \mathbf{\Lambda} \odot \mathbf{I}$.
- $\mathrm{diag}(\mathbf{A})$: Vector formed from the diagonal entries of matrix $\mathbf{A}$.
- $\mathrm{diag}^\dagger(\vec{v})$: Diagonal matrix whose diagonal entries are the components of $\vec{v}$.
- $\mathbb{P}$: Probability measure.
- $\mathcal{O}(\cdot)$: Big-$\mathcal{O}$ notation.
- $\mathcal{G}$: Group and $g \in \mathcal{G}$: an element of that group.
- $\varrho^{(1)}(g)$, $\varrho^{(2)}(g)$, $\varrho\,(g)$: Different group representations.
- $\mathrm{O}(n)$: Orthogonal group in dimension n.
- $\mathrm{GL}_n(\mathbb{R})$: General linear group of invertible $n \times n$ real matrices.
- S_n: Permutation group on n elements.
- $\mathbf{P}_\pi$: Matrix representation of permutation π.
- $P = \{\varrho(\pi) \mid \pi \in S_n\}$: Set of all permutations in the group in a chosen matrix representation.
- $O = \{\varrho(Q) \mid Q \in \mathrm{O}(n)\}$: Set of all elements in the orthogonal group in a chosen matrix representation.
- $[\mathbf{R}, \mathbf{f}]$: Commutator $\mathbf{R} \circ \mathbf{f} - \mathbf{f} \circ \mathbf{R}$.
- $\mathcal{F}$: Functional class.
- $\mathcal{F}_d$: Functional class of isotropic networks of depth d.
- f, $\mathbf{f}$: Function, emboldened notation is a vector-valued function.
- $f_{\mathrm{Aff.}}$: Affine map.
- $f_{\mathrm{Iso.}}$: Isotropic activation function.
- σ: Scalar nonlinearity used to construct an isotropic activation, with alternative formulations $\tilde{\sigma}$, $\bar{\sigma}$ defined by *Eqns.* 5.
- $J_{\mathbf{f}}$: Jacobian of function $\mathbf{f}$.
- $\mathbf{N}(\vec{x})$: Parameterless L_2 normalisation map, $\mathbf{N}(\vec{x}) = \vec{x}/\|\vec{x}\|_2$.
- LN: LayerNorm.
- BN: BatchNorm.
- $\mathbf{\Lambda}$: Diagonal or rectangular diagonal matrix of singular values.
- ϑ: Singular-value threshold.
- Ξ: Desired number of scaffold neurons to maintain below threshold.
- ω: Current number of scaffold neurons, defined by singular values below ϑ.
- o: Intrinsic length parameter introduced for neuroadaptation.
- b_+: Bias component associated with an appended neuron/channel
- $b_\times$: Bias component associated with a deleted neuron/channel
- $\vec{W}_+$: Appended downstream weight column associated with a scaffold neuron/channel.
- $\vec{W}_\times$: Deleted downstream weight column associated with a neuron/channel.
- $\vec{\psi}$: Compensatory vector parameter used to absorb additive terms arising during pruning.
- ϵ: Mean per-example L_2 discrepancy between logits or activations before and after adaptation or sparsification.
- ε: Small positive stabilisation constant in normalisers.
- Σ_0: Mean initial sum of singular values in Experiment One.
- Σ_1: Sum of singular values after neuroadaptation in Experiment One.
- θ: Generic model parameter; also used when counting non-zero parameters.
- $\mathcal{L}$: Loss function. $\nabla_{\vec{a}}\mathcal{L}$: Gradient of the loss with respect to parameter vector $\vec{a}$.
- η: Learning rate.
- $\mathbf{R}^{\odot 2} = \mathbf{R} \odot \mathbf{R}$: Shorthand for elementwise square of an orthogonal matrix. Generally $\mathbf{R}^{\odot n}$ for elementwise power.

H Plain-Language Glossary

This glossary provides straightforward, plain-worded explanations of several terms used in this paper. These are not precise formal or mathematical definitions; they are meant to reduce the burden of niche terminology and provide a directional sense of the intended meaning.

First, we begin with some general terminology, encompassing logical and philosophical meanings:

- **Prescribed**: A central descriptor used in the paper to indicate our approach to symmetry: it is used to indicate that symmetry is not a property that is found as a happenstance when examining activation functions (or other operations), but instead is implemented on purpose with the desire to introduce some property that requires it. It is used in the same sense as 'prescribing' a medicine: an act of intentionally adding something to yield some consequence. This distinguishes it from approaches that find and examine symmetries already incidentally present in a deep learning system.
- **Deduced**: Indicates that some knowledge is derived logically from something known. In particular, we use it to indicate that properties such as symmetry can be determined from known functional forms—like how standard elementwise activation functions can be shown to obey a particular symmetry relation. We use this in contrast to 'prescribed', to indicate methods which take known functions and derive conclusions from them (loosely: prescribed acts in reverse: taking a property and designing a function to obey it).
- **Ontology**: Used in the sense of which structures are considered fundamental to a construction, distinguishing them from what follows from those foundations. It indicates which concepts form the base of a framework, from which others are constructed. It is somewhat connected to the weaker term 'definitional', but carries a stronger philosophical meaning of the most foundational building blocks needed to capture the broader structure. In this case, for the broad topic of neural network systems, classically, there are some units, 'neurons', that interconnect in some manner, the 'network', transforming and intercommunicating information. Here, the neuron-unit is foundational to that description.
- **Emergent**: A property that occurs indirectly as a consequence of another or others; it is not a basic part of the system, but arises from the construction. For example, elementwise individual neurons were originally used as foundational building blocks of neural systems circa the 1950s [54], but since circa the 1990s [27] have been understood to have symmetry properties — and may also be influential in the observed distribution of representations [30, 32]. This is not solely intended as 'strong emergence' where a property is more than its parts.
- **Ontological Reversal**: This swaps something seen as emergent/derivative with something foundational or definitional. In this case, a neuron-unit is classically considered a fundamental constituent of a neural network, from which symmetries may arise; instead, the 'ontological reversal' replaces this with symmetry-fundamentality, from which neurons can then be recovered as an emergent property.

The following terms belong to more mathematical origins, in particular linear algebra (including eigenanalysis) and group theory.

- **Group Theory**: A broad mathematical topic examining the relations among elements under an operation.
- **Symmetry**: Indicates that a specified property remains unchanged or varies with it whilst another is transformed.
- **Group**: A collection of objects interrelated through an operation with specified rules. A classic example is 2-dimensional 90-degree rotations; you can reach 180 degrees by composing several different combinations of 90-degree actions together.
- **Orthogonal**: Classically used as 'perpendicular', such as two vectors perpendicular to one another; this can be achieved through rotations and reflections. An orthogonal *group* is a collection of rotations and reflections. It gets the name 'orthogonal' because, in its standard representation, one property of the columns of its matrix is that they are perpendicular vectors.

- **Group Representation**: The elements of a group are often very abstract, like a rotation. There are many ways to write them down, whilst keeping the group's structure; for example, a rotation represented as a rotation matrix. This is one representation of that group; another may be the imaginary number symbol (or even quaternions) and how it relates to rotations through euler's formula.
- **Group action**: Groups may contain many elements; a group action is taking a single member of the group and applying it to something. For example, applying a 90-degree rotation to a rectangle.
- **Commutation/Commute**: Meaning that the order of application doesn't matter: A then B or B then A.
- **Commutator**: A specific operation related to commutation; in this case, action A then B minus action B then A.
- **Basis independence**: Not tied to any particular basis (coordinate system). For example, mass does not depend on whether we choose south to be north on the Earth, or east to west. In neural network terms, one may consider something basis-independent if the observed quantity does not fundamentally change when all matrices, vectors, etc. are measured in a different basis. This property is not present when using elementwise functions, as their transforms depend on specific directions corresponding to the coordinate system used.
- **Conjugation**: A combination of group actions acting on an object in a particular way. In this case, one could take a group element, like a 2D clockwise 60-degree rotation, then conjugate it with an anticlockwise 120-degree rotation: this yields anticlockwise by 120 degrees, then 60-degrees clockw120 degrees20-degrees clockwise (in this case, they commute, so the anticlockwise and clockwise 120-degree actions cancel). Conjugation is commonly used to change an object's basis.
- **Reparameterisation**: Representing the same thing differently. If one chooses matrices such that both can be rotated in opposing ways that cancel, and they always occur as multiplying one another, then one could choose to show that multiplication as they originally were or with the rotations applied first — they appear different superficially, but achieve the same result.
- **Equivariance**: A property that varies in a predictable way in response to another's variations.
- **Invariance**: A property that does not vary in response to another's variations.
- **Functional Invariance**: When something changes within the system, the system's overall output remains the same.
- **Functional Form**: A general structural family of equations, or more straightforwardly a general equation with placeholders for specific maps. For example, an elementwise activation function describes no single activation function, but a collection of familiar specific instances: ReLU, Leaky-ReLU, sigmoid, etc.
- **Equivalence class**: Given some form of criterion, a collection of objects which are treated as the same. For example, a collection of differently parameterised networks may be treated as an equivalence class if their function is identical for all given inputs.
- **Singular value**: Non-negative real numbers that are square roots of the eigenvalues of a matrix premultiplied by its transpose.
- **Diagonalisation**: The process of reparameterising a matrix (and nearby matrices) so that all non-diagonal elements of the target matrix are zeroed, leaving only a diagonal of non-zero elements.
- **Standard representation**: loosely, a representation of a group acting on its corresponding vector space. Very loosely, this is the most typical way to represent a group element; for example, 3D rotations are represented as 3-by-3 rotation matrices.
- **Trivial representation**: This maps all group elements to a single identity element, since it preserves the group relations. This is used particularly in invariance equations, since all of a group's elements become an identity transformation, so they leave a quantity unchanged, as required by 'invariance'.

The following terms stem from more neuroscientific, general deep learning principles, or newly introduced terminology.

- **Neuroadaptation**: In this paper's narrower sense, this means changing a network's construction by varying the number of neurons. More broadly, it may refer to the network's overall structure, including interconnectedness.
- **Neurogenesis**: Creating new neurons to the system. In this paper, this constitutes enlarging a layer's width by adding new neurons.
- **Neurodegeneration**: Typically, this has a broad meaning referring to neuronal damage, loss-of-function, etc. In this instance, we use it narrowly to mean the removal of neurons from the system, effectively the antonym of neurogenesis.
- **Connectivity pruning**: Removing interconnections between neurons, not the neurons themselves, contrasting with neurodegeneration.
- **Sparsification**: Removing connections in a matrix, particularly to reduce the computational burden. Typically, this removes connections deemed unimportant and therefore approximates function; however, in this paper it arises as a mathematical consequence of reparameterisation rather than direct conditional removal, so it is not an approximation.
- **Individuated Neuron**: A unit in a neural learning system that is inherently distinct and separable from other units in an agreeable way, regardless of the basis used to measure it. In this paper, the elementwise functions distinguish directions (those of the standard, canonical basis) and therefore create a distinction to those directions, to which we assign the classical term 'neuron'. It is unnatural to deviate from those directions as a representing neural unit, since they are inherently special because of network functions.
- **Deindividuated**: Something that may typically be an individual unit having its distinctive separation from other units removed. Neurons are usually single units, but this paper uses mathematics that makes such separation between neurons not fundamental — differing coordinate systems (bases) may disagree on the natural way to separate them into units.
- **Isotropic**: Means the same (iso-) in every direction (-tropic). This paper uses it to mean a function which applies equally in all directions. This does not imply homogeneity; a special centre may still exist even if it is isotropic. This is standard Greek and is frequently used in several scientific disciplines, including machine learning. For example, in astrophysics, a common assumption is that the universe is isotropic on large scales — *approximately* uniform no matter the patch of sky we observe.
- **Anisotropic**: Not (an-) the same (-iso-) in every direction (-tropic). Something that varies with direction.
- **Isotropic Activation Function**: A special form of activation function that applies the same transform regardless of direction. This contrasts with standard elementwise activation functions, which inherently vary with direction (anisotropic activation functions).
- **Primitives**: Basic building blocks of a neural network, although not necessarily ontological parts. For example, activation functions, affine maps, normalisations and initialisations are all largely irreducible components frequently used in composition or more generally to construct a neural network.
- **Neural Learning Systems**: The system of algorithms structured based upon neurons and their interconnectivity — this need not actually be machine learning if hand-chosen implementations occur.
- **Scaffold neuron**: A neuron present in the system which is currently inactive, providing little to no meaningful participation in the computation being performed; at its extreme, it is present structurally only. The aim is that these 'inactive'-like neurons may acquire function as needed through the learning algorithm, specialising into a computational role.
- **Intrinsic length**: A newly introduced quantity, sometimes a trainable parameter; it changes a vector's measured length regardless of its entries. For technical reasons (even in the positive-definite case, it violates absolute homogeneity $\|\alpha\vec{x}\| \neq |\alpha| \, \|\vec{x}\|$ and zero-norm $\left\|\vec{0}\right\| \neq 0$), it does not strictly satisfy the mathematical notion of a length due to several of its axioms.
- **Functional class**: Given a general functional form, or many composed together in a system, the set of all possibilities which are reachable by choosing specific instances of that form. For example, all two-layer perceptrons, with any possible weights and biases.

- **Closure Symmetry**: The weakest of our three proposed symmetry strengths, but encompasses all stronger ones. It states that given a collection of objects, if each is acted on by a specified transformation in a group, the resulting transformed object becomes another entry in that original collection. Hence, the transformation leaves all objects still within their original collection.
- **Probabilistic Symmetry**: A stronger, but intentionally flexible, symmetry condition that indicates the transformation leaves some probability or statistical property unchanged. For example, if vectors of the same length are equally likely to be initialised, then rotating one vector to another remains equally probable, as length has not changed.
- **Algebraic Symmetry**: The strongest of the three proposed symmetry constraints, it indicates that applying the transform to an object in a collection results in the exact same object as you started with (not just another member of the collection).

I Limitations, Assumptions, and Applicability

This section clarifies and discusses the known limitations of the approaches discussed, ranging from diagonalisation and architectural considerations to approximations.

Due to the nature of the diagonalising transforms, this approach requires two parameterised maps that exhibit closure invariance under standard orthogonal transforms on the respective sides. One may use no or any number of interspaced standard orthogonal algebraic equivariant functions to achieve this. In practice, this likely necessitates two densely parameterised layers for the orthogonal transform to act on, and a non-linear function that is orthogonal-equivariant, e.g. isotropic activation functions. Isotropic multilayer perceptrons are straightforward networks that exhibit these properties, with a potential generalisation to convolutional networks described in *App.* E.1 and to other state-of-the-art networks, such as transformers [65] (*App.* E), with more nuanced relations. Thus, it seems possible to broaden the architectural constraints beyond MLP networks, but further work is needed to determine the exact implementation details. Some residual networks may be redesigned to exhibit this property, but diagonalisation introduces complications, as all residually connected layers must be transformable simultaneously. Nevertheless, MLP architectures remain pertinent in applications with unknown inductive biases, inhibiting the specialisation of the architecture class. They are also widely used as backbone subcomponents in larger networks, including transformers, where such neuroadaptive techniques can be applied when two or more such layers occur sequentially.

Another consideration is the orthogonal-prescribed primitives in an MLP, and whether it retains the expressibility of typical construction with permutation-prescription. The adaptive topology construction does not depend on any single isotropic activation function, but on the general form. Furthermore, the competitiveness of this form cannot be ascertained from any single function or application, since it represents a broad functional form whose inherent performance relative to elementwise formulae is currently unknown. This is especially evident in the incomparability of even superficially analogous functions like isotropic-tanh and standard-tanh in the experiments of *App.* K.2 showing markedly different spectra-learning dynamics. Moreover, the chosen isotropic-tanh is merely a superficial analogue of standard tanh (itself not typically optimal), intended as a provisional placeholder that saturates below 1 to prevent the need for normalisations that introduce confounders. However, because neuroadaptation depends only on isotropic forms, the verification empirics did not require the function to be optimal; isotropic-tanh was sufficient to confirm the underlying mechanisms. Nevertheless, the results of *App.* K.6 are encouraging, showing that despite this placeholder, it typically outperforms standard-tanh for deeper and wider MLP architectures on more challenging visual-classification datasets.

Similarly, no *general* universal approximation theorem (UAT) exists for this form, which does not imply its absence either. Despite this, the pragmatic usage of elementwise formulae proceeded for the intermediate 31 years between Rosenblatt's canonical work [54] and Cybenko's first universal approximation theorem [53], alongside their underappreciated specificity to particular architectures. Thus, this unknown is not immediately discouraging; additionally, precedent exists for incidental normed nonlinearities [66, 32] with motivation outside a symmetry-prescription formalism (not to be confused with radial-basis functions [67]). Furthermore, a specific-function, 2-layer UAT has been demonstrated for the 'step-ReLU' radial function [38], which, under our terminology, is an instance of an isotropic function. Regardless, in the worst-case scenario, the newly enabled behaviours may,

in several circumstances, outweigh mild accuracy suboptimality. To optimise for competitiveness or production, we motivate a substantial search over the class of isotropic functions; this remains a limitation for future research, given that the underlying mechanisms are now substantiated.

This also relates to the weight-parameterisations and redundancies therein. Under the standard permutation-prescription, the unique weight-space is $\mathbb{R}^{n\times m}/S_n$ in shorthand, with a generic group size of $|S_n| = n!$ multiplying the functionally equivalent parameterisations. Further reduction occurs for hyperoctahedral-prescribed activation functions to $\mathbb{R}^{n\times m}/B_n$, in shorthand, with group size $|B_n| = 2^n n!$ multiplying unique parameterisations[6]. Whether this retains expressibility, given increasingly redundant parameterisations, remains unclear. Parameter count and function-class expressivity are fundamentally different quantities. Orthogonal-prescribed multilayer perceptrons further reduce the number of effective, unique parameters, as they are a supergroup, $S_n \subset B_n \subset \mathrm{O}(n)$, yielding a continuous orbit of reparameterisations. Pursuing this to larger supergroups eventually yields $\mathrm{GL}_n(\mathbb{R})$, where the induced 'linear activation functions' collapse expressibility to linear regression; therefore, expressibility is compromised in the extreme case.

Whether these degeneracies and reduced unique parameterisations are harmful or helpful for expressibility is presently unclear. Such considerations may, reductionistically, come down to whether better minima emerge along what would be the orthogonal orbit, but when using activations described maximally by typical elementwise subgroups, e.g. S_n, B_n, etc. With these, a potentially *usefully* nuanced loss structure arises because of the absence of continuous redundancy, whereas such minima wouldn't arise for the flattened-loss orbits induced by orthogonal-group prescriptions. Conversely, such local minima could be troublesome for optimisation if unfavourable, whereas a flattened orbit may offer better learning efficiency by avoiding them, since it has a zero gradient in the orbit direction, reducing exploration of the continuous orbit-flattened loss. This appears highly specific to the particularities of the seemingly non-comparable, non-linear relationship of the two functional forms. Some results do suggest a parameter-symmetry dependence on training phenomena [68].

Forming a canonical parameterisation may be more intuitive. We propose the following directionally, assuming that 0 singular-valued neurons are pruned and that positive values are distinct, which is likely after training. An example is the partial-diagonal representations for all but the final layer $\mathbf{R}\mathbf{\Lambda} \in \mathbb{R}^{n\times m}$, where $\mathbf{R}$ has degrees of freedom $0.5n(n-1)$ from its Lie-algebra parameterisation, and magnitude-ordered singular values $\mathbf{\Lambda}$ with $\min(n, m)$ degrees of freedom — these can also be transformed to have $\det \mathbf{R} = +1$. However, a sign-flip redundancy remains in $\mathbf{R}$ despite the positive determinant, and subspace redundancy remains when $m < n$. Further conventions can be explored to reduce this ambiguity. A benefit is that, if a network can be put into a unique canonical form, we could explore an intuitive sense of 'model-distance' — though this may still be possible despite the subspace and column-signed ambiguities, by taking such an inter-model quasi-distance in the Lie-Algebra coordinates (given you restrict them to a non-repeating domain).

A further limitation is that neurodegeneration is, in general, unavoidably approximate in its function invariance. This is because, unlike neurogenesis, one cannot *generally* embed the original parameters, $\mathrm{rank}(\mathbf{W}') \leq \mathrm{rank}(\mathbf{W})$ — non-zero singular values are removed. However, this motivated several compensatory parameters, the intrinsic length and psi-vector, which were shown (*Tab.* 1 and *App.* K.1). Using $\vec{\psi}$ induces an ordering from final-to-first-layer neuroadaptation, ensuring psi-vector compensation propagates to only one subsequent layer. For the same reason, this $\vec{\psi}$-parameter should decay before the next neuroadaptation, which may also help mitigate non-linear instability in its action (although baseline percentages of *Tabs.* 4, 6 and 8 suggest this is inconsequential as they remain comparable).

Furthermore, with a saturating isotropic activation function, or with the discussed normalisation, activation magnitudes cannot grow to compensate for a decaying singular value — thus reliably implying that such singular values have little impact on overall model functionality. Hence, despite the inherent limitations and function invariance during neurodegeneration, the methodology presented does take several steps to mitigate this function drift — confirmed then empirically in *Tab.* 1 results.

Similarly, the results of *App.* A.3 implies that neurogenesis initialisation should not be zero in $\vec{W}^+$ due to inducing an optimisation fixed-point, and $b^+ = 0$ is a necessity to ensure true function invariance,

[6]These should not be confused with the two-sided closures B_n, B_m, or S_n, S_m or $O(n)$, $O(m)$ also applicable, but would reduce parameterisation for every other layer, instead one-side closures are detailed which are relevant for *every* layer except the output.

as otherwise the subsequent biases would need a non-linear correction. This produces an explicit symmetry break at initialisation, biasing the network along a particular optimisation trajectory. As discussed, these explicit symmetry-breaking initialisations can be explored to achieve various effects. Neither is explicit symmetry breaking inherently problematic, as evidenced by the many initialisers in standard use.

It is also essential to clarify that several claims hold only asymptotically; for example, the 50% sparsification figure holds only in the infinite-width and depth limit, while the finite equations are highly architecture-specific to constant-width isotropic multilayer perceptrons. Considerations also pertain to isotropic nested functional classes, which apply only to equal- or monotonically increasing-width isotropic MLPs in the regime where the final activation function becomes 'saturated' (constant in its non-linear term). Nevertheless, several isotropic functions, including isotropic-tanh, do display such saturating behaviour, particularly when considering the positive-definite intrinsic length. Further considerable sparsification was demonstrated, *App.* K.5, for finite width and depth architectures, consistent with *Eqns.* 62 and 63.

Other considerations, such as hyperparameter optimisation, SVD-scheduling, initialisations, isotropic function-searches and thresholding alternatives, could be explored if one wished to demonstrate competitive performance beyond validating the theoretical claims prioritised in this paper. Importantly, the current neuroadaptation approach was shown to add negligible compute time compared to standard training and test-set evaluation, showing that the method's practical cost is dominated by underlying model training rather than the adaptation procedure itself. Thus, one may also increase the frequency of neuroadaptation beyond once per epoch with little ramification on compute time, and alternative variations should be mindful of efficiency.

Further limitations regarding diagonalisation approximations for rank-2 quantities, as used in existing optimisers and normalisers, can also affect invariance in the optimisation trajectory and forward-pass inference, respectively. However, they can be approximately transformed to explicitly compensate; moreover, the concern is mitigated by the likely rotationally degenerate parameterisations being explored regardless through training, with little impact observed thus far on such quantities.

Finally, the empirical component is intended as a mechanism verification rather than a competitive or SOTA-scale implementation. This reflects the paper's primarily theoretical-conceptual contribution: general isotropic mechanisms for structural invariance that support a broader symmetry-prescriptive ontology. Building on this foundation, orthogonal-prescribed primitives were used to admit orthogonal reparameterisations, enabling behaviours including basis independence, SVD-based diagonalisation, sparsification and function-preserving neuroadaptation. These analytical results therefore require empirical verification first; the contribution lies primarily in these novel attributes afforded by a symmetry-prescriptive ontology, rather than in any single optimised experimental procedure. Alongside the restriction to multilayer perceptrons, limiting SOTA possibilities.

Accordingly, the experiments focus on invariance, adaptation dynamics, and computational overhead as the most direct, meaningful tests of the proposed mechanism. The experiments are therefore not presented as evidence of competitiveness but as controlled, minimalist tests of whether the proposed symmetry mechanism behaves as predicted, while minimising confounders to the mechanism. Large-scale competitive benchmarks would introduce architecture choice, activation-function optimality, optimiser interactions, normalisation, data augmentation, and more as confounders; hence, these minimal mechanism-verification experiments were preferred. Moreover, comparing elementwise neuroadaptive approaches to our isotropic formulation introduces significant confounders again, stemming from the markedly different learning dynamics expressed by the differing functional forms. One cannot determine whether differences result from the neuroadaptive method or the underlying functions. This cannot be rectified by reformulating prior techniques onto isotropic primitives or the proposed ones onto elementwise forms, as each typically [22, 23, 25, 26] presupposes and is dependent on its respective form.

Architecturally, one may consider whether elementwise functions have circularly benefited current model constructions. Transformer or convolutional systems may themselves be predicated on primitive choices. Thus, given an altered symmetry-prescribed foundation, unknown selection confounders may exist in more elaborate constructions, encouraging the straightforward verification on multilayer perceptrons. Overall, results with placeholder activation functions should therefore be interpreted as verification of the mechanism, not as assessments of the broader functional class enabled by symmetry-prescribed primitives. Overall, establishing novel mechanistic possibilities

upfront, before refining every component for competitive optimality, is preferable and enables testing of the paper's core claims. However, substantial future work is required to determine the exact pipeline optimisation, hyperparameter search and activation function selection needed for competitive SOTA comparisons.

J Related Works

The works of 'Net2Net' [22], particularly 'Net2WiderNet', and 'Network Morphism' [24], display some relation to this work in terms of neurogenesis considerations. These works effectively propose an approach to embed a narrower network architecture within a wider one, while preserving function. The observation in the Net2WiderNet work is that for an elementwise activation function, one may arbitrarily duplicate an existing column vector to widen a layer's weight matrix, shown in *Eqn.* 57 with $\vec{W}_{(1)} \in \text{column}\left(\mathbf{W}_{(1)}\right)$, provided that the corresponding outgoing weights in the subsequent layer are divided among the duplicated units so that their sum equals the original outgoing weight, shown on the right of *Eqn.* 57 with $\vec{C}_{\text{duplicity}}$ being a vector correcting for duplicity. The appended vectors are functions of their corresponding weights: $\vec{W}^{+}_{(i)}\left(\mathbf{W}_{(i)}\right)$.

$$\mathbf{W}'_{(1)} = \left(\mathbf{W}_{(1)}, \vec{W}^{+}_{(1)}\left(\mathbf{W}_{(1)}\right)\right) \qquad \mathbf{W}'_{(2)} = \vec{C}_{\text{duplicity}} \odot \begin{pmatrix} \mathbf{W}_{(2)} \\ \vec{W}^{+}_{(2)}\left(\mathbf{W}_{(2)}\right) \end{pmatrix} \tag{57}$$

This is arguably a form of neurogenesis for elementwise activations that increases a layer's neuron count, but is otherwise distinct from the singular-value approach enabled by isotropic formulations, discussed in this paper. As a result, embeddings differ considerably; for example, the SVD approach appends a zero column to the singular values and then makes an unconstrained choice in the corresponding row of the subsequent layer. This is achieved only by exploiting the reparameterisation symmetry prescriptively introduced into primitives for this purpose. Additionally, neurodegeneration is not covered by the Net2Net formulations, and their neurogenesis requires some function-invariance noise to break the optimisation symmetry and drive neural differentiation — this noise results in a small error in function preservation. Our approach using isotropic primitives does not require this function-invariance breaking noise: the explicit symmetry breaking introduced in this paper is qualitatively different, preserving the function exactly via null-like reparameterisations whilst desirably breaking the optimiser's symmetry. It would appear that near-zero noisy initialisations may also be possible within the Net2WiderNet framework. Network morphism addresses a problem similar to Net2Net's, but within a more general framework using non-identity factorisations.

Overall, although they share the aim of increasing a network's width, these approaches differ significantly from this paper's in how they achieve function invariance. Further, the results of *App.* K and Bird [32] suggests an incomparability of functional forms even for superficially similar non-linear terms, with Net2WiderNet contingent on elementwise formulae, and this work is centred on symmetry-led isotropic formulae. However, both formulations demonstrate function-invariance under neurogenesis.

Additionally, some techniques from Net2DeeperNet and Network Morphism may be combined with the present work by parameterising the isotropy $\mathbf{f}_*\left(\vec{x};\alpha\right) = \alpha\vec{x} + (1-\alpha)\mathbf{f}\left(\vec{x}\right)$, $\alpha \in [0,1]$ in order to insert new layers. Alternatively, one may exploit the local linearity about the origin $\mathbf{J}_{\mathbf{f}}|_{\vec{0}} \propto \mathrm{I}_{n\times n}$ of isotropic activation functions to similar effect, given a normalisation that guarantees the magnitudes will not enter a highly non-linear region of the activation function. Thus, overlap could be established through Net2DeeperNet.

Rigging-The-Lottery (RigL) [20] more clearly draws a distinction with the present work. Motivationally, RigL responds to the Lottery-Ticket hypothesis [69]. This hypothesis posits that a sparse network found through pruning can match or improve the original network by training that sparse architecture alone from its original initialisation; RigL demonstrates the performance of a training-time sparsely adaptive network. Thus, Rigging-the-Lottery focuses on training networks with sparse representations from the start and progressively performing further connectivity pruning or re-enabling connectivity throughout learning. These actions are based on criteria such as comparison to dense-network gradients for growth and weight-magnitude-based pruning. Thus, this is an example of adaptable connectivity rather than neuroadaptability introduced in this paper; the dense network's

width remains static. This connectivity pruning does change the directionality of the weight-matrix columns, unlike a singular-value thresholding-of-importance approach, which may be undesirable. With such a singular-value approach, these weight columns have a single non-zero entry; thus, decay or removal does not affect directionality, while it also makes connectivity pruning equivalent to full neurodegeneration. Furthermore, this paper's diagonalised representation yields a significant $1 - \min\left(m^{-1}, n^{-1}\right)$ connectivity sparsity for a single weight matrix with an identically invariant function, which experiments could explore in terms of the lottery-ticket hypothesis.

Together with the Single-Shot Network Pruning (SNIP) [19] (also a connectivity-pruning technique), these papers also indicate an alternative thresholding approach, namely loss sensitivity, which could be similarly explored for this paper's neurodegeneration pipeline — and is directionally discussed in *App.* C.

Independently, neuron merging [26] provides a useful comparison on neurodegeneration. This approach uses a weight 'factorisation/decomposition', which retrospectively may be considered to leverage a scaled-permutation-described network. It does this for neurodegeneration by attempting to compensate for functional loss by merging the removed neuron's contribution into a retained, similar neuron. Together with a strong ReLU constraint, it effectively uses an (unstated) positive-scaled permutation (positive-monomial) equivariance exhibited by ReLU [29] to propagate such scalings through the activation function.

The decomposition leverages the interdependence among the weight column vectors to reexpress the removed (individuated) neurons as a scaling of a single remaining column vector (not a general linear combination), using cosine similarity for comparability. However, it is important to consider that this is not an algebraic weight-factorisation like SVD; it is an approximate nearest-retained-neuron compensation rule that presupposes column-wise redundancy for good approximation, distinct from a spectral ranking of function contributions. Thus, a strong approximation enters earlier than the neurodegenerative approximation generally requires, especially compared to the present isotropic-SVD approach.

We argue that the singular-value structure also provides a clearer proxy for the importance of (diagonalised-)neurons than columnwise comparisons, as the former, combined with bounded-norm activations, indicates the functional impact of neurodegeneration. This contrasts with cosine similarity and the separate pruning-criterion heuristics required by the individuated-neuron constraint, which only roughly indicate a neuron's functional redundancy among existing neurons.

The isotropic constraint in this work is also a broad functional-form constraint rather than a specific-instance constraint like elementwise-ReLU in the neuron-merging work. However, this may be generalised retrospectively to positive-monomial-prescribed primitives for this paper, with ReLU as the practical and commonplace example. Overall, this approach has similarities but is explicitly limited to the elementwise-ReLU case and is restrictive in that removed weight columns must be approximately redundant. Thus requiring near-collinearity with *a* preserved column vector. However, viewed retrospectively and alongside Net2WiderNet's work on neurogenesis, one could argue that this combination enables a form of neuroadaptability in networks described by scaled permutation equivariance (ReLU multilayer perceptrons).

The works of "Elastic Weight Consolidation" [51] (EWC), "Progressive Neural Networks" [23] (PNN) and "Dynamically Expandable Networks" [25] (DEN) can also be contextualised in reference to the presented approach. The former is motivated by mammalian dendritic spine preservation during learning of subsequent tasks, this is emulated by 'anchoring' parameters to a low-loss region for the previous task through use of a quadratic loss term multiplied by the diagonal of the Fisher matrix — analogous to a physical elastic-spring system with parameter-wise varying stiffness coefficients. Principally, this approach addresses this different 'catastrophic forgetting problem' [70–72], which is not the focus of our work. However, the EWC approach could be generalised to isotropic networks with relative ease, since the underlying Fisher-quadratic term treatment remains applicable, but it is basis-dependent, requiring consideration and at least appropriate transforms. The diagonal Fisher approximation prevents the latter exactly, but may be approximated via *App.* A's approximate rank-2 basis-change, or transforming the full Fisher prior to diagonal approximation, or a full recomputation of approximate Fisher diagonal after a basis-change. Progressive neural networks approach continual learning by adding subnetworks for new tasks (a neurogenesis-like approach) while freezing already converged parameters. Lateral 'adapters' are used to permit transfer learning from prior trained subnetworks to the newly appended ones if useful. This differs considerably from the isotropic

approach, as PNNs presuppose elementwise independence of the activations when preserving function. Isotropy would re-couple these frozen columns through the non-linearity, as affine layers are no longer the only neuron-mixing operations present; thus, function-independence is infeasible without separately reapplying isotropic non-linearities with blockwise-orthogonal equivariance, preventing full SVD diagonalisations. The blockwise isotropic application also reduces to odd-symmetric elementwise activation functions in a single-column-width case.

Dynamically expandable networks consider these various approaches and generalise from them: sparse connectivity through regularisation; preserving existing network functionality through selective retraining and frozen parameters alongside an EWC-like regularisation term (an L_2 rather than Fisher-scaled quadratic term); adding a variable number of neurons based on a loss-based criterion; or split-duplication where drift exceeds a threshold. Together, this ensemble of actions produces a dynamic programme for neurogenesis, mitigating perceived oversights in prior approaches. Similar considerations regarding presupposition of neuron-independence and functionality freezing prevent a naive isotropic generalisation. Thus, isotropic-adaptive and dynamically expandable networks are contingent on their respective activation forms for intended function, with neurogenesis mechanisms that trigger on considerably different metrics. Thresholded split-duplication cannot be performed on diagonalised neurons, because the SVD likely produces different orthogonal transforms as training evolves, making differently timed diagonalised neurons incomparable. This largely stems from neuron indistinguishability and basis freedom. However, group sparsity (not a group-theoretic approach, despite $g \in \mathcal{G}$ notation; more set-theoretic) is drastically simpler in the diagonalised isotropic picture, as the one-to-one connectivity makes classic L_1 (or L_2) per singular value equivalent to the neuron-wise group sparsity used in Yoon et al. [25] (and nuclear norm) — thus, is comparable to the decay we used.

Overall, comparing these continual learning mechanisms directly requires isolating each mechanism from significant confounders, but fundamental differences prevent this. Substituting isotropic forms into either of the latter methods breaks their respective function-preserving mechanisms, similar for Net2Net, Network Morphism and Neuron Merging — while EWC addresses a distinct problem. Each method implicitly relies on elementwise formulae, or specific instances thereof, and/or the independence brought by individuated neurons to restructure the architecture without function interference; however, our reframing removes this requirement and replaces it with isotropic forms. In the contrary case: running each method with its own native activation form avoids this failure, but introduces a significant confound: even superficially similar functions like standard- and isotropic-tanh learn in inherently disparate ways, even in static networks shown by diverse spectral evolutions of *App.* K.2. This activation form confound is inseparable from the continual learning algorithms built upon it. Thus, taken together, a suitable ablation strategy is needed to isolate continual learning algorithms from the activation forms on which they depend, but it is unclear how this can be undertaken whilst faithfully presenting each method within its intended regime. Thus, this discussion provides structural and mechanistic comparisons between the various approaches.

Finally, what we term 'isotropic activation functions' has precedent under the 'radial rescaling activations' [38] term, which should not be conflated with the historical and contemporary prevalence of 'radial basis functions' [67] or $\mathrm{U}(1)^n \rtimes S_n$ (elementwise complex-norm) permutation and independent-phase (unitary) equivariant C-ReLU functions [73, 74] (acting more like complexified elementwise activation functions). Ganev et al. [38] independently [75, 30–32] recognised this form, and analogously its basis-freedom, under a rather different QR-decomposition-based approach. This compression approach leverages the distinct QR decomposition to reduce the width of subsequent layers, by also exploiting the orthogonal reparameterisation. Hence, the network is analytically function invariant to this as well. For a preceding width n, this decomposition effectively finds a transform that embeds the subspace activations can reach under the affine map's image into the first $n + 1$ neurons of the subsequent layer—thus the network is restructured to grow layer-by-layer by at most one neuron. The additional $+1$ arises from considering the bias-term $(\mathbf{W}, \vec{b}) \times (\vec{x}, 1)^T$. Comparable compression remains possible via our distinct SVD-based approach; however, this was strictly not undertaken due to gradient inequivalence and the importance of the buffer of so-called 'scaffold neurons' we wished to retain with small (potentially zero or non-zero) singular values. The derivation of *projected* gradient descent [38] equivalence does not alleviate concerns about scaffold neurons, as projected formulations prevent their update and specialisation. Near-but-not-zero singular values are also not directly exposed through QR decomposition either; they cannot be identified solely from this decomposition. This inhibits our neurodegenerative approach, which requires these.

However, because they have distinct remits, we highlight an important consequence of combining both independent pursuits: QR compression followed by SVD sparsification—these are not mutually exclusive actions.

One can consider an amplified QR-SVD inference-time compression by diagonalising the compressed model. For $m + 1 < n$, weights of $\mathbf{W} \in \mathbb{R}^{n\times m}$ then QR-compression yields a matrix of $\mathbf{W} \in \mathbb{R}^{m+1\times m}$, which SVD-sparsification (on alternating layers) then sparsifies to $\vec{W} \in \mathbb{R}^m$, thus yielding a connectivity reduction of $1 - n^{-1}$ instead of $1 - m/n - 1/n$ or $1 - \min(m^{-1}, n^{-1})$ for QR-compression or SVD-diagonalisation respectively. However, since the $m + 1 < n$ case yields $n^{-1} = \min(m^{-1}, n^{-1})$, this offers no improvement over SVD alone when the criteria for QR compression apply. Intuitively, and simplifying to the biasless case, this is because SVD's rectangular diagonal matrix zeros out rows beyond the square diagonal, which QR compression explicitly removes anyway; hence, the main difference may be the full-matrix footprint when used in combination. However, we note that one could store only the SVD's diagonal vector rather than the full matrix, which reduces the matrix footprint further and more than QR compression. SVD-diagonalisation does not have such criteria either, so it may be applied regardless on alternating layers; yet, QR-compression could meaningfully reduce footprint and computation on the intervening layers left dense by the alternating SVD-diagonalisation procedure.

The authors also present a special-case, bounded-width universal approximation theorem to accompany this result, which they explicitly acknowledge as restricted to the Step-ReLU case. Despite this special-case regime, it nevertheless provides important insights into the expressibility of such forms and potentially broader generalisations.

Overall, alongside distinctly motivated specific cases [66], instances of mutual convergence on this functional form have occurred. Yet across these, distinct considerations, consequences, architectures [66] and behaviours have arisen. These fruitful directions encourage analysis of this form in particular and the broader, symmetry-led primitives and possibilities available, without presupposing neuron-like historical constructions or global model adherence. This need not be limited to activation functions either, but can span normalisers, optimisers, initialisers, convolution, operations and may extend to residuals, attention, and more. It is this symmetry-prescriptive use at the level of individual primitives, extending beyond compositional model-scale or reparameterisation considerations, to induce useful or potentially confounding [32] biases, behaviours and benefits, that this approach and taxonomy emphasise as a potentially productive avenue for generalisation. Such symmetry consequences may arise individually or compositionally (symmetry making and breaking), while the taxonomy may helpfully organise these consequences and associate them with their underlying symmetry roots.

Returning to neuroadaptation, the isotropic approach enables both considerable connectivity pruning through sparsification and neurogenesis, and is complemented by full neurodegeneration. Thus, the three typically separated constructions: connectivity-pruning, neurogenesis and neurodegeneration, are unusually unified in this approach, unlike the discussed works. Prior work recovers only aspects of this full regime, limited to neurogenesis, neurodegeneration or connectivity sparsity, rather than a fully integrated and cohesive approach. Moreover, these prior neuroadaptive works are limited by neuron individuation, identified as a significant hurdle to network adaptation due to the high interconnectivity among individuated neurons, which is not present among diagonalised neurons. This paper's approach to reformulating primitives through orthogonal, equivariant prescriptions to deindividuate neurons enables a broader use of decomposition tools, most notably singular-value decomposition, to construct function-invariant diagonalised representations. This yields a basis for treating neurogenesis, connectivity sparsification, and neurodegeneration within a common framework, rather than as separate heuristic procedures. The latter two become equivalent for the diagonalised neurons. The various thresholdings enable arbitrarily well-approximated function invariances during neuroadaptation by exploiting their symmetry-principled design.

Consequently, direct comparison with elementwise approaches is limited: they operate within the inherited symmetry structure of conventional neuron-indexed networks, whereas the present work alters the primitive symmetry class itself, making adaptive topology a consequence of the network's reparameterisation structure — and incurring considerably different computational approaches, shown in *App.* K.2, further limiting comparison.

K Experimental Details

This section covers all experimental details used in the production of the plots and tables in *Sec.* 3. It also includes additional verification results, such as sparsification invariance and spectra evolution of standard and isotropic-tanh networks of various widths. This paper primarily presents the theoretical developments underpinning such architectures, and the experiments are intended to substantiate those theoretical claims.

Generic hyperparameters were used across all experiments to ensure consistency and that observed effects were attributable to the general theoretical formulation rather than to carefully selected parameter settings. These hyperparameters are as follows: learning-rate $\eta = 1 \times 10^{-3}$, batch-size, $B_s = 48$, orthogonal initialisation, all networks are trained with ADAM-w [76] with L_2 weight decay, $\lambda = 1 \times 10^{-3}$ (otherwise PyTorch [77] defaults were used), additional psi-vector L_2 decay $\lambda_{\vec{psi}} = 0.1$ (due to the fast decay to zero being an important consideration for the compensation mechanism), and elementwise standardisation of the input. We chose these hyperparameters as standard values and kept them fixed across training to ensure maximal comparability. Standardisation ensures that, elementwise, the samples have zero mean and a standard deviation of 1 across the dataset, unless all samples have an element equal to 0, in which case it remains 0. This approach was used across all experiments, and default train-test data splits were utilised ($80:20$ for Caltech256). Caltech256 also had all samples resized to 64×64 pixels across all experiments, which we judged a good balance: large enough to distinguish classes while remaining reasonably minimal. Cross-entropy loss was used across all classification experiments alongside isotropic-tanh, $\mathbf{f}(\vec{x}) = \tanh(\|\vec{x}\|_2)\,\hat{x}$, as the activation function in every case. All experiments report KL-divergence with natural-base logarithms, often termed 'nats'.

For neurogenesis, the appended scaffold neurons were initialised with outgoing weights as $\vec{W}^+ \sim \mathcal{N}(\vec{0}, 0.02 \times \mathbf{I})$, intended to provide a small symmetry-breaking perturbation as needed (*App.* A.3), while being sufficiently small such that this initial biased direction can be potentially overridden by optimisations. Other approaches to non-zero initialisation could have been explored, such as a linear combination of existing columns, but this was considered a stronger inductive bias. The number of independent repeats varied per experiment. All code used to produce the plots is available at `https://github.com/GeorgeBird1/Isotropic-Adaptive-MLPs` to enable reproduction. The code provides base seeds for many runs; however, we ran concurrent instances in tandem, each with distinct seeds to prevent duplicate samples. Experiments were conducted on a single NVIDIA RTX 3080 10GB GPU with 128 GB of DDR3 RAM.

Architecture, number of epochs and dataset varied by experiment to show a breadth of networks in which the theoretical claims can be verified, and epochs varied due to the individual nature of the experiments.

K.1 Experiment One: Function-Invariance to Neuroadaptation

The objective of this experiment was to verify that adapting the width of a trained network can be achieved with minimal functional change, owing to the symmetry-led network construction. In particular, this invariance should hold without subsequent retraining following the architectural adaptation. The experiment explored several different approaches for the compensatory parameters of intrinsic length, o, and $\vec{\psi}$.

For this verification experiment, we used Caltech256 classifiers on an isotropic multilayer perceptron architecture with the structure $[12288, 100, 100, 256]$. We trained each network for 50 epochs across 20 independent repeats per configuration. This section also provides identical experimental results but for CIFAR-10 and CIFAR-100, or network size $[3072, 100, 100, 10]$ and $[3072, 100, 100, 100]$, respectively.

The columnwise configurations are as follows:

1. "$o_{\text{none}}, \vec{\psi}_{\text{decay}}$" consists of networks with no intrinsic length parameters, but they do include a trainable psi-vector compensatory parameter, which does have an L_2 decay term such that it tends to the zero vector. The stated algebra is used to transform psi under neuroadaptation.

2. "o_{constant}, $\vec{\psi}_{\text{decay}}$" consists of networks with a constant, non-trainable, intrinsic length parameter, and they include a trainable psi-vector compensatory parameter, decaying to the zero vector. The stated algebra is used to transform intrinsic length and psi under neuroadaptation.
3. "$o_{\text{trainable}}$, $\vec{\psi}_{\text{decay}}$" consists of networks with a trainable intrinsic length parameter and a trainable psi-vector compensatory parameter that decays to the zero vector. The stated algebra is used to transform intrinsic length and psi under neuroadaptation.
4. "$o_{\text{trainable}}$, $\vec{\psi}_{\text{none}}$" consists of networks with a trainable intrinsic length parameter and no psi-vector compensatory parameter. The stated algebra is used to transform the intrinsic length.
5. "$o_{\text{trainable}}$, $\vec{\psi}_{\text{constant}}$" consists of networks with a trainable intrinsic length parameter but a constant psi-vector compensatory parameter. The stated algebra is used to transform intrinsic length and psi under neuroadaptation.
6. "$o_{\text{trainable}}$, $\vec{\psi}_{\text{trainable}}$" consists of networks with a trainable intrinsic length parameter and a trainable psi-vector compensatory parameter that does not decay to zero. The stated algebra is used to transform intrinsic length and psi under neuroadaptation.
7. "$o_{\text{trainable}}$, $\vec{b}_{\text{adjusted}}$" consists of networks with a trainable intrinsic length parameter and no psi-vector compensatory parameter. The stated algebra is used to transform intrinsic length, and it uses the $\vec{b}_2' = \vec{b}_2 + \mathbb{E}\left[\bar{\sigma}\left(\cdot\right)\right] \vec{W}_{\times} b_{\times}$ to transform the bias vector.

Exploring all permutations of approaches to the psi-vector and intrinsic length would have required extensive compute, without yielding further meaningful insight or interpretation. Therefore, comparing intrinsic length approaches against a trainable psi-vector and comparing psi-vector approaches against a trainable intrinsic length provided the most informative combinations for practical implementation of this neuroadaptive methodology.

These trained networks then undergo adaptation of their $\mathbb{R}^{100} \to \mathbb{R}^{100}$ affine map, *in k sequential steps* increasing or decreasing the width to the stated quantity, such as $+75$, yielding a map $\mathbb{R}^{100} \to \mathbb{R}^{175}$ or -75 yielding a map $\mathbb{R}^{100} \to \mathbb{R}^{25}$ — no retraining is then undertaken, such that the invariance of the adaptation alone can be analysed. The row stating 0 refers to just a layerwise partial diagonalisation, where the map retains the $\mathbb{R}^{100} \to \mathbb{R}^{100}$ dimensionality but is reparameterised accordingly.

From this, we compute various metrics to compare before-and-after distributions of the final-layer activations. These are then averaged over all samples in the dataset, and then the final quoted mean and standard error were computed repeat-wise. The metrics utilised were relative-change (*Eqn.* 58), KL-divergence (*Eqn.* 59), sum of singular values (*Eqn.* 60) and change in test-set accuracy. Relative change applies to the logits before softmax, $\vec{z}$, while KL-divergence is computed from elements $p^{(i)}$ of the vector after softmax, $\vec{p}$.

$$\epsilon_{\text{rel}} = \mathbb{E}_{(x,y)\sim D_{\text{test}}} \frac{\left\|\vec{z}_{\text{post}} - \vec{z}_{\text{pre}}\right\|_2}{\left\|\vec{z}_{\text{pre}}\right\|_2} \quad \times 100[\%] \tag{58}$$

$$D_{KL} = \mathbb{E}_{(x,y)\sim D_{\text{test}}} \sum_i p_{\text{pre}}^{(i)} \ln\left(\frac{p_{\text{pre}}^{(i)}}{p_{\text{post}}^{(i)}}\right) \quad [\text{nats}] \tag{59}$$

$$\Sigma = \sum_i^{\min(m,n)} \mathbf{\Lambda}_{ii} \quad \text{for} \quad \mathbf{U}, \mathbf{\Lambda}, \mathbf{V} = \text{SVD}\left(\mathbf{W}\right) \tag{60}$$

These metrics were chosen as informative, robust measures of function invariance during neuroadaptation. KL-divergence indicates the overall distribution shift after adaptation compared to before, whilst relative-change indicates how much the logits shift as a percentage relative to their original scale. Singular values are non-negative, and their sums thus quantitatively measure how neuroadaptation changes the underlying affine transform's linear scaling through the weights. Test-set accuracy, also recorded, is likely the least informative, as more substantial shifts in logits may be required to cross a 'decision boundary', but it still indicates model performance. This overlaps somewhat

with relative-change as a coarser metric: sufficiently small ϵ_{rel} results in a negligible change in test accuracy, which may vary more discretely in steps.

Baseline accuracies indicate how the various approaches to psi-vector and intrinsic-length compensatory parameters affected training over the 50 epochs preceding neuroadaptation. They show only a marginal effect, suggesting their role is largely limited to the neuroadaptive compensations for which they were derived, rather than broader utility as training parameters.

This section now presents the full tabulated results of experiment one, across all metrics and datasets (CIFAR-10, CIFAR-100, Caltech256) tested.

Across the results (*Tab.* 3 to 6), the broad pattern is consistent: the network is exactly function invariant under neurogenesis and diagonalisation/sparsification, whilst invariance is well approximated under modest neurodegeneration, which inevitably degrades with more substantial pruning due to the significant reduction in singular values available to manipulate the data usefully. Despite this, accuracy, particularly for CIFAR-10 (*Tab.* 6), remains more consistent.

Table 3: Experiment One (Caltech-256): singular-value structure and relative function change under neuroadaptation at layer $L = 1$ of the $[12288, 100, 100, 256]$ classifier. Results are mean $\pm$ standard error over 20 independent repeats. Row 0 applies diagonalisation only; $+k$ subsequently applies k neurogenesis operations in a single step and $-k$ applies k neurodegeneration operations in a single step. Σ_0 and Σ_1 denote the sums of singular values before and after neuroadaptation, respectively. Singular-value standard errors are omitted for legibility; the largest singular-value standard error is 0.38. ϵ_{rel} is the *Eqn.* 58's metric, bold indicates the smallest value within each neuroadaptation condition.

Neuroadaptation	o_{none}, $\vec{\psi}_{\text{decay}}$ $\Sigma_0 = 240$		o_{constant}, $\vec{\psi}_{\text{decay}}$ $\Sigma_0 = 240$		$o_{\text{trainable}}$, $\vec{\psi}_{\text{decay}}$ $\Sigma_0 = 241$		$o_{\text{trainable}}$, $\vec{\psi}_{\text{none}}$ $\Sigma_0 = 240$		$o_{\text{trainable}}$, $\vec{\psi}_{\text{constant}}$ $\Sigma_0 = 237$		$o_{\text{trainable}}$, $\vec{\psi}_{\text{trainable}}$ $\Sigma_0 = 237$		$o_{\text{trainable}}$, $\vec{b}_{\text{adjusted}}$ $\Sigma_0 = 240$	
	Σ_1	ϵ_{rel} (%)	Σ_1	ϵ_{rel} (%)	Σ_1	ϵ_{rel} (%)	Σ_1	ϵ_{rel} (%)	Σ_1	ϵ_{rel} (%)	Σ_1	ϵ_{rel} (%)	Σ_1	ϵ_{rel} (%)
$+75$	240	$(8.4\pm0.2)\times10^{-4}$	240	$(8.5\pm0.2)\times10^{-4}$	241	$\mathbf{(8.2\pm0.2)\times10^{-4}}$	240	$(8.6\pm0.1)\times10^{-4}$	237	$(8.5\pm0.2)\times10^{-4}$	237	$(8.3\pm0.2)\times10^{-4}$	240	$(8.3\pm0.1)\times10^{-4}$
$+50$	240	$(8.4\pm0.2)\times10^{-4}$	240	$(8.5\pm0.2)\times10^{-4}$	241	$\mathbf{(8.2\pm0.2)\times10^{-4}}$	240	$(8.6\pm0.1)\times10^{-4}$	237	$(8.5\pm0.2)\times10^{-4}$	237	$(8.3\pm0.2)\times10^{-4}$	240	$(8.3\pm0.1)\times10^{-4}$
$+25$	240	$(8.4\pm0.2)\times10^{-4}$	240	$(8.5\pm0.2)\times10^{-4}$	241	$\mathbf{(8.2\pm0.2)\times10^{-4}}$	240	$(8.6\pm0.1)\times10^{-4}$	237	$(8.5\pm0.2)\times10^{-4}$	237	$(8.3\pm0.2)\times10^{-4}$	240	$(8.3\pm0.1)\times10^{-4}$
$+5$	240	$(8.4\pm0.2)\times10^{-4}$	240	$(8.5\pm0.2)\times10^{-4}$	241	$\mathbf{(8.2\pm0.2)\times10^{-4}}$	240	$(8.6\pm0.1)\times10^{-4}$	237	$(8.5\pm0.2)\times10^{-4}$	237	$(8.3\pm0.2)\times10^{-4}$	240	$(8.3\pm0.1)\times10^{-4}$
$+1$	240	$(8.4\pm0.2)\times10^{-4}$	240	$(8.5\pm0.2)\times10^{-4}$	241	$\mathbf{(8.2\pm0.2)\times10^{-4}}$	240	$(8.6\pm0.1)\times10^{-4}$	237	$(8.5\pm0.2)\times10^{-4}$	237	$(8.3\pm0.2)\times10^{-4}$	240	$(8.3\pm0.1)\times10^{-4}$
0	240	$(8.4\pm0.2)\times10^{-4}$	240	$(8.5\pm0.2)\times10^{-4}$	241	$\mathbf{(8.2\pm0.2)\times10^{-4}}$	240	$(8.6\pm0.1)\times10^{-4}$	237	$(8.5\pm0.2)\times10^{-4}$	237	$(8.3\pm0.2)\times10^{-4}$	240	$(8.3\pm0.1)\times10^{-4}$
-1	240	$(2.4\pm0.1)\times10^{-1}$	240	$(2.4\pm0.1)\times10^{-1}$	240	$(2.6\pm0.2)\times10^{-1}$	240	$\mathbf{(2.2\pm0.1)\times10^{-1}}$	237	$(2.7\pm0.1)\times10^{-1}$	236	$(2.6\pm0.1)\times10^{-1}$	240	$(2.3\pm0.2)\times10^{-1}$
-5	236	$(3.5\pm0.2)\times10^{-1}$	236	$(3.7\pm0.2)\times10^{-1}$	236	$(3.8\pm0.3)\times10^{-1}$	236	$\mathbf{(3.4\pm0.2)\times10^{-1}}$	233	$(6.7\pm0.2)\times10^{-1}$	233	$(6.4\pm0.2)\times10^{-1}$	236	$(3.7\pm0.3)\times10^{-1}$
-25	212	$(7.2\pm0.2)\times10^{-1}$	212	$(7.3\pm0.2)\times10^{-1}$	212	$(7.5\pm0.3)\times10^{-1}$	212	$(7.1\pm0.2)\times10^{-1}$	209	$(1.0\pm0.0)\times10^{0}$	209	$(9.9\pm0.2)\times10^{-1}$	212	$\mathbf{(7.1\pm0.3)\times10^{-1}}$
-50	170	$(3.7\pm0.1)\times10^{0}$	170	$(3.6\pm0.1)\times10^{0}$	170	$(3.9\pm0.2)\times10^{0}$	170	$\mathbf{(3.6\pm0.2)\times10^{0}}$	167	$(4.2\pm0.1)\times10^{0}$	167	$(4.1\pm0.2)\times10^{0}$	170	$(3.7\pm0.1)\times10^{0}$
-75	99	$(6.3\pm0.0)\times10^{+1}$	99	$(6.3\pm0.0)\times10^{+1}$	99	$(6.2\pm0.1)\times10^{+1}$	99	$(6.3\pm0.0)\times10^{+1}$	98	$(6.3\pm0.0)\times10^{+1}$	97	$\mathbf{(6.2\pm0.0)\times10^{+1}}$	99	$(6.3\pm0.0)\times10^{+1}$

Table 4: Experiment One (Caltech-256): additional function-change measures for the same neuroadaptation conditions. Results are mean $\pm$ standard error over 20 independent repeats. D_{KL} is provided by *Eqn.* 59 and $\Delta a = a_{\text{post}} - a_{\text{pre}}$ is the change in test accuracy in percentage points; positive values indicate increased accuracy. The baseline row reports the unadapted test accuracy a_0 as a percentage. Bold values indicate the smallest mean D_{KL} and the accuracy change closest to zero within each neuroadaptation condition.

Neuroadaptation	o_{none}, $\vec{\psi}_{\text{decay}}$		o_{constant}, $\vec{\psi}_{\text{decay}}$		$o_{\text{trainable}}$, $\vec{\psi}_{\text{decay}}$		$o_{\text{trainable}}$, $\vec{\psi}_{\text{none}}$		$o_{\text{trainable}}$, $\vec{\psi}_{\text{constant}}$		$o_{\text{trainable}}$, $\vec{\psi}_{\text{trainable}}$		$o_{\text{trainable}}$, $\vec{b}_{\text{adjusted}}$	
	D_{KL}	Δa (pp)	D_{KL}	Δa (pp)	D_{KL}	Δa (pp)	D_{KL}	Δa (pp)	D_{KL}	Δa (pp)	D_{KL}	Δa (pp)	D_{KL}	Δa (pp)
$+75$	$\mathbf{(1.0\pm0.3)\times10^{-9}}$	$\mathbf{0.0\pm0.0}$	$(1.4\pm0.2)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.4\pm0.3)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.2\pm0.3)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.2\pm0.3)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.4\pm0.3)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.5\pm0.3)\times10^{-9}$	$\mathbf{0.0\pm0.0}$
$+50$	$(1.3\pm0.3)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$\mathbf{(9.0\pm3.2)\times10^{-10}}$	$\mathbf{0.0\pm0.0}$	$(1.3\pm0.3)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.3\pm0.3)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.5\pm0.3)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.2\pm0.3)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.5\pm0.2)\times10^{-9}$	$\mathbf{0.0\pm0.0}$
$+25$	$(1.3\pm0.3)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$\mathbf{(1.1\pm0.3)\times10^{-9}}$	$\mathbf{0.0\pm0.0}$	$(1.3\pm0.3)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.3\pm0.3)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.5\pm0.3)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.2\pm0.3)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.5\pm0.2)\times10^{-9}$	$\mathbf{0.0\pm0.0}$
$+5$	$(1.3\pm0.3)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$\mathbf{(1.1\pm0.3)\times10^{-9}}$	$\mathbf{0.0\pm0.0}$	$(1.3\pm0.3)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.3\pm0.3)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.5\pm0.3)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.2\pm0.3)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.5\pm0.2)\times10^{-9}$	$\mathbf{0.0\pm0.0}$
$+1$	$(1.3\pm0.3)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$\mathbf{(1.1\pm0.3)\times10^{-9}}$	$\mathbf{0.0\pm0.0}$	$(1.2\pm0.3)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.3\pm0.3)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.7\pm0.3)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.2\pm0.3)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.5\pm0.2)\times10^{-9}$	$\mathbf{0.0\pm0.0}$
0	$\mathbf{(6.9\pm2.9)\times10^{-10}}$	$\mathbf{0.0\pm0.0}$	$(1.2\pm0.2)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.9\pm0.3)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(8.0\pm3.0)\times10^{-10}$	$\mathbf{0.0\pm0.0}$	$(1.2\pm0.3)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.5\pm0.3)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(9.0\pm3.0)\times10^{-10}$	$\mathbf{0.0\pm0.0}$
-1	$(1.6\pm0.2)\times10^{-5}$	$\mathbf{0.0\pm0.0}$	$(1.6\pm0.2)\times10^{-5}$	$+0.0\pm0.0$	$(2.0\pm0.4)\times10^{-5}$	$+0.0\pm0.0$	$\mathbf{(1.3\pm0.1)\times10^{-5}}$	-0.0 ± 0.0	$(2.0\pm0.2)\times10^{-5}$	-0.0 ± 0.0	$(1.9\pm0.2)\times10^{-5}$	-0.0 ± 0.0	$(1.5\pm0.2)\times10^{-5}$	$+0.0\pm0.0$
-5	$(2.9\pm0.3)\times10^{-5}$	-0.0 ± 0.0	$(3.3\pm0.4)\times10^{-5}$	-0.0 ± 0.0	$(3.7\pm0.7)\times10^{-5}$	$\mathbf{+0.0\pm0.0}$	$\mathbf{(2.8\pm0.5)\times10^{-5}}$	-0.0 ± 0.0	$(1.1\pm0.1)\times10^{-4}$	-0.0 ± 0.0	$(10.0\pm0.6)\times10^{-5}$	-0.0 ± 0.0	$(3.5\pm0.7)\times10^{-5}$	$+0.0\pm0.0$
-25	$(9.9\pm0.5)\times10^{-5}$	-0.0 ± 0.0	$(1.0\pm0.1)\times10^{-4}$	-0.0 ± 0.0	$(1.1\pm0.1)\times10^{-4}$	$\mathbf{-0.0\pm0.0}$	$\mathbf{(9.8\pm0.6)\times10^{-5}}$	-0.0 ± 0.0	$(2.2\pm0.1)\times10^{-4}$	-0.1 ± 0.0	$(2.0\pm0.1)\times10^{-4}$	-0.1 ± 0.0	$(9.8\pm0.9)\times10^{-5}$	-0.0 ± 0.0
-50	$\mathbf{(2.8\pm0.2)\times10^{-3}}$	-0.3 ± 0.0	$(2.8\pm0.2)\times10^{-3}$	-0.3 ± 0.0	$(3.4\pm0.4)\times10^{-3}$	-0.3 ± 0.0	$(2.8\pm0.3)\times10^{-3}$	-0.3 ± 0.0	$(3.8\pm0.3)\times10^{-3}$	$\mathbf{-0.3\pm0.0}$	$(3.6\pm0.4)\times10^{-3}$	-0.3 ± 0.1	$(2.8\pm0.2)\times10^{-3}$	-0.3 ± 0.0
-75	$(9.0\pm0.2)\times10^{-1}$	-11.9 ± 1.0	$(9.0\pm0.1)\times10^{-1}$	-11.2 ± 0.9	$(9.0\pm0.1)\times10^{-1}$	-11.4 ± 1.0	$\mathbf{(8.6\pm0.1)\times10^{-1}}$	$\mathbf{-11.0\pm0.9}$	$(9.7\pm0.1)\times10^{-1}$	-11.8 ± 1.1	$(9.4\pm0.1)\times10^{-1}$	-11.5 ± 1.0	$(8.8\pm0.1)\times10^{-1}$	-11.8 ± 1.0
Baseline	–	20.6 ± 2.2	–	19.4 ± 2.1	–	19.4 ± 2.1	–	19.4 ± 2.1	–	19.7 ± 2.2	–	19.7 ± 2.2	–	20.4 ± 2.2

The Caltech256 neurogenesis KL-divergence is particularly informative, as shown in *Tab.* 4, indicating values on order of 10^{-9} nats, which is a practically negligible change in function, confirming the exact function invariance desired. The comparability of these values to the diagonalisation row may suggest they reflect overall precision rather than a small error associated with neurogenesis. These small errors may reflect amplification of numerical errors by subsequent layers, or may stem from the precision of the SVD decomposition itself. At -50 neurons degenerated, the network shows a more modest change in function while retaining its classification accuracy, suggesting robustness to moderate degeneration even for Caltech256's many classifications. At a 75% reduction in layer width, the invariance drops more drastically, due to the aggressive loss in singular values and their transforms critical to expressivity. However, even with aggressive neurodegeneration, the network continues to perform better than chance, suggesting the remaining learned transforms are still persistently

useful rather than fully compromising utility. Separately, the column-wise comparisons indicate that while intrinsic-length helps a small but inconsistent amount in neurodegeneration, $\vec{\psi}$-corrections are modestly detrimental in this example.

The results (*Tab.* 5 and 6) show these metrics on the CIFAR-10 experimental setup, documenting similar trends but greater indifference to test accuracy in neurodegeneration, likely due to the smaller number of classes and wider margins.

Table 5: Experiment One (CIFAR-10): singular-value structure and relative function change under neuroadaptation at layer $L = 1$ of the $[3072, 100, 100, 10]$ classifier. Results are mean $\pm$ standard error over 20 independent repeats. Row 0 applies diagonalisation only; $+k$ subsequently applies k neurogenesis operations in a single step and $-k$ applies k neurodegeneration operations in a single step. Σ_0 and Σ_1 denote the sums of singular values before and after neuroadaptation, respectively. Singular-value standard errors are omitted for legibility; the largest singular-value standard error is 0.32. ϵ_{rel} is the *Eqn.* 58's metric, bold indicates the smallest value within each neuroadaptation condition.

Neuroadaptation	$o_{\text{none}},\ \vec{\psi}_{\text{decay}}$ $\Sigma_0 = 174$		$o_{\text{constant}},\ \vec{\psi}_{\text{decay}}$ $\Sigma_0 = 174$		$o_{\text{trainable}},\ \vec{\psi}_{\text{decay}}$ $\Sigma_0 = 174$		$o_{\text{trainable}},\ \vec{\psi}_{\text{none}}$ $\Sigma_0 = 174$		$o_{\text{trainable}},\ \vec{\psi}_{\text{constant}}$ $\Sigma_0 = 164$		$o_{\text{trainable}},\ \vec{\psi}_{\text{trainable}}$ $\Sigma_0 = 164$		$o_{\text{trainable}},\ \vec{b}_{\text{adjusted}}$ $\Sigma_0 = 174$	
	Σ_1	ϵ_{rel} (%)	Σ_1	ϵ_{rel} (%)	Σ_1	ϵ_{rel} (%)	Σ_1	ϵ_{rel} (%)	Σ_1	ϵ_{rel} (%)	Σ_1	ϵ_{rel} (%)	Σ_1	ϵ_{rel} (%)
+75	174	$(3.1\pm0.1)\times10^{-3}$	174	$(2.9\pm0.1)\times10^{-3}$	174	$(2.9\pm0.1)\times10^{-3}$	174	$(3.0\pm0.1)\times10^{-3}$	164	$\mathbf{(2.5\pm0.1)\times10^{-3}}$	164	$(2.7\pm0.1)\times10^{-3}$	174	$(3.1\pm0.2)\times10^{-3}$
+50	174	$(3.1\pm0.1)\times10^{-3}$	174	$(2.9\pm0.1)\times10^{-3}$	174	$(2.9\pm0.1)\times10^{-3}$	174	$(3.0\pm0.1)\times10^{-3}$	164	$\mathbf{(2.5\pm0.1)\times10^{-3}}$	164	$(2.7\pm0.1)\times10^{-3}$	174	$(3.1\pm0.2)\times10^{-3}$
+25	174	$(3.1\pm0.1)\times10^{-3}$	174	$(2.9\pm0.1)\times10^{-3}$	174	$(2.9\pm0.1)\times10^{-3}$	174	$(3.0\pm0.1)\times10^{-3}$	164	$\mathbf{(2.5\pm0.1)\times10^{-3}}$	164	$(2.7\pm0.1)\times10^{-3}$	174	$(3.1\pm0.2)\times10^{-3}$
+5	174	$(3.1\pm0.1)\times10^{-3}$	174	$(2.9\pm0.1)\times10^{-3}$	174	$(2.9\pm0.1)\times10^{-3}$	174	$(3.0\pm0.1)\times10^{-3}$	164	$\mathbf{(2.5\pm0.1)\times10^{-3}}$	164	$(2.7\pm0.1)\times10^{-3}$	174	$(3.1\pm0.2)\times10^{-3}$
+1	174	$(3.1\pm0.1)\times10^{-3}$	174	$(2.9\pm0.1)\times10^{-3}$	174	$(2.9\pm0.1)\times10^{-3}$	174	$(3.0\pm0.1)\times10^{-3}$	164	$\mathbf{(2.5\pm0.1)\times10^{-3}}$	164	$(2.7\pm0.1)\times10^{-3}$	174	$(3.1\pm0.2)\times10^{-3}$
0	174	$(3.1\pm0.1)\times10^{-3}$	174	$(2.9\pm0.1)\times10^{-3}$	174	$(2.9\pm0.1)\times10^{-3}$	174	$(3.0\pm0.1)\times10^{-3}$	164	$\mathbf{(2.5\pm0.1)\times10^{-3}}$	164	$(2.7\pm0.1)\times10^{-3}$	174	$(3.1\pm0.2)\times10^{-3}$
−1	174	$(6.4\pm1.8)\times10^{-3}$	174	$(3.0\pm0.1)\times10^{-3}$	174	$\mathbf{(2.9\pm0.1)\times10^{-3}}$	174	$(1.9\pm0.5)\times10^{-2}$	164	$(3.9\pm0.4)\times10^{-3}$	164	$(3.7\pm0.3)\times10^{-3}$	174	$(7.9\pm1.7)\times10^{-3}$
−5	174	$(1.2\pm0.1)\times10^{-1}$	174	$(6.8\pm0.7)\times10^{-2}$	174	$(5.7\pm0.6)\times10^{-2}$	174	$(6.3\pm0.9)\times10^{-1}$	164	$(5.3\pm0.6)\times10^{-2}$	164	$\mathbf{(5.3\pm0.6)\times10^{-2}}$	174	$(1.7\pm0.2)\times10^{-1}$
−25	162	$(1.4\pm0.0)\times10^{0}$	162	$\mathbf{(1.2\pm0.0)\times10^{0}}$	162	$(1.2\pm0.0)\times10^{0}$	162	$(5.3\pm0.2)\times10^{0}$	152	$(1.4\pm0.0)\times10^{0}$	152	$(1.4\pm0.0)\times10^{0}$	162	$(1.8\pm0.0)\times10^{0}$
−50	139	$(1.7\pm0.0)\times10^{0}$	139	$\mathbf{(1.3\pm0.0)\times10^{0}}$	139	$(1.4\pm0.0)\times10^{0}$	139	$(5.9\pm0.2)\times10^{0}$	130	$(1.6\pm0.0)\times10^{0}$	130	$(1.5\pm0.0)\times10^{0}$	139	$(2.0\pm0.0)\times10^{0}$
−75	115	$(1.9\pm0.0)\times10^{0}$	114	$\mathbf{(1.5\pm0.0)\times10^{0}}$	115	$(1.6\pm0.0)\times10^{0}$	115	$(6.8\pm0.3)\times10^{0}$	105	$(1.8\pm0.0)\times10^{0}$	105	$(1.7\pm0.0)\times10^{0}$	114	$(2.3\pm0.0)\times10^{0}$

Table 6: Experiment One (CIFAR-10): additional function-change measures for the same neuroadaptation conditions. Results are mean $\pm$ standard error over 20 independent repeats. D_{KL} is provided by *Eqn.* 59 and $\Delta a = a_{\text{post}} - a_{\text{pre}}$ is the change in test accuracy in percentage points; positive values indicate increased accuracy. The baseline row reports the unadapted test accuracy a_0 as a percentage. Bold values indicate the smallest mean D_{KL} and the accuracy change closest to zero within each neuroadaptation condition.

Neuroadaptation	$o_{\text{none}},\ \vec{\psi}_{\text{decay}}$		$o_{\text{constant}},\ \vec{\psi}_{\text{decay}}$		$o_{\text{trainable}},\ \vec{\psi}_{\text{decay}}$		$o_{\text{trainable}},\ \vec{\psi}_{\text{none}}$		$o_{\text{trainable}},\ \vec{\psi}_{\text{conetant}}$		$o_{\text{trainable}},\ \vec{\psi}_{\text{trainable}}$		$o_{\text{trainable}},\ \vec{b}_{\text{adjusted}}$	
	D_{KL}	Δa (pp)	D_{KL}	Δa (pp)	D_{KL}	Δa (pp)	D_{KL}	Δa (pp)	D_{KL}	Δa (pp)	D_{KL}	Δa (pp)	D_{KL}	Δa (pp)
+75	$(1.3\pm0.2)\times10^{-9}$	-0.0 ± 0.0	$(1.5\pm0.2)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.2\pm0.1)\times10^{-9}$	-0.0 ± 0.0	$(1.3\pm0.2)\times10^{-9}$	$+0.0\pm0.0$	$\mathbf{(1.1\pm0.2)\times10^{-9}}$	$\mathbf{0.0\pm0.0}$	$(1.2\pm0.1)\times10^{-9}$	-0.0 ± 0.0	$(1.6\pm0.2)\times10^{-9}$	$+0.0\pm0.0$
+50	$(1.4\pm0.1)\times10^{-9}$	-0.0 ± 0.0	$(1.5\pm0.2)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.5\pm0.2)\times10^{-9}$	-0.0 ± 0.0	$(1.4\pm0.2)\times10^{-9}$	$+0.0\pm0.0$	$\mathbf{(1.1\pm0.2)\times10^{-9}}$	$\mathbf{0.0\pm0.0}$	$(1.2\pm0.1)\times10^{-9}$	-0.0 ± 0.0	$(1.7\pm0.2)\times10^{-9}$	$+0.0\pm0.0$
+25	$(1.4\pm0.1)\times10^{-9}$	-0.0 ± 0.0	$(1.4\pm0.2)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.3\pm0.2)\times10^{-9}$	-0.0 ± 0.0	$(1.2\pm0.1)\times10^{-9}$	$+0.0\pm0.0$	$\mathbf{(1.1\pm0.2)\times10^{-9}}$	$\mathbf{0.0\pm0.0}$	$(1.3\pm0.2)\times10^{-9}$	-0.0 ± 0.0	$(1.6\pm0.2)\times10^{-9}$	$+0.0\pm0.0$
+5	$(1.4\pm0.2)\times10^{-9}$	-0.0 ± 0.0	$(1.3\pm0.2)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.3\pm0.2)\times10^{-9}$	-0.0 ± 0.0	$(1.3\pm0.1)\times10^{-9}$	$+0.0\pm0.0$	$(1.1\pm0.2)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$\mathbf{(1.0\pm0.2)\times10^{-9}}$	-0.0 ± 0.0	$(1.6\pm0.2)\times10^{-9}$	$+0.0\pm0.0$
+1	$(1.3\pm0.2)\times10^{-9}$	-0.0 ± 0.0	$(1.4\pm0.3)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.3\pm0.2)\times10^{-9}$	-0.0 ± 0.0	$(1.1\pm0.2)\times10^{-9}$	$+0.0\pm0.0$	$\mathbf{(1.0\pm0.1)\times10^{-9}}$	$\mathbf{0.0\pm0.0}$	$(1.1\pm0.1)\times10^{-9}$	-0.0 ± 0.0	$(1.7\pm0.2)\times10^{-9}$	$+0.0\pm0.0$
0	$(1.3\pm0.1)\times10^{-9}$	-0.0 ± 0.0	$(1.5\pm0.2)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.6\pm0.1)\times10^{-9}$	-0.0 ± 0.0	$(1.2\pm0.1)\times10^{-9}$	$+0.0\pm0.0$	$\mathbf{(1.0\pm0.2)\times10^{-9}}$	$\mathbf{0.0\pm0.0}$	$(1.2\pm0.2)\times10^{-9}$	-0.0 ± 0.0	$(1.7\pm0.2)\times10^{-9}$	$+0.0\pm0.0$
−1	$(8.1\pm4.1)\times10^{-9}$	-0.0 ± 0.0	$(1.5\pm0.2)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$\mathbf{(1.2\pm0.1)\times10^{-9}}$	-0.0 ± 0.0	$(5.3\pm2.4)\times10^{-8}$	$+0.0\pm0.0$	$(1.7\pm0.3)\times10^{-9}$	-0.0 ± 0.0	$(1.4\pm0.2)\times10^{-9}$	-0.0 ± 0.0	$(1.1\pm0.5)\times10^{-8}$	-0.0 ± 0.0
−5	$(1.3\pm0.2)\times10^{-6}$	$\mathbf{+0.0\pm0.0}$	$(6.3\pm1.3)\times10^{-7}$	-0.0 ± 0.0	$(4.5\pm0.9)\times10^{-7}$	-0.0 ± 0.0	$(4.1\pm0.9)\times10^{-5}$	$+0.0\pm0.0$	$\mathbf{(3.7\pm0.8)\times10^{-7}}$	$+0.0\pm0.0$	$(3.8\pm0.7)\times10^{-7}$	-0.0 ± 0.0	$(3.3\pm0.7)\times10^{-6}$	$\mathbf{-0.0\pm0.0}$
−25	$(2.0\pm0.1)\times10^{-4}$	-0.0 ± 0.0	$\mathbf{(1.4\pm0.0)\times10^{-4}}$	-0.0 ± 0.0	$(1.5\pm0.1)\times10^{-4}$	-0.0 ± 0.0	$(2.2\pm0.2)\times10^{-3}$	-0.1 ± 0.0	$(2.0\pm0.1)\times10^{-4}$	$\mathbf{-0.0\pm0.0}$	$(2.0\pm0.1)\times10^{-4}$	-0.0 ± 0.0	$(3.0\pm0.2)\times10^{-4}$	-0.1 ± 0.0
−50	$(2.6\pm0.1)\times10^{-4}$	-0.0 ± 0.0	$\mathbf{(1.8\pm0.1)\times10^{-4}}$	-0.0 ± 0.0	$(1.9\pm0.1)\times10^{-4}$	-0.0 ± 0.0	$(2.7\pm0.2)\times10^{-3}$	-0.1 ± 0.0	$(2.5\pm0.1)\times10^{-4}$	$\mathbf{-0.0\pm0.0}$	$(2.5\pm0.1)\times10^{-4}$	-0.0 ± 0.0	$(3.9\pm0.2)\times10^{-4}$	-0.1 ± 0.0
−75	$(3.4\pm0.1)\times10^{-4}$	$\mathbf{-0.0\pm0.0}$	$\mathbf{(2.3\pm0.1)\times10^{-4}}$	-0.1 ± 0.0	$(2.5\pm0.1)\times10^{-4}$	-0.1 ± 0.0	$(3.5\pm0.2)\times10^{-3}$	-0.1 ± 0.0	$(3.3\pm0.1)\times10^{-4}$	-0.0 ± 0.0	$(3.2\pm0.1)\times10^{-4}$	-0.1 ± 0.0	$(5.2\pm0.2)\times10^{-4}$	-0.1 ± 0.0
Baseline	–	44.2 ± 0.1	–	44.1 ± 0.1	–	44.1 ± 0.1	–	44.0 ± 0.1	–	43.9 ± 0.1	–	44.1 ± 0.1	–	44.0 ± 0.1

Furthermore, CIFAR-10 results (*Tab.* 5 and 6) show a much less dramatic rise in function drift, even with a significant number of degenerated neurons. This plateauing is reflected in KL divergence, relative epsilon and test-set accuracy, demonstrating robustness despite a large decrease in the summed singular-value count. This greater tolerance to neurodegeneration suggests a dataset-dependent effect and may intuitively indicate a greater redundancy in networks at constant width for simpler datasets. Additionally, the constant intrinsic length appears more impactful, and, by contrast, the decaying $\vec{\psi}$ approach performed better on CIFAR-10 than on Caltech256. Constant versus trainable may show comparable performance because all cases lack post-neuroadaptive retraining, but these approaches do differ in initial training pre-neuroadaptation regardless, so they have the potential to act as generally useful parameters for the model.

The CIFAR-100 results of *Tab.* 7 and 8 remain more comparable to Caltech256, likely due to their many-class labellings.

Overall, neurogenesis and diagonalisation appear inconsequential to performance or even logit distributions. This directly confirms the theoretical exact function invariance enabled through the approach up to numerical precision. Moreover, the sparsified networks reduce from 100201 parameters in this layer to just 301, whilst remarkably retaining exact performance. This is a 99.7% reduction, with no measurable loss in classification performance across all datasets tested — this is

Table 7: Experiment One (CIFAR-100): singular-value structure and relative function change under neuroadaptation at layer $L = 1$ of the $[3072, 100, 100, 100]$ classifier. Results are mean $\pm$ standard error over 20 independent repeats. Row 0 applies diagonalisation only; $+k$ subsequently applies k neurogenesis operations in a single step and $-k$ applies k neurodegeneration operations in a single step. Σ_0 and Σ_1 denote the sums of singular values before and after neuroadaptation, respectively. Singular-value standard errors are omitted for legibility; the largest singular-value standard error is 0.23. ϵ_{rel} is the *Eqn.* 58's metric, bold indicates the smallest value within each neuroadaptation condition.

Neuroadaptation	$o_{\text{none}},\ \vec{\psi}_{\text{decay}}$ $\Sigma_0 = 260$		$o_{\text{constant}},\ \vec{\psi}_{\text{decay}}$ $\Sigma_0 = 260$		$o_{\text{trainable}},\ \vec{\psi}_{\text{decay}}$ $\Sigma_0 = 260$		$o_{\text{trainable}},\ \vec{\psi}_{\text{none}}$ $\Sigma_0 = 260$		$o_{\text{trainable}},\ \vec{\psi}_{\text{constant}}$ $\Sigma_0 = 252$		$o_{\text{trainable}},\ \vec{\psi}_{\text{trainable}}$ $\Sigma_0 = 252$		$o_{\text{trainable}},\ \vec{b}_{\text{adjusted}}$ $\Sigma_0 = 260$	
	Σ_1	ϵ_{rel} (%)	Σ_1	ϵ_{rel} (%)	Σ_1	ϵ_{rel} (%)	Σ_1	ϵ_{rel} (%)	Σ_1	ϵ_{rel} (%)	Σ_1	ϵ_{rel} (%)	Σ_1	ϵ_{rel} (%)
+75	260	$(1.7\pm0.1)\times10^{-3}$	260	$(1.7\pm0.1)\times10^{-3}$	260	$(1.6\pm0.1)\times10^{-3}$	260	$(1.6\pm0.0)\times10^{-3}$	252	$\mathbf{(1.4\pm0.1)\times10^{-3}}$	252	$(1.5\pm0.1)\times10^{-3}$	260	$(1.7\pm0.1)\times10^{-3}$
+50	260	$(1.7\pm0.1)\times10^{-3}$	260	$(1.7\pm0.1)\times10^{-3}$	260	$(1.6\pm0.1)\times10^{-3}$	260	$(1.6\pm0.0)\times10^{-3}$	252	$\mathbf{(1.4\pm0.1)\times10^{-3}}$	252	$(1.5\pm0.1)\times10^{-3}$	260	$(1.7\pm0.1)\times10^{-3}$
+25	260	$(1.7\pm0.1)\times10^{-3}$	260	$(1.7\pm0.1)\times10^{-3}$	260	$(1.6\pm0.1)\times10^{-3}$	260	$(1.6\pm0.0)\times10^{-3}$	252	$\mathbf{(1.4\pm0.1)\times10^{-3}}$	252	$(1.5\pm0.1)\times10^{-3}$	260	$(1.7\pm0.1)\times10^{-3}$
+5	260	$(1.7\pm0.1)\times10^{-3}$	260	$(1.7\pm0.1)\times10^{-3}$	260	$(1.6\pm0.1)\times10^{-3}$	260	$(1.6\pm0.0)\times10^{-3}$	252	$\mathbf{(1.4\pm0.1)\times10^{-3}}$	252	$(1.5\pm0.1)\times10^{-3}$	260	$(1.7\pm0.1)\times10^{-3}$
+1	260	$(1.7\pm0.1)\times10^{-3}$	260	$(1.7\pm0.1)\times10^{-3}$	260	$(1.6\pm0.1)\times10^{-3}$	260	$(1.6\pm0.0)\times10^{-3}$	252	$\mathbf{(1.4\pm0.1)\times10^{-3}}$	252	$(1.5\pm0.1)\times10^{-3}$	260	$(1.7\pm0.1)\times10^{-3}$
0	260	$(1.7\pm0.1)\times10^{-3}$	260	$(1.7\pm0.1)\times10^{-3}$	260	$(1.6\pm0.1)\times10^{-3}$	260	$(1.6\pm0.0)\times10^{-3}$	252	$\mathbf{(1.4\pm0.1)\times10^{-3}}$	252	$(1.5\pm0.1)\times10^{-3}$	260	$(1.7\pm0.1)\times10^{-3}$
−1	260	$(4.4\pm0.1)\times10^{-1}$	260	$(4.4\pm0.1)\times10^{-1}$	259	$(4.5\pm0.2)\times10^{-1}$	259	$(4.6\pm0.1)\times10^{-1}$	252	$\mathbf{(3.5\pm0.1)\times10^{-1}}$	252	$(3.6\pm0.2)\times10^{-1}$	259	$(4.5\pm0.1)\times10^{-1}$
−5	257	$(2.0\pm0.0)\times10^{0}$	257	$\mathbf{(1.9\pm0.0)\times10^{0}}$	257	$(2.0\pm0.0)\times10^{0}$	256	$(2.1\pm0.0)\times10^{0}$	249	$(2.0\pm0.0)\times10^{0}$	249	$(2.1\pm0.0)\times10^{0}$	257	$(2.0\pm0.0)\times10^{0}$
−25	237	$(2.6\pm0.0)\times10^{0}$	237	$\mathbf{(2.5\pm0.0)\times10^{0}}$	237	$(2.6\pm0.0)\times10^{0}$	237	$(2.7\pm0.0)\times10^{0}$	229	$(2.7\pm0.0)\times10^{0}$	229	$(2.7\pm0.0)\times10^{0}$	237	$(2.6\pm0.0)\times10^{0}$
−50	197	$(1.3\pm0.0)\times10^{+1}$	197	$(1.4\pm0.0)\times10^{+1}$	197	$(1.4\pm0.0)\times10^{+1}$	197	$\mathbf{(1.3\pm0.0)\times10^{+1}}$	188	$(1.6\pm0.0)\times10^{+1}$	188	$(1.7\pm0.0)\times10^{+1}$	197	$(1.4\pm0.1)\times10^{+1}$
−75	117	$(6.2\pm0.0)\times10^{+1}$	117	$(6.1\pm0.0)\times10^{+1}$	117	$(6.1\pm0.0)\times10^{+1}$	117	$(6.1\pm0.0)\times10^{+1}$	111	$(6.2\pm0.0)\times10^{+1}$	111	$(6.2\pm0.0)\times10^{+1}$	117	$\mathbf{(6.1\pm0.0)\times10^{+1}}$

Table 8: Experiment One (CIFAR-100): additional function-change measures for the same neuroadaptation conditions. Results are mean $\pm$ standard error over 20 independent repeats. D_{KL} is provided by *Eqn.* 59 and $\Delta a = a_{\text{post}} - a_{\text{pre}}$ is the change in test accuracy in percentage points; positive values indicate increased accuracy. The baseline row reports the unadapted test accuracy a_0 as a percentage. Bold values indicate the smallest mean D_{KL} and the accuracy change closest to zero within each neuroadaptation condition.

Neuroadaptation	$o_{\text{none}},\ \vec{\psi}_{\text{decay}}$		$o_{\text{constant}},\ \vec{\psi}_{\text{decay}}$		$o_{\text{trainable}},\ \vec{\psi}_{\text{decay}}$		$o_{\text{trainable}},\ \vec{\psi}_{\text{none}}$		$o_{\text{trainable}},\ \vec{\psi}_{\text{constant}}$		$o_{\text{trainable}},\ \vec{\psi}_{\text{trainable}}$		$o_{\text{trainable}},\ \vec{b}_{\text{adjusted}}$	
	D_{KL}	Δa (pp)	D_{KL}	Δa (pp)	D_{KL}	Δa (pp)	D_{KL}	Δa (pp)	D_{KL}	Δa (pp)	D_{KL}	Δa (pp)	D_{KL}	Δa (pp)
+75	$(1.6\pm0.2)\times10^{-9}$	$+0.0\pm0.0$	$(1.5\pm0.2)\times10^{-9}$	$+0.0\pm0.0$	$\mathbf{(1.2\pm0.2)\times10^{-9}}$	-0.0 ± 0.0	$(1.6\pm0.2)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.2\pm0.1)\times10^{-9}$	$+0.0\pm0.0$	$(1.6\pm0.2)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.4\pm0.2)\times10^{-9}$	$+0.0\pm0.0$
+50	$(1.6\pm0.2)\times10^{-9}$	$+0.0\pm0.0$	$(1.6\pm0.2)\times10^{-9}$	$+0.0\pm0.0$	$(1.2\pm0.2)\times10^{-9}$	-0.0 ± 0.0	$(1.6\pm0.2)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$\mathbf{(1.1\pm0.2)\times10^{-9}}$	$+0.0\pm0.0$	$(1.6\pm0.2)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.4\pm0.2)\times10^{-9}$	$+0.0\pm0.0$
+25	$(1.6\pm0.2)\times10^{-9}$	$+0.0\pm0.0$	$(1.5\pm0.2)\times10^{-9}$	$+0.0\pm0.0$	$\mathbf{(1.2\pm0.2)\times10^{-9}}$	-0.0 ± 0.0	$(1.6\pm0.2)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.2\pm0.1)\times10^{-9}$	$+0.0\pm0.0$	$(1.6\pm0.2)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.4\pm0.2)\times10^{-9}$	$+0.0\pm0.0$
+5	$(1.5\pm0.1)\times10^{-9}$	$+0.0\pm0.0$	$(1.4\pm0.2)\times10^{-9}$	$+0.0\pm0.0$	$(1.2\pm0.2)\times10^{-9}$	-0.0 ± 0.0	$(1.5\pm0.2)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$\mathbf{(1.2\pm0.1)\times10^{-9}}$	$+0.0\pm0.0$	$(1.6\pm0.2)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.5\pm0.2)\times10^{-9}$	$+0.0\pm0.0$
+1	$(1.6\pm0.2)\times10^{-9}$	$+0.0\pm0.0$	$(1.5\pm0.2)\times10^{-9}$	$+0.0\pm0.0$	$\mathbf{(1.2\pm0.2)\times10^{-9}}$	-0.0 ± 0.0	$(1.9\pm0.2)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.2\pm0.2)\times10^{-9}$	$+0.0\pm0.0$	$(1.4\pm0.2)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.4\pm0.2)\times10^{-9}$	$+0.0\pm0.0$
0	$(1.7\pm0.2)\times10^{-9}$	$+0.0\pm0.0$	$(1.4\pm0.2)\times10^{-9}$	$+0.0\pm0.0$	$(1.2\pm0.1)\times10^{-9}$	-0.0 ± 0.0	$(1.7\pm0.2)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$\mathbf{(1.2\pm0.2)\times10^{-9}}$	$+0.0\pm0.0$	$(1.4\pm0.2)\times10^{-9}$	$\mathbf{0.0\pm0.0}$	$(1.5\pm0.2)\times10^{-9}$	$+0.0\pm0.0$
−1	$(7.8\pm0.4)\times10^{-5}$	-0.0 ± 0.0	$(7.6\pm0.5)\times10^{-5}$	-0.0 ± 0.0	$(8.3\pm0.6)\times10^{-5}$	-0.0 ± 0.0	$(8.9\pm0.5)\times10^{-5}$	-0.0 ± 0.0	$\mathbf{(4.9\pm0.4)\times10^{-5}}$	$\mathbf{-0.0\pm0.0}$	$(5.2\pm0.5)\times10^{-5}$	-0.0 ± 0.0	$(7.9\pm0.4)\times10^{-5}$	-0.0 ± 0.0
−5	$(8.9\pm0.3)\times10^{-4}$	-0.1 ± 0.0	$\mathbf{(8.1\pm0.2)\times10^{-4}}$	-0.1 ± 0.0	$(8.8\pm0.3)\times10^{-4}$	-0.1 ± 0.0	$(1.0\pm0.0)\times10^{-3}$	-0.1 ± 0.0	$(9.8\pm0.3)\times10^{-4}$	-0.1 ± 0.0	$(10.0\pm0.3)\times10^{-4}$	$\mathbf{-0.1\pm0.0}$	$(9.3\pm0.3)\times10^{-4}$	-0.1 ± 0.0
−25	$(1.5\pm0.1)\times10^{-3}$	-0.1 ± 0.0	$\mathbf{(1.4\pm0.0)\times10^{-3}}$	-0.1 ± 0.0	$(1.5\pm0.0)\times10^{-3}$	-0.1 ± 0.0	$(1.6\pm0.1)\times10^{-3}$	$\mathbf{-0.1\pm0.0}$	$(1.7\pm0.1)\times10^{-3}$	-0.1 ± 0.0	$(1.7\pm0.1)\times10^{-3}$	-0.1 ± 0.0	$(1.6\pm0.0)\times10^{-3}$	-0.1 ± 0.0
−50	$(4.7\pm0.4)\times10^{-2}$	-1.1 ± 0.1	$(5.4\pm0.4)\times10^{-2}$	-1.2 ± 0.1	$(5.0\pm0.4)\times10^{-2}$	-1.2 ± 0.1	$\mathbf{(4.5\pm0.4)\times10^{-2}}$	$\mathbf{-1.1\pm0.1}$	$(7.1\pm0.3)\times10^{-2}$	-1.5 ± 0.1	$(7.3\pm0.3)\times10^{-2}$	-1.5 ± 0.1	$(5.2\pm0.4)\times10^{-2}$	-1.2 ± 0.1
−75	$(8.1\pm0.1)\times10^{-1}$	-11.5 ± 0.1	$(8.2\pm0.1)\times10^{-1}$	-11.6 ± 0.1	$(7.9\pm0.1)\times10^{-1}$	-11.4 ± 0.1	$\mathbf{(7.6\pm0.1)\times10^{-1}}$	$\mathbf{-11.2\pm0.1}$	$(8.5\pm0.1)\times10^{-1}$	-11.7 ± 0.1	$(8.5\pm0.1)\times10^{-1}$	-11.8 ± 0.1	$(7.7\pm0.1)\times10^{-1}$	-11.3 ± 0.1
Baseline	–	18.6 ± 0.1	–	18.6 ± 0.0	–	18.6 ± 0.0	–	18.6 ± 0.1	–	18.6 ± 0.0	–	18.7 ± 0.0	–	18.7 ± 0.0

without any retraining or fine-tuning. Moreover, this reaches 99.98% sparsification for no intrinsic length or $\vec{\psi}$.

Neurodegeneration also approximates function well for most networks up to around 50% width reduction, while deterioration occurs more sharply when many singular values are removed. It appears that CIFAR-10 is more resilient to this process, likely due to the substantially fewer output classes, which may require fewer functionally important directions to maintain its prediction. Many-class datasets showed greater deterioration in relative logit change and KL divergence, not just test accuracy, suggesting less redundancy in their fixed-width layers, where more complex transforms are required of the weights to separate classes. This is expected for a more demanding dataset.

Furthermore, we observed a negligible difference between constant and trainable $\vec{\psi}$ or o in the table. This is directly because these networks did not undergo retraining, where the constant or trained values would exhibit static or changing values. Therefore, the nature of these parameters did not have the opportunity to influence the later invariance, leading to comparable results. Intrinsic length appeared to improve neurodegeneration invariance, while $\vec{\psi}$-compensations appeared less influential and produced more comparable performance; however, the preference for using it was clearer on CIFAR-10, where its absence resulted in poorer performance at -75 neurons degenerated.

K.2 Experiment Two: Singular Value and Bias Spectra Evolutions

The experiment displayed in *Fig.* 2 shows the evolution of the layerwise singular value and bias spectra of a static (non-neuroadaptive) isotropic multilayer perceptron over 40 repeats for the architecture $[3072, 100, 100, 10]$, trained on CIFAR-10. This experiment examines spectral and bias shifts during training and how they may affect thresholding choices. Although they have little bearing on thresholding, the bias elements are included for completeness and to allow a more holistic comparison of the training dynamics that differ between anisotropic and isotropic networks.

The spectra are pooled across all independent repeats at each epoch snapshot, enabling an evolving joyplot of the results over 40 epochs. We performed the aforementioned standard training approach as described above, including input standardisation. As with all experiments, we consistently used orthogonal initialisation to ensure all initial singular values were equal to 1. This reduces ambiguity about whether a singular value is increasingly significant to the function or decaying to become negligible by providing a known initial value of 1. This contrasts with random-normal approaches, which typically have a more distributed spectrum, obfuscating such trends.

We used a slightly boosted learning rate of $\eta = 0.05$ to accentuate long-term trend evolution over the 40-epoch window, allowing more repeats to be taken. This made it easier to observe the increase, decay and separation of singular values in this experiment. This was preferred because final classification accuracy was not important for this experiment; rather, we needed a clearer diagnosis of whether singular-value thresholding is experimentally plausible. Together with experiment three, we deemed this thresholding appropriate.

The plots use log-scaled heights and signed-log ($\operatorname{sign} l \ln(1 + |l|)$) x-axis scaling, with the x-axis range custom-chosen to include all notable details.

To complement the CIFAR-10, 100-width plot presented in *Sec.* 3, we provide an array of additional spectra plots. Particularly isotropic (isotropic-tanh) and anisotropic (standard, elementwise tanh) are trained under identical conditions, allowing us to study how these superficially similar activation functions actually lead to drastically different spectra through learning and thus approach the problem very differently. Furthermore, a broad sweep of widths for both of these is provided: $[3072, m, m, 10]$, for $m \in \{100, 500, 1000\}$, whose purpose is to examine how the spectra change across a range of widths (accessible by neuroadaptation), and how this may also impact thresholding. Finally, motivated by the analyses of *App.* K.1, where neurodegeneration appeared less impactful for easier datasets such as CIFAR-10, but had a sharper degradation in Caltech256, we repeat the isotropic width-sweeps for a Caltech256 network to examine how the spectra differ in such datasets. These Caltech256 networks are trained otherwise identically to CIFAR-10, but for an architecture of $[12288, m, m, 256]$, for $m \in \{100, 500, 1000\}$.

K.2.1 Comparison to Identical Architecture with Standard Tanh

The experiments in this section are trained identically, with plots showing evolving layerwise singular value and bias spectra for a non-neuroadaptive isotropic-tanh or standard-tanh MLP trained on CIFAR-10 classification. We collate the spectra over 40 independent repeats, with an emerging strong consistency. Singular values begin at one, indicated by the dashed line, due to orthogonal initialisation, and biases at zero due to zero initialisation. The plots of *Fig.* 4 and 5 show an isotropic-tanh and standard-tanh 100-hidden width CIFAR-10 classification networks respectively. *Fig.* 4 is identical to *Fig.* 2 of *Sec.* 3.

The isotropic network shows key diversification across all layers, but minimal spread around zero for layer-one biases. Both layer-zero and layer-one singular value spectra show a general bulk distribution that initially increases with training before separating into two distinct clusters, the smallest of which appears to decay toward zero, while the higher cluster remains stable. The split from this unimodal to bimodal cluster occurs rapidly within the first epoch. It is this descending tail that thresholding may begin to progressively neurodegenerate if applied. The most striking characteristic is arguably the layer-two ($\mathbb{R}^{100} \to \mathbb{R}^{10}$ output-layer), which rapidly separates into several distinct singular-value clusters with clear gaps between the dense bands. These likely reflect strong consistency over the 40 repeats for the 10 singular values in this layer, showing that isotropic-tanh reliably finds a comparable solution. This may result from the flattened loss around orthogonal-group orbits, where similar trajectories are produced despite differing initialisations, compared with the more nuanced loss structure of discrete-group-defined activations, which may cause more unpredictable deflections. The

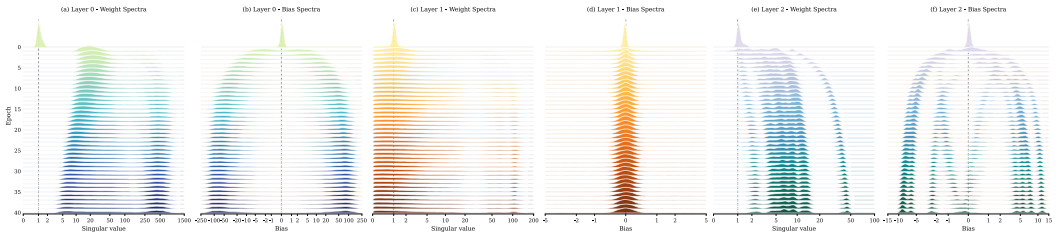


Figure 4: Experiment Two (*App.* K.2) evolving layerwise singular value and bias spectra for a $[3072, 100, 100, 10]$ **isotropic-tanh** CIFAR-10 classifier.

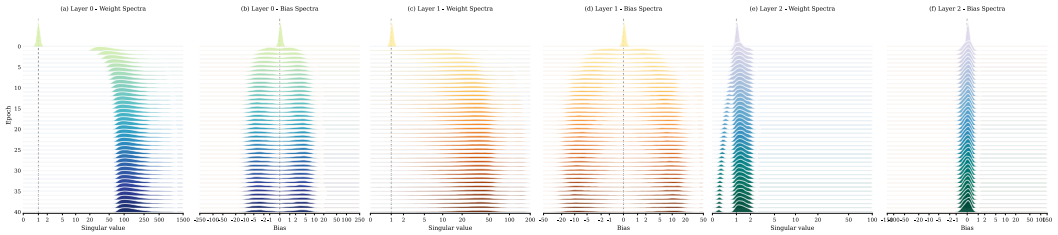


Figure 5: Experiment Two (*App.* K.2) evolving layerwise singular value and bias spectra for a $[3072, 100, 100, 10]$ **standard-tanh** CIFAR-10 classifier.

layer-two biases also show a rich structure of distinct bands (some progressively decaying to zero), unlike the simple bimodal structure of layer-zero biases or the unimodal, zero-centred cluster of layer-one. The decaying band crossovers which occur in layer-two biases suggest these correspond to distinct elementwise outputs, since the crossovers do not appear to induce any interference in the decaying trajectories.

Practically, because isotropic-tanh is spherically symmetric and we use signed-log x-axis scaling, the bimodal distribution of layer-zero biases instead indicates a unimodal, non-zero distribution of magnitudes shown in *Fig.* 6 — this basis-freedom is constrained by the lack of two-sided orthogonal symmetry of layer-zero and layer-two and the dataset's symmetry breaking. This also clarifies a bifurcation in layer-one biases.

Overall, this suggests that isotropic networks are particularly anisotropic in their transform: some directions are amplified considerably more than others, and this is consistent between initialisations. Layers zero and two appear to produce the dominant transforms in this case, while layer-one has only a few singular values that remain stably separated from the general bulk distribution, implying that only a few directions are anisotropically amplified. All layers also appear to have large-magnitude singular values, suggesting the network amplifies scale considerably, perhaps using saturation of the isotropic-tanh for non-linear reshaping.

Overall, this is significantly dissimilar to the standard-tanh case. The standard-tanh case shows a more pronounced, non-decaying spectrum in layers zero and one, suggesting a largely coherent bulk evolution that may contribute to pruning difficulties. The final layer shows little diversification, with training staying peaked around the initialised values and only a small decay in some singular values — suggesting it may act largely as a semi-orthogonal transform. Thus, anisotropic functions (standard-tanh) conversely have a more isotropic-like affine transform, with all directions scaled similarly, markedly different from isotropic-tanh's behaviour. Besides layer-two, they have somewhat comparable amplification, suggesting the networks prefer a specific activation scale, indicative of exploiting the saturating non-linear regime. Standard-tanh also appears to have distinctly different layers dominating the transform; layer-two shows little affine specialisation, unlike the rich structure in isotropic-tanh's last layer. However, layer-two's bias-magnitude structure is more nuanced in standard-tanh, as shown in *Fig.* 7. Unlike isotropic-tanh, care is needed when interpreting bias-magnitude over elementwise distribution in standard-tanh because it lacks rotational symmetry.

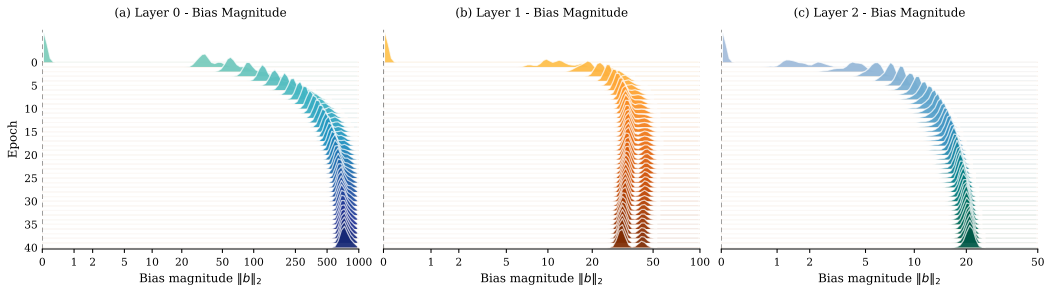


Figure 6: Experiment Two (*App.* K.2) evolving layerwise bias-magnitude spectra for a $[3072, 100, 100, 10]$ **isotropic-tanh** CIFAR-10 classifier.

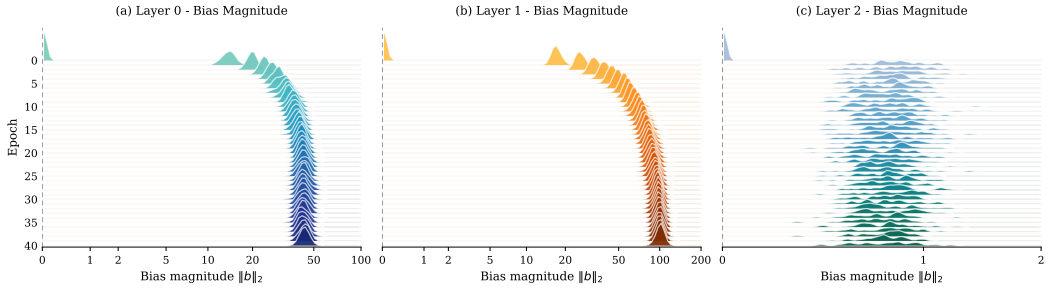


Figure 7: Experiment Two (*App.* K.2) evolving layerwise bias-magnitude spectra for a $[3072, 100, 100, 10]$ **standard-tanh** CIFAR-10 classifier.

Considered together, isotropic-tanh plots consistently show stronger stratification than in *Fig.* 5, suggesting parameters become highly anisotropic despite the isotropic activation form. The data and task are intrinsically anisotropic due to the image-to-classification objective, and the emergence of anisotropic parameters in isotropic networks indicates a clear, spontaneous symmetry breaking necessitated to achieve this, with a few singular directions becoming specialised while many decay. This contrasts with standard tanh networks, which tend to distribute their spectra as a broad bulk, aligning with more classical interpretations of them as an assortment of somewhat comparably acting individual feature detectors.

Additionally, isotropic-tanh networks appear to have a greater utilisation of the final layer compared to standard-tanh networks, whilst standard-tanh shows a smoother zeroth-layer spectrum, consistent with many directions being amplified to comparable magnitudes. Thus, they appear to exhibit very different computational regimes despite superficial similarities of the function (even identical along the axes). Thus, functional form has a marked effect on training and should not be treated as comparable—each induces different biases in optimisation. Collectively, there appears to be an inverse tradeoff between the an/isotropy of the activation and the an/isotropy of the affine map.

Isotropic tanh exhibits a distinct, well-separated ordering of its singular values, particularly in the last layer, supporting the paper's motivation for thresholded neurodegeneration. Additionally, this supports interpreting singular values as having a perturbative impact on the function — isotropic networks arrange into distinct bands of singular values, indicating increasingly amplified anisotropic scaling of activations. The strong similarity across repeats may also stem from the flat-loss orbit induced by continuous orthogonal equivariance per layer, where many optimisation trajectories may follow comparable paths up to rotation because they are not deflected by the loss structure that emerges along those orbits for permutation-derived primitives.

Overall, both demonstrate qualitatively different modes of learning, indicating a lack of true comparability between the forms despite the superficial use of tanh. Caution around classical interpretations of

spectra is warranted, as they may be predicated on elementwise activation geometry, thereby limiting their direct reapplication to isotropic networks.

These comparisons also continue into the subsequent subsection's hidden-width sweeps.

K.2.2 Comparison Over Hidden-Layer Widths

Comparisons across hidden-layer widths reveal how the spectra generalise and differ between architectures. Moreover, neuroadaptation can explore a broad range of hidden widths, so establishing whether thresholding remains robust across these changes is essential, particularly if the spectrum densifies and removes clear band-gaps between singular values in the isotropic case, which some thresholding approaches may rely upon.

We use the results from the prior subsection as examples with a hidden width of 100, while this section also includes hidden widths of 500 and 1000 for the CIFAR-10 classifiers. All other training conditions are identical.

For the isotropic case, comparing across *Figs.* 4, 8 and 9, the overarching trends are stable and consistent with those observed for *Fig.* 4. The distinct band behaviour remains pronounced, with comparable specialisation in layer-zero and two's singular values and biases. What differs is the proportion of the spectrum in the singular value bulk of layers zero and one. This suggests that as layer width increases, most new singular values join the main bulk, which shows more pronounced decay-to-zero behaviour with greater width. The epoch-wise scaling then suppresses the relative height of the peak of stable singular values in the more-positive cluster, despite a comparable number of values comprising it.

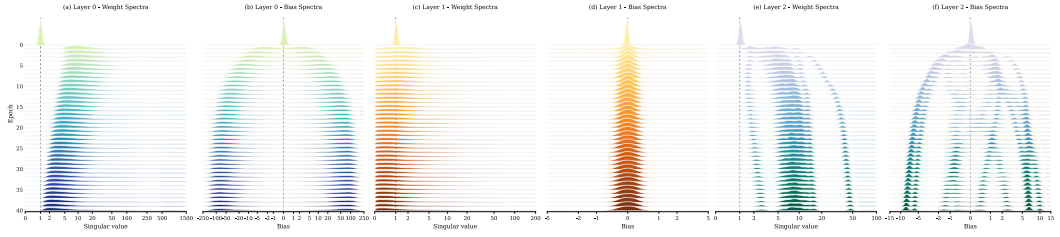


Figure 8: Experiment Two (*App.* K.2) evolving layerwise singular value and bias spectra for a $[3072, 500, 500, 10]$ **isotropic-tanh** CIFAR-10 classifier.

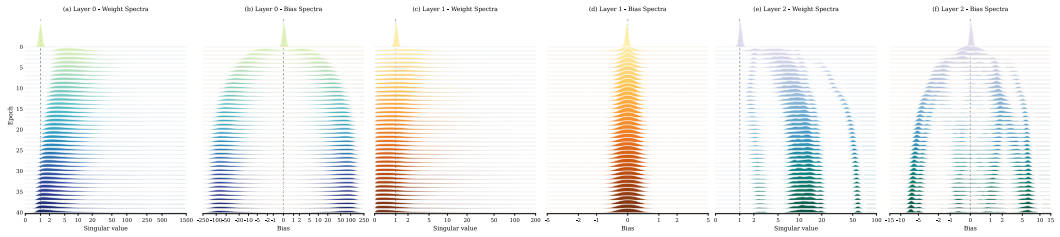


Figure 9: Experiment Two (*App.* K.2) evolving layerwise singular value and bias spectra for a $[3072, 1000, 1000, 10]$ **isotropic-tanh** CIFAR-10 classifier.

This is a reassuring result, suggesting that the more-positive cluster is important to the classification task, while the lower-decaying bulk is redundant width that can be neurodegenerated. That lower bulk scales proportionally with width, as one might expect from this hypothesis, as more redundant neurons are introduced. This suggests that regardless of width, isotropic-tanh appears to maintain a comparable solution to the task. The clear band-gap and decaying spectra therefore encourage continued use of thresholding to trigger neurodegeneration in the overly large architecture. This also raises questions regarding the nature of overparameterisation behaviours in isotropic networks.

The overall scale of singular values also remains consistent, as does the elementwise histogram of biases; however, bias magnitudes scale with width, indicating that most biases are consistently non-zero and scale roughly as $\approx \sqrt{m}$, as indicated by *Fig.* 10 (500-width case comparably interpolates this trend).

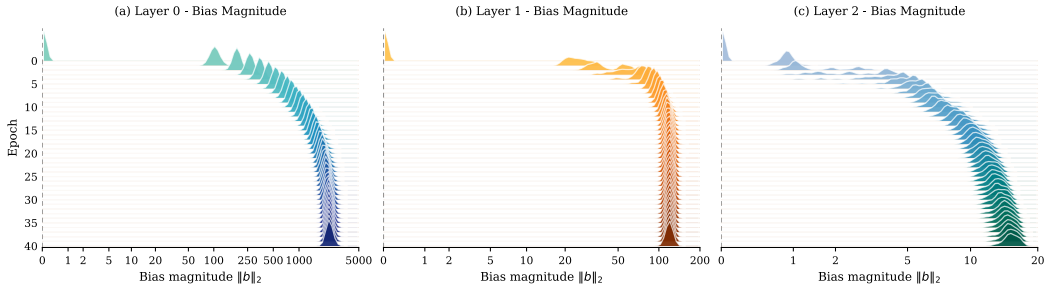


Figure 10: Experiment Two (*App.* K.2) evolving layerwise bias-magnitude spectra for a $[3072, 1000, 1000, 10]$ **isotropic-tanh** CIFAR-10 classifier.

The standard-tanh case (*Figs.* 5, 11 and 12) also reproduces comparable trends for larger widths with some distinctions. Layer-zero's main singular-value bulk coherently decreases in scale with width, while the primary peak of layer-one's and two's bulk increases with scale, with layer-two showing greater separation from its decaying tail. That decaying tail is not proportionally smaller despite increasing width, suggesting that proportions of singular values in each cluster are maintained despite increasing width, unlike isotropic-tanh's enlarging lower-magnitude cluster. This suggests a continued distinction between standard- and isotropic-tanh learning even across differing widths: the latter appears to maintain a similarly functioning subset of singular values, while the former approaches the task via an increasing number of distinct but comparably scaled feature detectors, rather than retaining just the sufficient subset. Additionally, the bias structure becomes increasingly nuanced for layer-two standard-tanh networks.

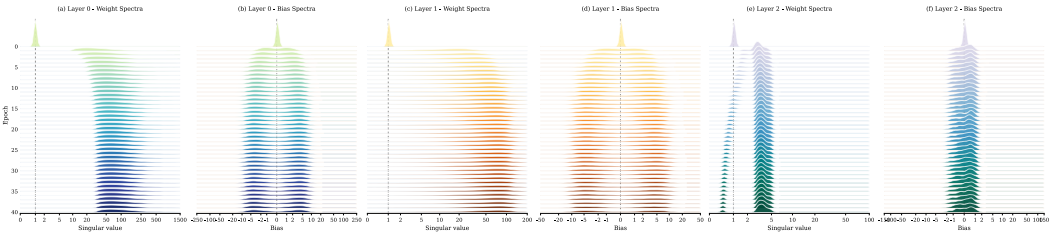


Figure 11: Experiment Two (*App.* K.2) evolving layerwise singular value and bias spectra for a $[3072, 500, 500, 10]$ **standard-tanh** CIFAR-10 classifier.

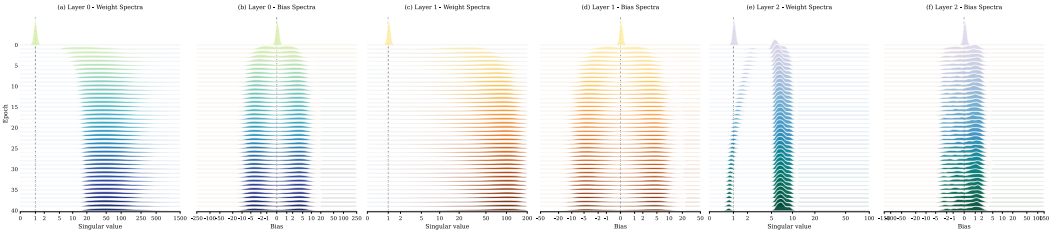


Figure 12: Experiment Two (*App.* K.2) evolving layerwise singular value and bias spectra for a $[3072, 1000, 1000, 10]$ **standard-tanh** CIFAR-10 classifier.

Overall, these results are encouraging for the use of thresholding, and are used as motivation for its various potential forms (*App.* C). Moreover, they reinforce the distinct incomparability between elementwise and isotropic functional forms, even in the seemingly most similar instances of two tanh-based functions (with identical non-linear maps along basis directions). This encourages a deeper exploration into the biases these symmetry-led primitives may bring to general networks.

Finally, the next section investigates how the spectra differ for the more complicated Caltech256 dataset across different widths when using isotropic-tanh, with this subsection's results on CIFAR-10 offering an illuminating comparison.

K.2.3 Comparison to Caltech256 Across Various Hidden-Layer Widths

These Caltech256 networks are trained identically to the CIFAR-10 cases, except for the architectural differences required for the increased number of classes and 64×64 pixel RGB inputs instead of CIFAR-10's native 32×32. We also standardised the inputs consistently, so the elementwise scale is identical across the dataset, as in CIFAR-10. Results over three hidden-layer widths are also provided: $[12288, m, m, 256]$ for $m \in \{100, 500, 1000\}$ for *Figs.* 13, 14, and 15 respectively.

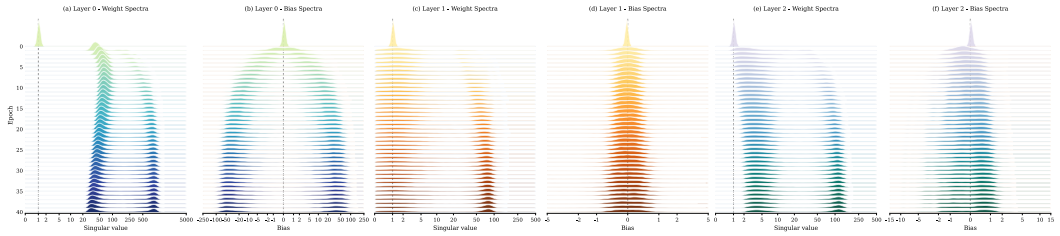


Figure 13: Experiment Two (*App.* K.2) evolving layerwise singular value and bias spectra for a $[12288, 100, 100, 256]$ **isotropic-tanh** Caltech256 classifier.

Surprisingly, Caltech256 shows very comparable overarching behaviour to CIFAR-10: layer-zero singular values have a decaying bulk with a distinct, stable, more-positive cluster remaining (albeit with somewhat sharper clusters for Caltech256); the same holds for layer-one and layer-two singular values. Crucially, this structure is now evident across all layers, whereas the small layer-two 10-singular value distribution of CIFAR-10 obscured this repeated structure, which the 256-singular value distribution doesn't. Thus, isotropic-tanh appears to be characterised by a bimodal behaviour wholly distinct from standard-tanh, and it occurs reliably across all layers — suggesting each reproduces an analogous anisotropic affine scaling for its sub-task. Seemingly, a comparable subset of singular values is reused regardless of initialisation across differing widths for the same task.

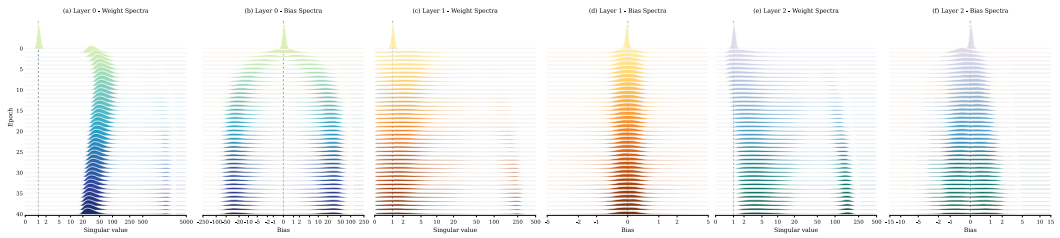


Figure 14: Experiment Two (*App.* K.2) evolving layerwise singular value and bias spectra for a $[12288, 500, 500, 256]$ **isotropic-tanh** Caltech256 classifier.

Bias structure also appears more nuanced in the Caltech256 case, as shown in *Fig.* 16 and 17. The richer structure for the 100-width case may be due to the signed-log x-axis scaling. Curiously, *Fig.* 16's bias first appears to increase to a stable value, but then eventually begins to increase again, suggesting an evolving metastable solution.

From this, we conclude an important finding: differences in function form produce fundamentally different modes of learning (even with identical non-linear maps along the axes). Standard-tanh

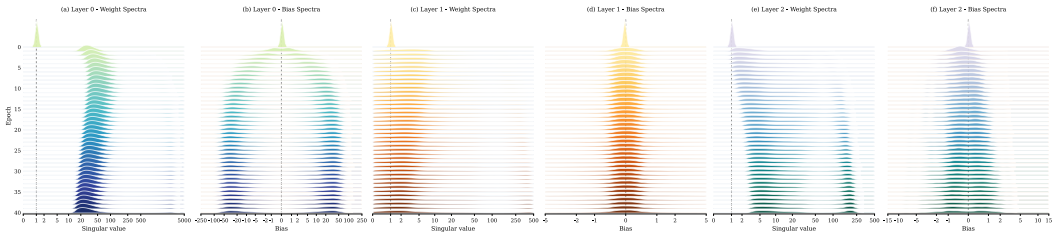


Figure 15: Experiment Two (*App.* K.2) evolving layerwise singular value and bias spectra for a $[12288, 1000, 1000, 256]$ **isotropic-tanh** Caltech256 classifier.

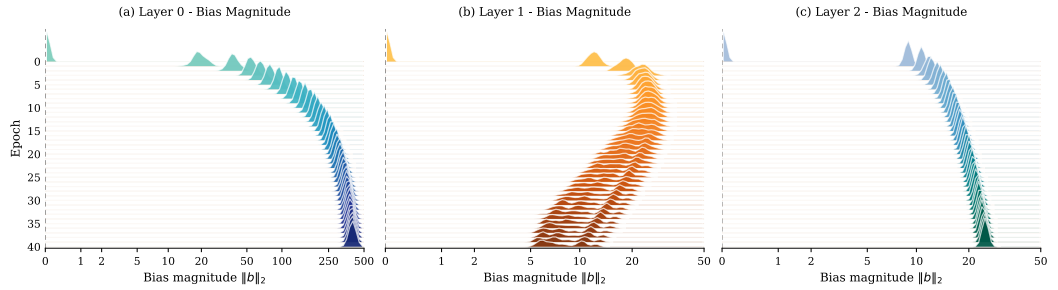


Figure 16: Experiment Two (*App.* K.2) evolving layerwise bias-magnitude spectra for a $[12288, 100, 100, 256]$ **isotropic-tanh** Caltech256 classifier.

networks are characterised by a single coherent bulk of similarly scaled so-called 'feature detectors', whereas, at sufficient width, isotropic-tanh is characterised by a bimodal distribution. This appears to be a reproducible finding across classification networks for different widths and datasets, suggesting an inherent incomparability of how the two functional forms bias a network in its optimised solutions. Thus, with increasing confidence, one can assume that even superficially similar instances of isotropic and elementwise activations behave in fundamentally distinct ways during training and their resultant approach to classification.

We argue this continues to motivate a broader taxonomic exploration of symmetry-defined primitives and how they bias, helpfully, detrimentally or in a poorly understood but importantly interpretative way (e.g. grandmother neurons [32]) general networks, regardless of overall model symmetries or available reparameterisations. Primitive symmetry alone may offer an informative avenue for further research, and these results demonstrate a clear influence on learned representations.

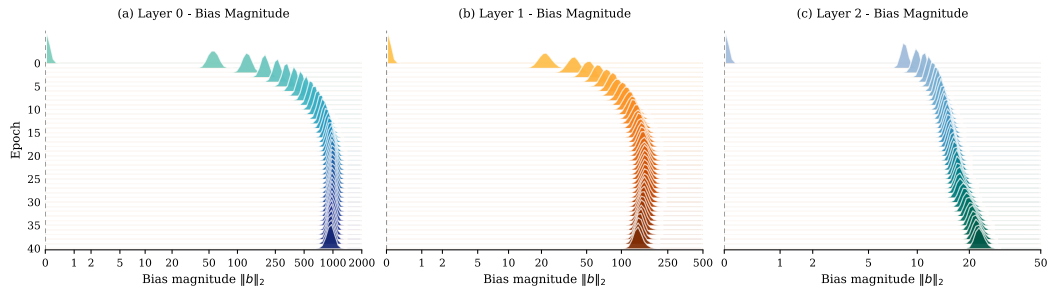


Figure 17: Experiment Two (*App.* K.2) evolving layerwise bias-magnitude spectra for a $[12288, 1000, 1000, 256]$ **isotropic-tanh** Caltech256 classifier.

K.3 Experiment Three: Thresholded Neuroadaptation to Changing Task Schedule

This experiment directly tests the threshold-based approach to neuroadaptation by observing how the isotropic multilayer perceptron adapts to a changing task scheduler. Unlike experiment one (*App.* K.1), neuroadaptation is possible after every epoch, using thresholding to automatically adapt the widths of three different layers. This enables us to determine whether this approach adapts as expected to changing task demands while maintaining competitive task performance relative to a non-adaptive control network.

Three distinct image-classification datasets are utilised: MNIST [48], Fashion-MNIST (abbreviated to FMNIST) [49], and the 'A' to 'J' character samples derived from EMNIST [50]. During training, all samples are provided as input; the scheduler only determines whether each dataset's cross-entropy term contributes to the overall loss, which the network is optimised against. Hence, the architecture is described by the structure $[2352(=28\times28\times3), \text{H1}, \text{H2}, \text{H3}, 30]$, where $\text{H1}, \text{H2}, \text{H3}$ are initialised uniformly: $\text{H1}, \text{H2}, \text{H3} \sim U\,(500, 2500)$. This was determined to be a reasonable range, producing adaptations for both initial under- and overabundance of neurons.

It was noted that when there were fewer than ≈ 500 initial neurons, the growth permitted by the $\Xi = 5$ scaffold neurons was often insufficient to meet demand, repeatedly triggering neurogenesis with too little time to grow before the next schedule. We chose the singular value thresholding at $\vartheta = 0.95$ to quickly neurodegenerate unused parameters, decaying immediately from an initial value of 1 due to the orthogonal initialisers. Using this, 5 'scaffold' neurons are maintained as a buffer below this threshold. The 30-width output corresponds to three disjoint ten-class label blocks, each softmaxed separately as a ten-class block. This structure permits the scheduler to activate a task via a Boolean switch, S_{dataset}, and the total loss is averaged to reduce discontinuity upon task-switching, as shown by *Eqn.* 61. The plot of *Fig.* 3 utilised the same learning rate as *App.* K.2, $\eta = 5\times10^{-2}$, such that 150-epochs per task-schedule was sufficient for parameters to change enough to trigger adaptation.

$$\mathcal{L}_{\text{total}} = \frac{S_{\text{MNIST}}\mathcal{L}_{\text{MNIST}} + S_{\text{FMNIST}}\mathcal{L}_{\text{FMNIST}} + S_{\text{EMNIST}}\mathcal{L}_{\text{EMNIST}}}{S_{\text{MNIST}} + S_{\text{FMNIST}} + S_{\text{EMNIST}}} \tag{61}$$

Using this setup, 25 independent repeats were taken to produce *Fig.* 3 with a schedule described by *Tab.* 9. Accuracies were still gathered for all datasets, regardless of whether the corresponding loss contributed to the total loss. A static model with identical initialisation (and the same starting random architecture $[2352, \text{H1}, \text{H2}, \text{H3}, 30]$) was used as a control across all repeats and allowed the stated adaptive-versus-control accuracies to be determined at the T3 schedule boundary.

Table 9: Task schedule used in Experiment Three.

Epoch range	Active dataset(s)
0–149	MNIST
150–299	MNIST + Fashion-MNIST
300–449	MNIST + Fashion-MNIST + EMNIST A–J
450–599	Fashion-MNIST

Fig. 18 shows an idealistic example: neurogenesis triggers for each successive dataset added, while neurodegeneration occurs when a simpler task is provided. One can observe that neurodegeneration rapidly reduces the unused dataset's accuracies after the scheduler removes them; this suggests it quickly removes the corresponding neurons for those tasks, with test accuracy falls faster than the non-neuroadaptive (fixed) counterpart.

Non-prototypical examples also occur, as shown in *Figs.* 19 and 20.

The example shown in *Fig.* 19 shows a layer-wdth rearrangement which occured around epoch 400, causing a $H2 > H3 > H1$ network to become a classical contracting $H1 > H2 > H3$ network rapidly. This could have been forced by QR-compression [38] (*App.* J), but it is especially interesting that this occurs naturally through the neuroadaptive procedure, showing how it tends toward the more classical construction when architecture flexibility is introduced. *Fig.* 20 is another interesting case; although convoluted, it can be understood as transitioning from $H3 > H2 > H1$ to a temporary $H2 > H3 > H1$ intermediate, and finally to the classical contracting $H1 > H2 > H3$. Thus, it also

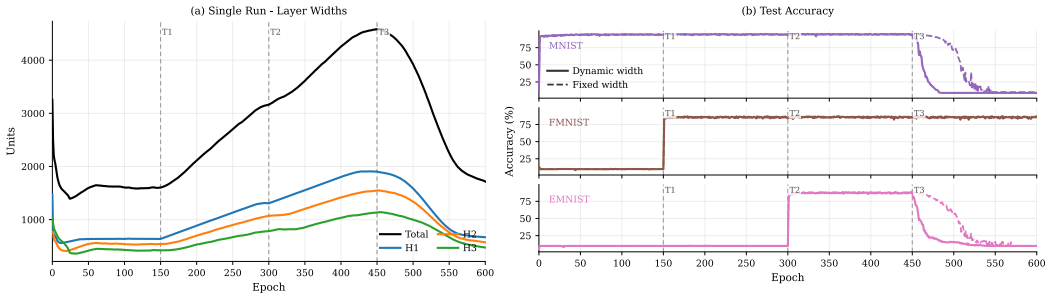


Figure 18: Experiment Three (*App.* K.3) of a single idealistic example

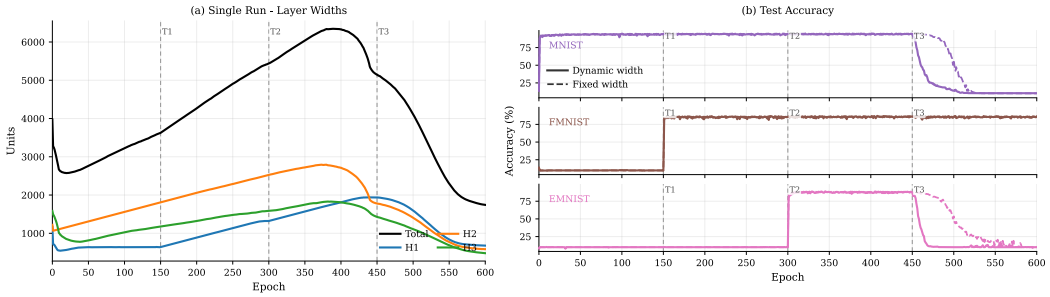


Figure 19: Experiment Three (*App.* K.3) of a single non-idealistic example

demonstrates this meta-stable architecture behaviour. Its test-accuracy response is also unusual after T3, perhaps suggesting the network's internal representations are not well-organised or separable due to the frequent and considerable architectural rearrangements in its recent history. This is suggestive that internal learned representations may have a 'memory-like' property; if its earlier optimisation trajectories became particularly unusual, this disorganisation may persist.

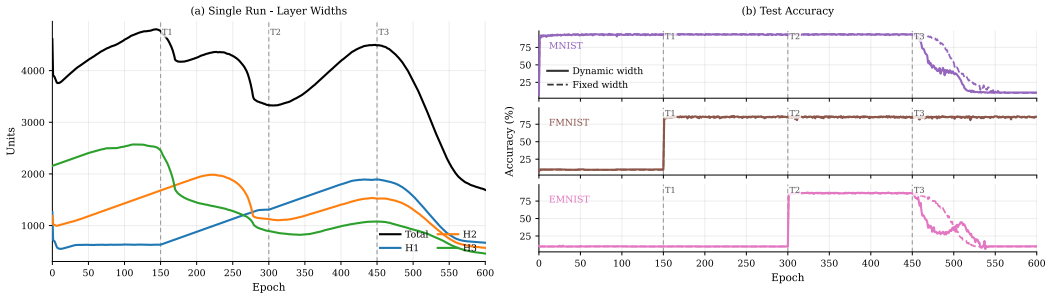


Figure 20: Experiment Three (*App.* K.3) of a single non-idealistic example

Similarly, several instances had a consistently growing $H2$ or $H3$ layer, but rarely both concurrently. This suggests that thresholding may be repeatedly triggering neurogenesis under some conditions, such as insufficiently decaying singular values or optimisation increasing the scale of singular values. This motivates exploration of more thresholding schemes, such as those discussed in *App.* C.

K.4 Experiment Four: Initial Under-to-Over Neuron Abundance

This experiment aimed to determine whether initial neuron under- or over-abundance benefits test accuracy. It did so by training the starting networks for 15 epochs at the initial width, then gradually (at a linear rate) neuroadapting for 75 epochs to the desired width, followed by a final 10 epochs of post-training at the final width. This was undertaken for fifteen repeats per permutation, with all permutations of $n = \{7, 10, 25, 50, 75, 100\}$ starting and ending widths trialled for a CIFAR-10 classification network with architecture $[3072, n, n, 10]$. This gives 540 independent networks trained for this experiment; we report the mean $\pm$ standard error. We added $n = 7$ permutations to investigate insufficient-width networks for CIFAR-10. All networks across these experiments use trainable positive intrinsic-length and trainable decaying $\vec{\psi}$ compensations. The CIFAR-10 results are displayed in *Tab.* 10.

Table 10: Experiment Four (*App.* K.4) shows CIFAR-10 test accuracy for scheduled neuroadaptation between two set widths, $m \to n$.

Start width ↓ \ End width →	7	10	25	50	75	100
7	**39.89±0.08**	41.62±0.05	43.66±0.09	**44.06±0.12**	**44.07±0.10**	**44.08±0.10**
10	39.78±0.10	**41.76±0.08**	**43.68±0.08**	43.95±0.11	43.86±0.11	44.07±0.07
25	38.85±0.11	41.21±0.09	43.59±0.06	43.64±0.10	43.84±0.05	43.71±0.11
50	38.93±0.10	40.96±0.09	43.19±0.09	43.43±0.07	43.73±0.10	43.58±0.09
75	38.89±0.12	41.16±0.10	42.96±0.09	43.68±0.05	43.50±0.10	43.68±0.11
100	38.86±0.12	40.98±0.10	43.26±0.10	43.68±0.06	43.67±0.08	43.56±0.12

The CIFAR-10 results show comparable accuracies, with a trend toward lower starting values providing an advantage in wider networks, whilst constant-width networks appear to underperform for wider networks. Furthermore, the neuro-underabundant case of $n = 7$ performs optimally on wider networks. Because accuracies are close, statistical significance based on standard errors needs examination; however, narrower starting widths collectively outperform wider networks, such as end width 100, with starting values 7 and 10 significantly different from 25, 50, 75 and 100 in this example. Two-sided Welch's t-tests with Holm correction showed neuroadaptation from width 7 outperformed constant-width training at final widths of 50, 75, and 100 (adjusted $p < 0.01$), but not 25 ($p \approx 0.52$). In particular, adapting from width 7 to 100 improved test accuracy by 0.52 percentage points over static-width training $p \approx 0.005$, Holm-adjusted. Overall, and perhaps more importantly, no disadvantage to neuroadaptation is evident for larger widths; moreover, it appears to regularly outperform the static-width counterparts. This indicates that neuroadaptation performs adequately with little functional loss or impedance to learning, and may offer a benefit for CIFAR-10's case. This neuro-underabundance success on CIFAR-10 may reflect the dataset's simplicity, where only a few primary singular values are needed to separate clusters; this trend reverses on the more complicated Caltech256.

The results for Caltech256 (*Tab.* 11) also show a deterioration in performance for static ($m = n$) networks, yet conversely tend to show an advantage to an initial abundance followed by neurodegeneration. Due to the increased 256-class dataset, the network widths were adjusted for all permutations of $n = \{32, 128, 256, 384, 512, 640\}$ for the architecture $[12288, n, n, 256]$, trained identically to the CIFAR-10 case, with 15 initial width epochs, then 75 epochs for neuroadaption, then 10 epochs for training at the final width. We also ran fifteen repeats and report the mean $\pm$ standard error. Unlike the CIFAR-10 case, which varied by at most one neuron width per epoch (except for the added 7 case at 2), these Caltech256 networks could vary by up to 9 neurons per epoch. This experiment was designed with widths of 32 and 128 estimated to be roughly underabundant, with 512 and 640 being roughly overabundant; despite this, the final widths of 32, 128 and 640 neurons reported the highest accuracies.

Table 11: Experiment Four (*App.* K.4) shows Caltech256 64×64 test accuracy for scheduled neuroadaptation between two set widths, $m \to n$. This generally indicates an adaption benefit when $m \neq n$.

Start width ↓ \ End width →	32	128	256	384	512	640
32	**12.29±0.09**	**11.76±0.07**	11.47±0.06	11.41±0.08	11.48±0.10	11.39±0.07
128	10.46±0.09	11.37±0.09	11.20±0.08	11.34±0.08	11.41±0.09	11.39±0.08
256	10.11±0.08	11.21±0.09	10.91±0.07	11.42±0.05	11.48±0.10	11.34±0.09
384	10.15±0.08	11.23±0.08	11.47±0.09	10.88±0.06	11.40±0.09	11.41±0.10
512	10.28±0.07	11.23±0.05	**11.65±0.08**	**11.48±0.05**	10.78±0.10	**11.63±0.07**
640	10.82±0.07	11.33±0.07	11.49±0.08	11.41±0.10	**11.49±0.07**	10.82±0.07

These results demonstrate a more complicated relationship between starting and ending width. For what we deemed as likely insufficient width ($n = \{32, 128\}$) constant or initially narrower networks performed best, comparable to the CIFAR-10 results. However, this drastically reversed for $n = \{256, 384, 512\}$ where increased initial widths all performed better. This neuron overabundance is more comparable to the mammalian approach to initial overabundance [9]. Two-sided Welch's t-tests with Holm correction showed that neurodegeneration from widths 512 to 256, 512 to 384, and 640 to 512 significantly outperformed their respective static network counterparts (adjusted $p < 0.001$).

Curiously, results of 256, 384, 512 and 640 all show significant underperformance when there is no neuroadaptation (start width equal to end width). Again, this result is intriguing, suggesting a benefit of forced neuroadaptation during training and reinforcing that these processes are not implicitly detrimental. The trend reverses for $n = 640$, where initially narrower networks are optimal, but this may be a data limitation since $m > 640$ is not tested and $m = n$ typically underperforms at larger widths. Overall, this is an encouraging result for neuroadaptation, particularly neurodegeneration, which is unavoidably a lossy process, as the network demonstrates resilience and even a benefit under such adaptations. The strong comparability between initially narrower or wider networks to the same end-point suggests broadly that any neuroadaptation is beneficial for wider networks.

In conclusion, this experiment was less revealing than others in terms of a clear data-agnostic trend. However, this suggests that the neuroadaptive methodology successfully preserved function path-independently, since dynamic networks achieved similar or higher performance than the fixed-width controls. This remains a positive indication of successful neuroadaptation.

K.5 Experiment Five: Sparsification

In *Sec.* 2.2.1, we detail parameter reduction using the full-diagonalisation approach. This sparsification approach is analytically function-preserving; we demonstrate this function retention numerically in this subsection alongside its parameter reductions. This complements the $k = 0$ results of *App.* K.1.

These results verify the theoretical claims, showing that a substantial parameter reduction can be achieved with negligible function loss. The derived formulae show that the reduction can asymptotically achieve 50% sparsity at infinite width and depth, while this section empirically details various finite-width and depth architectures and the corresponding practical sparsification fractions achieved.

To ensure initial functionality, these networks are trained with the aforementioned hyperparameters stated at the beginning of *App.* K. Results span CIFAR-10, CIFAR-100 and Caltech256 classification datasets over 5 independent repeats (mean and standard error reported), each trained for 25 epochs using trainable positive intrinsic length and trainable decaying $\vec{\psi}$ compensation. Good test accuracy does not matter to show sparsification and function preservation beyond a reasonable output sensitivity to the input samples. We provide a variety of width and depth architectures.

It is also important to clarify that *Sec.* 2.2.1's *Eqns.* 14 and 15 count just the weight and bias parameters; however, the networks below utilise intrinsic length and $\vec{\psi}$ additionally. Hence, we detail a slight modification to account for these additional parameters in *Eqn.* 62 for odd-length networks and *Eqn.* 63 for even-length networks. These additional parameters contribute $+N+1$ parameters per layer, which cannot be diagonalised and therefore persist in the numerator and denominator. These do not affect the stated asymptotic results and apply to constant-width networks, as $D, N \to \infty$ then $S_{\text{odd}}, S_{\text{even}} \to 0.5$.

$$S_{\text{odd}} = \frac{D\left(3N+1\right) + \left(D+1\right)\left(N^2+2N+1\right)}{\left(2D+1\right)\left(N^2+2N+1\right)} = \frac{1+D^{-1}}{2+D^{-1}} + \frac{D\left(3N+1\right)}{\left(2D+1\right)\left(N+1\right)^2} \tag{62}$$

$$S_{\text{even}} = \frac{\left(D-1\right)\left(3N+1\right) + \left(D+1\right)\left(N^2+2N+1\right)}{2D\left(N^2+2N+1\right)} = \frac{1+D^{-1}}{2} + \frac{\left(D-1\right)\left(3N+1\right)}{2D\left(N+1\right)^2} \tag{63}$$

Due to the larger initial layer for *Tab.* 12, the sparsification is typically smaller than that achieved by *Tabs.* 14 and 13. As expected, increasing width and depth increases the percentage sparsified.

Table 12: Post-training sparsification of Caltech-256 classifiers. Architectures are notated compactly as $w \times k$ for k hidden layers of width w. Δa is post-sparsification minus pre-sparsification accuracy in percentage points. Parameters count non-zero parameters, using a numerical threshold of $|\theta| > 0.0001$. ϵ_{rel} is *Eqn.* 58, and D_{KL} is *Eqn.* 59. Values reported as mean $\pm$ standard error over 5 repeats.

Architecture	Acc. before	Δa ($\times 10^{-2}$ pp)	Params before	Params after	% Reduced	ϵ_{rel} ($\times 10^{-2}\%$)	D_{KL} ($\times 10^{-8}$)
$[12288, 500 \times 2, 256]$	15.8 ± 0.1	$+0.0 \pm 0.0$	6.52M	6.26M	4.1%	0.6 ± 0.0	0.4 ± 0.1
$[12288, 500 \times 3, 256]$	15.2 ± 0.1	$+0.0 \pm 0.0$	6.77M	6.51M	3.9%	0.6 ± 0.0	0.8 ± 0.1
$[12288, 500 \times 4, 256]$	14.0 ± 0.2	-0.3 ± 0.3	7.02M	6.51M	7.3%	0.9 ± 0.1	1.5 ± 0.5
$[12288, 500 \times 5, 256]$	12.7 ± 0.1	$+0.3 \pm 0.3$	7.27M	6.76M	7.0%	0.5 ± 0.0	0.5 ± 0.1
$[12288, 500 \times 6, 256]$	11.0 ± 0.3	-0.3 ± 0.3	7.52M	6.76M	10.1%	0.9 ± 0.1	1.7 ± 0.4
$[12288, 1000 \times 2, 256]$	15.6 ± 0.2	-0.3 ± 0.3	13.54M	12.50M	7.7%	1.2 ± 0.1	1.9 ± 0.2
$[12288, 1000 \times 3, 256]$	14.9 ± 0.1	$+0.0 \pm 0.0$	14.54M	13.50M	7.1%	0.6 ± 0.1	0.8 ± 0.2
$[12288, 1000 \times 4, 256]$	13.5 ± 0.2	$+0.0 \pm 0.0$	15.54M	13.51M	13.0%	1.3 ± 0.0	2.1 ± 0.2
$[12288, 1000 \times 5, 256]$	12.6 ± 0.2	-0.3 ± 0.3	16.54M	14.52M	12.2%	1.0 ± 0.1	1.9 ± 0.4
$[12288, 1000 \times 6, 256]$	10.9 ± 0.2	$+0.3 \pm 0.3$	17.54M	14.52M	17.2%	1.3 ± 0.2	2.2 ± 0.7

Most importantly, all results verify finite-width sparsity with function invariance, demonstrating negligible change in the final-layer output activations with an order of 10^{-8} KL-divergence in output distributions. Output depth appears to amplify small numerical differences, as previously expected, but these are often insignificant given the associated errors. Function drift, measured by $\epsilon_{\text{rel.}}$, is on order of 0.01%, and test-accuracy drift is similarly around 0.01%. This is approximately 1 sample classified differently in a 10000-sized test set (the test-set size of CIFAR-10 and CIFAR-100). This is consistent with a remarkable near-identical functionality, up to numerics and propagation thereof, during significant alternating-layer sparsification. Thus, one can reasonably assert that these networks are capable of being sparsified drastically without consequence on performance; moreover, function.

Table 13: Post-training sparsification of CIFAR-100 classifiers. Architectures are notated compactly as $w \times k$ for k hidden layers of width w. Δa is post-sparsification minus pre-sparsification accuracy in percentage points. Parameters count non-zero parameters, using a numerical threshold of $|\theta| > 0.0001$. ϵ_{rel} is *Eqn.* 58, and D_{KL} is *Eqn.* 59. Values reported as mean $\pm$ standard error over 5 repeats.

Architecture	Acc. before	Δa ($\times 10^{-2}$ pp)	Params before	Params after	% Reduced	ϵ_{rel} ($\times 10^{-2}\%$)	D_{KL} ($\times 10^{-8}$)
$[3072, 500 \times 2, 100]$	19.0 ± 0.1	$+0.2 \pm 0.2$	1.84M	1.58M	13.7%	0.7 ± 0.0	0.6 ± 0.0
$[3072, 500 \times 3, 100]$	18.9 ± 0.1	-0.2 ± 0.4	2.09M	1.84M	12.1%	1.2 ± 0.1	2.5 ± 0.3
$[3072, 500 \times 4, 100]$	17.5 ± 0.2	-0.8 ± 0.6	2.34M	1.84M	21.4%	1.5 ± 0.1	3.5 ± 0.4
$[3072, 500 \times 5, 100]$	15.2 ± 0.3	$+0.4 \pm 0.4$	2.59M	2.09M	19.4%	1.3 ± 0.1	2.9 ± 0.2
$[3072, 500 \times 6, 100]$	14.0 ± 0.1	$+0.2 \pm 0.5$	2.84M	2.09M	26.4%	1.4 ± 0.1	3.2 ± 0.5
$[3072, 1000 \times 2, 100]$	18.9 ± 0.1	$+0.4 \pm 0.2$	4.17M	3.17M	24.1%	1.2 ± 0.0	1.7 ± 0.1
$[3072, 1000 \times 3, 100]$	18.4 ± 0.1	-0.2 ± 0.2	5.17M	4.17M	19.4%	1.3 ± 0.1	3.4 ± 0.8
$[3072, 1000 \times 4, 100]$	16.9 ± 0.1	$+0.6 \pm 0.5$	6.17M	4.17M	32.5%	1.9 ± 0.1	5.5 ± 0.7
$[3072, 1000 \times 5, 100]$	14.9 ± 0.2	-0.2 ± 0.4	7.17M	5.17M	27.9%	2.5 ± 0.3	12.1 ± 3.4
$[3072, 1000 \times 6, 100]$	13.9 ± 0.1	$+0.2 \pm 0.2$	8.17M	5.17M	36.7%	2.3 ± 0.2	8.6 ± 2.2

Table 14: Post-training sparsification of CIFAR-10 classifiers. Architectures are notated compactly as $w \times k$ for k hidden layers of width w. Δa is post-sparsification minus pre-sparsification accuracy in percentage points. Parameters count non-zero parameters, using a numerical threshold of $|\theta| > 0.0001$. ϵ_{rel} is *Eqn.* 58, and D_{KL} is *Eqn.* 59. Values reported as mean $\pm$ standard error over 5 repeats.

Architecture	Acc. before	Δa ($\times 10^{-2}$ pp)	Params before	Params after	% Reduced	ϵ_{rel} ($\times 10^{-2}\%$)	D_{KL} ($\times 10^{-8}$)
$[3072, 500 \times 2, 10]$	43.1 ± 0.2	-0.4 ± 0.2	1.79M	1.54M	14.0%	1.3 ± 0.2	1.3 ± 0.4
$[3072, 500 \times 3, 10]$	44.9 ± 0.2	-0.2 ± 0.4	2.04M	1.79M	12.3%	2.8 ± 0.2	6.0 ± 0.7
$[3072, 500 \times 4, 10]$	45.4 ± 0.1	$+0.0 \pm 0.3$	2.29M	1.79M	21.8%	2.3 ± 0.1	5.6 ± 0.7
$[3072, 500 \times 5, 10]$	45.1 ± 0.2	$+0.4 \pm 0.5$	2.54M	2.04M	19.7%	1.8 ± 0.2	3.3 ± 0.6
$[3072, 500 \times 6, 10]$	42.8 ± 0.3	$+0.6 \pm 0.6$	2.79M	2.04M	26.9%	2.2 ± 0.5	4.4 ± 2.3
$[3072, 1000 \times 2, 10]$	43.2 ± 0.2	-0.2 ± 0.5	4.08M	3.08M	24.6%	1.8 ± 0.2	2.2 ± 0.6
$[3072, 1000 \times 3, 10]$	44.8 ± 0.2	-0.8 ± 0.4	5.08M	4.08M	19.7%	4.4 ± 0.4	14.0 ± 2.3
$[3072, 1000 \times 4, 10]$	45.2 ± 0.1	$+0.4 \pm 0.6$	6.08M	4.08M	32.9%	3.1 ± 0.3	10.7 ± 2.1
$[3072, 1000 \times 5, 10]$	43.7 ± 0.4	$+0.2 \pm 0.5$	7.08M	5.08M	28.3%	3.0 ± 0.4	7.4 ± 1.5
$[3072, 1000 \times 6, 10]$	41.8 ± 0.1	-0.8 ± 0.2	8.08M	5.08M	37.1%	3.3 ± 0.6	7.9 ± 2.7

Overall, the networks of finite width and depth still achieve a 3.9% to 37.1% parameter reduction, without meaningful consequences. The alternating diagonalised architecture is also more computationally efficient in the forward pass, as diagonalisation yields a Hadamard product rather than full matrix multiplication. This yields $\mathcal{O}(n)$ over the standard $\mathcal{O}(mn)$ for a layer $\vec{y} = \mathbf{A}\vec{x}$, with $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\vec{x} \in \mathbb{R}^n$ and $\vec{y} \in \mathbb{R}^m$ for $n = \min(m, n)$. Thus, this technique can be used before inference to obtain a more computationally and parameter-efficient isotropic multilayer perceptron architecture, with alternating layers scaling linearly in memory and compute rather than quadratically. These non-conventional lossless pruning results are a benefit of isotropic MLPs generally, independently of neuroadaptation, and can be similarly applied to convolution networks following *App.* E.1's sketch.

K.6 Experiment Six: Isotropic-Tanh Against Standard-Tanh Trends

Already established in the previous sections (particularly *App.* K.2) is that standard-tanh, $\mathbf{f}(\vec{x}; \{\hat{e}_i\}) = \sum_i \tanh\left(\hat{e}_i^T \vec{x}\right) \hat{e}_i$, does not operate comparably to isotropic-tanh, $\mathbf{f}(\vec{x}) = \tanh\left(\|\vec{x}\|_2\right) \hat{x}$, despite their superficial resemblance and function agreement for vectors colinear with basis directions, $\hat{e}_i$.

Nevertheless, comparisons can be undertaken across various datasets, but one should not assess either's broader functional form on a single instance alone. We stress that even for superficially similar functions, they are demonstrated to be inherently dissimilar.

The results below consist of four repeats for each network-activation function permutation, across datasets of arguably increasing difficulty: MNIST, A-J EMNIST, Fashion-MNIST, CIFAR-10, CIFAR-100 and Caltech-256 — each trained for 50 epochs under otherwise identical conditions. The architecture column specifies the *fixed* (non-neuroadapted) hidden-layer width W and the number of hidden layers D ($W \times D$), with input and output dimensions constrained by their respective datasets.

Table 15: Experiment Six: displays the final test-set accuracy for standard-tanh and isotropic-tanh for otherwise identical multilayer perceptrons across various datasets and architectures. Values are reported as mean ± standard error across 5 repeats. Boxed indicates the column-wise highest accuracy (e.g., best isotropic-tanh performance across architectures, and likewise for standard-tanh). Bold indicates the pairwise highest accuracy (e.g., which was best between standard-tanh and isotropic-tanh).

Architecture	MNIST		EMNIST_AJ		FMNIST		CIFAR-10		CIFAR-100		CALTECH-256	
	Standard	Isotropic	Standard	Isotropic	Standard	Isotropic	Standard	Isotropic	Standard	Isotropic	Standard	Isotropic
500×2	**97.18±0.03**	94.38±0.04	**93.26±0.09**	86.69±0.20	**87.58±0.23**	85.69±0.14	**44.70±0.11**	44.16±0.09	**18.70±0.07**	18.60±0.11	**14.22±0.19**	12.88±0.16
500×3	**97.07±0.04**	95.28±0.11	**93.24±0.02**	87.81±0.18	**87.77±0.09**	86.13±0.16	45.19±0.13	**45.36±0.21**	18.62±0.15	**19.62±0.10**	13.45±0.16	**13.63±0.12**
500×4	**96.83±0.08**	95.25±0.09	**93.26±0.05**	88.32±0.06	**86.96±0.13**	86.04±0.25	44.83±0.18	**45.92±0.16**	17.23±0.15	**18.91±0.10**	12.63±0.21	**12.82±0.15**
500×5	**96.61±0.08**	95.37±0.07	**93.27±0.09**	89.80±0.20	**86.66±0.30**	85.77±0.32	43.49±0.28	**45.95±0.34**	15.97±0.09	**17.22±0.05**	11.66±0.38	**12.41±0.09**
1000×2	**96.98±0.05**	94.33±0.06	**93.15±0.02**	86.62±0.17	**86.66±0.27**	85.62±0.15	42.18±0.34	**44.19±0.08**	17.74±0.06	**18.68±0.10**	12.71±0.12	**12.93±0.05**
1000×3	**96.77±0.07**	95.33±0.09	**93.04±0.20**	88.19±0.10	**86.72±0.16**	86.02±0.13	42.76±0.24	**45.45±0.22**	16.68±0.10	**19.59±0.10**	12.44±0.22	**13.54±0.09**
1000×4	**96.74±0.04**	95.32±0.18	**92.61±0.16**	88.49±0.05	**86.08±0.18**	85.97±0.28	41.97±0.39	**46.02±0.12**	14.45±0.19	**18.73±0.11**	10.57±0.18	**13.07±0.07**
1000×5	**96.60±0.13**	95.15±0.12	**92.22±0.03**	89.69±0.11	85.60±0.12	**85.98±0.22**	39.56±0.25	**45.58±0.57**	12.31±0.21	**17.02±0.15**	8.40±0.53	**12.37±0.10**

The trend appears quite stable, showing a preference towards isotropic-tanh for more conventionally challenging image classification datasets, particularly for deeper MLPs — with the converse for standard-tanh. This does demonstrate that their optimal regimes are dataset and architecture-dependent; however, isotropic-tanh is a more encouraging result for broader application, demonstrating preferable scalability with challenging datasets and deeper architectures. CIFAR and CALTECH256 results demonstrate a considerable advantage typically for isotropic-tanh, whilst MNIST and EMNIST-AJ favour standard-tanh, with a narrower performance gap for Fashion-MNIST.

Further tendencies emerge: standard-tanh seems to decline considerably with increasing depth or width, a phenomenon much more suppressed in the isotropic formulation. This suggests that the isotropic-tanh may alleviate depth-related pathologies present in standard-tanh networks — perhaps related to their Jacobians (*Sec.* 2.3.1) and their respective propensities for vanishing gradients [78].

Overall, this is a promising tendency for isotropic-tanh; it is superior for deeper-wider networks, whilst standard-tanh's regime is confined to narrower-shallower networks. This reliable superiority of isotropic tanh across these harder datasets and larger architectural regimes does alleviate some concerns raised in *App.* I regarding expressibility of isotropic functions; however, this must be considered only a single instance of isotropic-tanh and not yet expanded more broadly. Similarly, this is an empirical result that does not establish theoretical universal approximation or equivalence to standard elementwise formulae. Nevertheless, this remains encouraging, considering that the standard

tanh was a popular choice historically, before the advent of more modern activation functions. Thus, development from isotropic-tanh's foundation may be undertaken to a similar effect, via a naive search over function space or theoretical developments.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract discusses several points: adaptive dense networks, a buffer of thresholded scaffold neurons, symmetry-led orthogonal isotropic activation functions, one-to-one ordered connectivity, function invariance, sparsification to asymptotic 50%, adaptive networks to changing task demands, centred on primitive symmetry-prescriptions, neuron deindividuation, intrinsic length, parallel precomputed matrix-vector products for a generalised architecture and relation to interpretability (through ICA). Thus all of which is borne out theoretically or experimentally in the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Dedicated limitations section in the appendices and main-text conclusion, covering the stated points below.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The mathematics utilised in this paper is an implementation of standard results of group theory, representation theory, tensor-rank transformations, and singular value decomposition, applied to various aspects of the network. For claims stated, assumptions are clearly specified, including isotropic activation functions for commutation and asymptotic assumptions. Due to the standard nature of group- and representation-theoretic usage, these parts do not contain formal proofs, as they are well established in their fields and are only used in novel ways for these multilayer perceptrons. All usage and statements around linear algebra transforms are derived. It conforms to standard practice regarding this style of mathematically principled implementation paper.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Yes full experimental appendix section is provided, detailing all aspects needed for successful reproducibility along with justification around the experiments. The code is also provided, at the stated GitHub link.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.

(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Yes, code is available at the stated GitHub link and all datasets are established publically-accessible datasets.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Please see appendix detailing all experimental setup.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: All plots and tabulated data include error bars. Exception being the ensemble ridgeline plot of experiment one, which cannot display errors by its nature, but is still an aggregate over many independent repeats.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Time of execution was explicitly stated with errors, alongside relevant computing hardware.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes, it conforms.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This is an adaptive architecture methodology, and a group/representation-theoretic approach to network primitives. It is evaluated on simple image-classification settings, and is not introduced as a deployed system, generative model, decision system etc. There is no direct pathway to societal harm.

    Neither of which constitutes a broader societal impact.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Models are simple image classifiers.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Yes all datasets are properly cited and code is personally written by the authors.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: No new assets.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: No crowdsourcing.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: No human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLM usage declared. Used in grammatical/synonym/phrasing finding capacity for the text, and used in some aspects of the code to improve the presentation of plots using matplotlib in a way described by the authors. Similarly, some code helper functions were written by LLM to save time but were reviewed carefully.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.