TO GATE OR NOT TO GATE? WHEN DOES AN INPUT-DEPENDENT GATING PAY OFF IN MULTI-CLASS DENDRITIC GATED NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Dendritic Gated Networks (DGNs) consist of neurons that each have several weight vectors (their branches), each paired with a fixed random hyperplane. The input, together with the hyperplanes, determines which branches are active, contribute to the output and get updated. This mechanism makes DGNs resistant to catastrophic forgetting, able to approximate arbitrary nonlinear functions while being convex in their weights, and trainable without backpropagation. DGNs are binary classifiers, and existing multi-class extensions support K classes one of three ways: training one full network per class (one-versus-all), training $\lceil \log_2 K \rceil$ networks over an efficient binary code, or applying a shared gated expansion, in which one fixed set of random half-space gates is applied once per input, the per-branch blocks are concatenated into a single feature vector, and K independent Bernoulli readouts read that vector. We propose training those K readouts jointly by softmax regression: the learning objective stays jointly convex, the update stays local up to one shared normalizer per sample, and the design keeps the expansion's one to two orders of magnitude fewer parameters than the one-versus-all formulation, and its single expansion in place of the K forward passes, which is what makes $K{=}100$ tractable. We then investigate when using input-dependent gates pays off, against controls matched in dimension, parameter count, update rule and training protocol. Under domain-incremental training with smooth image rotation ($K{=}10$ and $K{=}100$) and pixel noise ($K{=}100$), our proposed method outperforms the dense, static-gate and linear controls on all-task and first-task accuracy. Given the same frozen features, it also outperforms the training-free nearest-class-mean classifier, 0.50 and 0.19 against 0.34 and 0.14 over all tasks under rotation and noise. With a replay buffer on split-CIFAR-100 it also outperforms a backpropagation-trained network with Dark Experience Replay++ (DER++) at every buffer size we ran, 0.60 against 0.53 at $5{,}000$ rows. We attribute the gating margin to regime structure, where the input-label mapping differs between regimes that can be identified directly from the input itself. In a class-incremental problem, which has no such regime structure, our method's margin over its static-gate control shrinks to the margin it already has under i.i.d. training, as predicted. It still outperforms one-versus-all and, storing no past data, reaches 0.47 against 0.27 for a backpropagation-trained network with Learning without Forgetting, but an untrained dense random projection of the same width matches it and the nearest-class-mean classifier leads it (0.53).

1 INTRODUCTION

A Dendritic Gated Network (DGN) (Sezener et al., 2021), a descendant of Gated Linear Networks (GLNs) (Veness et al., 2020), is a deep learning architecture that learns on a convex objective, can represent arbitrary nonlinear functions, and is naturally resistant to catastrophic forgetting. It learns via a local delta rule, which avoids the weight transport that backpropagation requires, making it a viable implementation of biologically plausible learning. In a DGN, each neuron comprises several weight vectors, its branches, where every branch has its own fixed hyperplane over the network input. Depending on which side of the hyperplane the input falls, the branch is either active or

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

inactive. The hyperplanes are drawn at random and never trained, and since an input activates only the branches whose hyperplanes it falls on the positive side of, learning in one region of input space leaves the weights serving other regions untouched, the source of the resistance to forgetting.

One limitation DGNs face is the lack of a shared representation. A K-class problem is framed as one-versus-all (OvA), K independent binary networks (Sezener et al., 2021), so the parameter count and the number of forward passes grow linearly in K. Shared Gated Expansions (SGEs) (Quah, 2026) share one fixed representation across the classes while keeping the convex objective, an update that is backpropagation-free, and the resistance to forgetting: a single fixed gated expansion is computed once per input, its per-branch blocks are concatenated into one feature vector, and each class keeps its own linear readout on that vector, training each of the K as independent Bernoulli units. We propose training them jointly by softmax regression (SGE-softmax), which keeps the expansion's parameter count (one to two orders of magnitude below OvA's), its single expansion in place of K forward passes, and its resistance to catastrophic forgetting. SGE-softmax maintains the biological plausibility property from DGNs: the softmax rule adds one normalizing scalar per sample across the K readouts, a divisive normalization documented across many brain regions (Carandini & Heeger, 2011).

DGNs, and the shared expansion, belong to a family of gating approaches for continual learning. Prior analysis of GLNs found them sensitive to data standardization and to tailoring the hyperparameters to the data distribution (Munari et al., 2022), which our centring result echoes. Other methods gate on a task signal: XdG assigns each task a random subnetwork determined by the task label, and active-dendrite networks use a per-task context vector to modulate each unit (Masse et al., 2018; Iyer et al., 2022). Our gates are fixed random functions of the input alone and receive no task label. We compare our method against regularization, distillation and replay methods (Kirkpatrick et al., 2017), under the class-incremental and domain-incremental scenarios of van de Ven & Tolias (2019). Quah (2026) introduced the expansion together with Definition 1 and reported that at $K{=}100$ input-dependent gates no longer beat a static-gate control. Under the protocol here (Appendix D lists what changed) they do, wherever the drift has regime structure.

We measure SGE-softmax against matched controls under two sequential protocols, a protocol being a dataset cut into tasks that the model trains on one after another, and Definition 1 says in advance which of them should show a gating margin, the lead of input-dependent gates over the static-gate control. Under domain-incremental training, where every task holds every class and the inputs drift, SGE-softmax beats the dense projection, the static-gate control, the linear readout and the prototype classifier on both all-task and first-task accuracy, at $K{=}10$ and at $K{=}100$. Under class-incremental training, where tasks differ only in which classes are present, it beats OvA at $K{=}10$ and the best rehearsal-free backpropagation method by 20 points at $K{=}100$, and with a matched buffer it leads DER++ at every buffer size we ran; but a dense random projection of the same width matches it once trained long enough, and the prototype classifier leads it by 6 points. A third finding holds under both protocols: per-pixel centring of the inputs decides which readout rule wins on split-CIFAR-100 and doubles the gated margin at $K{=}10$.

2 Dendritic Gated Networks (DGNs) and Shared Gated Expansions (SGEs)

2.1 The DGN mechanism

Given input $\mathbf{x} \in \mathbb{R}^n$, let $\tilde{\mathbf{x}} = [1, \mathbf{x}]^\top$ be the same vector with a leading bias entry. Neuron i of layer m has B dendritic branches, each with a fixed random hyperplane $(\mathbf{v}^b_{m,i}, \theta^b_{m,i})$ and a learnable weight vector $\mathbf{w}^b_{m,i}$, and branch b is active when its binary gating variable is one,

$$g^b_{m,i}(\mathbf{x}) = \mathbf{1}\left[\mathbf{v}^{b\top}_{m,i}\tilde{\mathbf{x}} > \theta^b_{m,i}\right], \tag{1}$$

with $\mathbf{1}[\cdot]$ the indicator function; every branch at every depth gates on the network input $\mathbf{x}$, so the partition of input space is fixed before training. Collecting across the branches gives the neuron's input,

$$\Phi_{m,i}(\mathbf{x}) = \left[\, g^b_{m,i}(\mathbf{x})\, \mathbf{h}_{m-1} \,\right]^B_{b=1}, \tag{2}$$

where $\mathbf{h}_{m-1} = [1, h_{m-1,1}, \ldots, h_{m-1,n_{m-1}}]^\top$ holds the synaptic drives of the n_{m-1} neurons below ($\mathbf{h}_0 = \tilde{\mathbf{x}}$). With the branch weights stacked in the same order, $\mathbf{w}_{m,i} = [\,\mathbf{w}^b_{m,i}\,]^B_{b=1}$, the neuron's

Table 1: The three multi-class DGN formulations: learnable parameters, their totals at MNIST's n=784 with K=10 and at the 512-dimensional CIFAR-100 features with K=100, networks evaluated per input, and measured time per epoch over 4,000 MNIST rows on one GPU. OvA and binary-code networks are $[n_1, n_2, n_3] = [128, 64, 1]$ with $B_{\text{OvA}} = 20$; the expansion has $B = 60$.

Formulation	parameters	MNIST	CIFAR-100	passes	s / epoch
One-vs-all	$K\,B_{\text{OvA}}[\,n_1(n{+}1) + n_2(n_1{+}1) + n_3(n_2{+}1)\,]$	21,760,200	147,970,000	K	0.12
Binary code	$\lceil \log_2 K \rceil\, B_{\text{OvA}}[\cdots]$	8,704,080	10,357,900	$\lceil \log_2 K \rceil$	0.05
SGE, SGE-softmax	$K\,B\,(n{+}1)$	471,000	3,078,000	1	0.001

own drive is $h_{m,i} = \mathbf{w}_{m,i}^{\top}\,\Phi_{m,i}(\mathbf{x})$ and its rate $r_{m,i} = \phi(h_{m,i})$, with ϕ the identity for regression and the logistic sigmoid for classification.

2.2 Training a DGN

Every neuron at every depth is trained against the network's target r^*, so there is no backward pass and no weight transport. Under quadratic loss with identity ϕ and under Bernoulli log loss with sigmoid ϕ, the derivative of the loss with respect to the neuron's own drive is $r_{m,i} - r^*$, so both have the same update,

$$\Delta \mathbf{w}_{m,i} = -\eta\,(r_{m,i} - r^*)\,\Phi_{m,i}(\mathbf{x}), \tag{3}$$

with η the learning rate.

2.3 The SGE Mechanism

The SGE of Quah (2026) takes Eq. 2 at a single layer, drops the neuron index from the gates so that one set of B hyperplanes serves every class, and reads the result with K linear readouts:

$$\Phi(\mathbf{x}) = \big[\, g_b(\mathbf{x})\,\tilde{\mathbf{x}} \,\big]_{b=1}^{B} \in \mathbb{R}^{B(n+1)}, \qquad \ell_k = \mathbf{w}_k^{\top}\Phi(\mathbf{x}), \qquad p_k = \frac{e^{\ell_k}}{\sum_j e^{\ell_j}}. \tag{4}$$

The blocks are concatenated, not summed, so the vector records which gates fired and therefore which weights should be updated. Our proposal is to train the K readouts jointly by softmax cross-entropy,

$$\Delta \mathbf{w}_k = -\eta\,(p_k - \mathbf{1}[y{=}k])\,\Phi(\mathbf{x}), \tag{5}$$

which is multinomial logistic regression on the SGE's fixed feature map; we call it SGE-softmax. Quah (2026) trains the same expansion with K independent Bernoulli readouts under Eq. 3. The two differ in locality: SGE is fully local, its readouts never interact, while each SGE-softmax prediction is normalised against the other $K{-}1$ through $\sum_j e^{\ell_j}$. The update rule in both is still prediction minus target multiplied by $\Phi(\mathbf{x})$.

Table 1 compares the three formulations on parameter count, networks evaluated per input and measured time per epoch.

Proposition 1. *If Φ is any fixed map that does not depend on $\mathbf{w}$, whether or not it depends on $\mathbf{x}$, softmax cross-entropy on Φ is jointly convex in all K readouts.*

Proof. Composed with the linear map $\mathbf{w} \mapsto (\mathbf{w}_k^{\top}\Phi)_k$, softmax cross-entropy is the multinomial logistic objective, whose Hessian in the logits is a covariance matrix and hence positive semidefinite. □

2.4 Controls and baselines

Between Proposition 1 here and Proposition 1 of Quah (2026), every arm below except the backpropagation-trained trunk trains a convex objective; the trunk's online-trained nonlinear representation makes Φ depend on $\mathbf{w}$.

Controls along feature dimension, nonlinearity, and the readout rule allow us to identify if any accuracy improvements SGE-softmax achieves can be attributed to input-dependent gating and the softmax formulation. Every fixed-expansion control is trained by the same readout rule, softmax or Bernoulli, as the arm it is matched to, through the same protocol, on the same learning-rate grid.

Static gates. The expansion with one random draw of $g \in \{0,1\}^B$ broadcast to every input, so each g_b is a fixed 0 or 1 for the whole run, as in Quah (2026). Feature dimension, gate marginal and readouts are identical to SGE-softmax, or to SGE under the Bernoulli rule; only the dependence on $\mathbf{x}$ is removed. Let S be the set of active blocks, then $|S| \approx B/2$ and the expansion is $|S|$ identical copies of $\tilde{\mathbf{x}}$; since every copy receives the same update from zero initialisation the arm is exactly one linear model on $\tilde{\mathbf{x}}$ at effective rate $|S|\eta$. This allows us to investigate if conditioning on the input data is necessary as a part of the gating process.

Dense random ReLU projection. A dense random projection to the expansion's dimension $B(n{+}1)$ followed by a rectifier, drawn once and frozen. Nonlinear and equally wide. It investigates if any nonlinear expansion of this dimensionality would suffice.

Linear readout. K readouts on $\tilde{\mathbf{x}}$ itself, no expansion, trained by the softmax rule: the floor that shows how much forgetting the protocol induces in the simplest standard model, and the model the static-gate control reduces to under the softmax rule.

Prototype classifier. Nearest-class-mean on the same inputs the readouts see: class means accumulated from the training rows, with a test row taking the class of the nearest mean by cosine similarity or Euclidean distance, whichever scores higher in a condition (both in Appendix F). This method stores K vectors and no data, has no training step, and has no training order. On frozen pretrained features this method has been shown to be a strong class-incremental baseline (Janson et al., 2023), so it measures how much of any class-incremental result the frozen representation provides on its own. Under drift its means average over every regime, one mean per class whatever the rotation, so it cannot exploit regime structure and is the natural contrast for Definition 1.

One-versus-all. The prior multi-class DGN (Sezener et al., 2021): K independent DGNs of shape $[128, 64, 1]$ with 20 branches per neuron, network k trained on $\mathbf{1}[y{=}k]$ under Bernoulli log loss, prediction by $\arg\max_k$. Under class-incremental training we report it both updating every network on every task (absent classes as negatives, the standard OvA protocol) and updating only the present classes' networks. It runs at $K{=}10$ only: extrapolating the per-epoch times from Table 1 to CIFAR-100 gives about a day of GPU for the three protocols against minutes for the SGEs.

Binary code. The cheaper coding Sezener et al. (2021) suggest: $\lceil \log_2 K \rceil$ networks of the same shape, network k trained on bit k of the class index, where the prediction is the codeword with the highest Bernoulli log-likelihood under the sigmoid outputs; four networks at $K{=}10$, seven at $K{=}100$, where its $40{,}000$-row full-batch step does not fit in GPU memory and each epoch is eight steps over disjoint $4{,}000$-row chunks. Every network is updated on every row of every task, since a bit network has no absent class.

Backpropagation-trained trunk and continual-learning methods. A DGN trunk of shape $[128, 64]$ feeding K softmax heads, trained end to end by backpropagation in minibatches of 64 rows: the standard deep-learning route to a shared representation for multi-class classification, and the architecture on which we run the standard continual-learning methods: Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017) in its online form, experience replay from a reservoir buffer (Vitter, 1985, algorithm R), Learning without Forgetting (LwF) (Li & Hoiem, 2017) with $T{=}2$ distillation (Hinton et al., 2015), and Dark Experience Replay++ (DER++) (Buzzega et al., 2020), each with its own hyperparameter swept and reported per value (mechanics and grids in Appendix E). Our DER++ deviates from the reference implementation in one respect: rows enter the reservoir once per task with end-of-task logits, not per minibatch with the current logits, so the buffer capacity stays comparable with the replay arm. SGE-softmax and SGE with replay train on the current task's rows with the same reservoir buffer, so the two families are comparable in terms of storage size; as in the trunk's replay, each step draws as many buffered rows as it has current rows, here the whole task, so the buffer carries half of every update whatever its size.

3 WHEN SHOULD GATING HELP?

Definition 1 (Regime structure, after Quah, 2026)**.** *An input distribution has* regime structure *when it decomposes into identifiable modes, the correct input-output mapping differs between modes, and the active mode is recoverable from the input itself.*

From it Quah (2026) predicts that the margin of a gated arm over an input-independent control is large under regime structure and, without it, no larger than the expressiveness a piecewise-linear readout adds to a linear one, which an i.i.d. run measures. The definition is binary, so the prediction is a contrast between conditions that have regime structure and conditions that do not.

Rotation meets all three conditions: every task holds every class, a rotated image needs different readout weights from an upright one, and the angle is visible in the input. Class-incremental training fails the second: the image-to-label mapping never changes, only the set of labels present does, so the definition predicts no gating margin there beyond the expressiveness gap. Noise depends on where the gates read. Additive pixel noise leaves the pixel-to-label mapping unchanged, so noised MNIST, where the gates read pixels, fails the second condition and is our negative control. On CIFAR-100 the gates read frozen ResNet-18 features, which the pixel noise moves, and Table 11 (Appendix F) measures how far: a linear readout fitted at $\sigma{=}0.1$ beats the clean-level readout by 16 points on that level, the same kind of gap as the 11 to 22 points along rotation, so the first noise level is a regime change in the representation the gates read. From $\sigma{=}0.2$ a readout fitted at the level itself reaches only 0.06 to 0.12: those levels have no mapping left to relearn, so they test feature destruction rather than regime structure.

4 EXPERIMENTAL SETUP

Protocols. In **class-incremental** training the same model is trained sequentially on tasks with disjoint label sets and evaluated K-way over every class at the end. Split-MNIST (Lecun et al., 1998) has five tasks of two classes, $y \in \{2t, 2t{+}1\}$; split-CIFAR-100 (Krizhevsky, 2009) has ten tasks of ten classes in natural label order, class-balanced, on frozen ImageNet-1k ResNet-18 features (He et al., 2015) of dimension 512. The fixed-expansion arms update only the present classes' readouts on each task; under the softmax rule the absent classes' logits still enter the normalizer, so the mask decides which rows are written, not what the present rows' gradient sees. In **domain-incremental** training every task holds the whole labelled dataset under a different input transform. **Rotation** by $\{0°, 15°, 30°, 45°, 60°\}$ runs on MNIST ($K{=}10$) and on CIFAR-100 ($K{=}100$), the CIFAR images rotated before feature extraction. **Noise**, additive and Gaussian with $\sigma_t \in \{0, 0.1, 0.2, 0.3, 0.4\}$ on pixels in $[0, 1]$, runs on CIFAR-100 and, as the pixel-space negative control, on MNIST (noised MNIST).

Metrics. *All tasks* is accuracy after the full sequence: the single K-way test set under class-incremental training, the mean over the drifted test sets under domain-incremental. *First task* is accuracy on the task trained first and never revisited, the forgetting measure. Wherever a pair is written a / b below, a is all-tasks accuracy and b is first-task accuracy. Every dataset is near-balanced, so chance is 0.10 at $K{=}10$ and 0.01 at $K{=}100$.

Data, splits and selection. MNIST is subsampled to 5,000 training and 1,000 test rows; CIFAR-100 uses its full 50,000/10,000 split. In every case 20% of the training rows, the same rows for every task and every arm, are held out for validation (stratified for split-CIFAR-100). Each arm's learning rate is the one with the best mean validation accuracy after the full sequence; the arm is then rerun at that rate on the untouched test set, training on the remaining 80% throughout. Grids step by factors of $\sqrt{10}$ (one-vs-all's by factors of 10, five points) and were widened until every arm's optimum was located within the grid; the table entries where one still sits on a grid edge are marked. The expansion's hyperplanes or projection and the initial weights of the OvA networks and the trunk were drawn across the seeds, while the validation split and the MNIST subsample were fixed across them. Five seeds were run throughout and standard deviations are below 0.01 for every fixed-expansion cell quoted and specified where larger.

Table 2: Domain-incremental accuracy, all tasks / first task: CIFAR-100 (pretrained features) at 40 epochs per task, MNIST at 100. Standard deviations are at most 0.012 except the SGE first task on rotated MNIST (±0.021). Bold: best arm per column.

Arm	CIFAR-100 rotation K=100	CIFAR-100 noise K=100	rotated MNIST K=10	noised MNIST K=10
SGE-softmax	**.500 / .596**	**.189 / .530**	**.828 / .660**	**.911 / .932**
SGE (Bernoulli readouts)	.497 / .591	.182 / .510	.796 / .559	.908 / .930
control: static gates, Bernoulli readouts	.471 / .555	.154 / .411	.712 / .480	.855 / .880
control: static gates, softmax readouts	.473 / .561	.177 / .452	–	–
control: dense random ReLU	.413 / .494	.125 / .284	.769 / .632	.887 / .896
control: linear readout	= static gates, softmax readouts		.764 / .596	.882 / .896
Prototype classifier (nearest class mean)	.335 / .408	.141 / .337	.734 / .654	.817 / .822
One-vs-all	–	–	.803 / .581	.895 / .901
Binary code ($\lceil \log_2 K \rceil$ networks)	.137 / .177	.029 / .071	.632 / .286	.813 / .866

Budgets, in epochs and in steps. An epoch of a fixed-expansion arm is one full-batch step; the trunk takes 63 minibatches per epoch on each split-CIFAR-100 task (4,000 training rows). We quote K=10 at 100 epochs per task and K=100 at 40, the budgets at which every quoted arm was run; Appendix A gives the same arms at 10, 40, 100 and 200 epochs.

Preprocessing. Every input is centred coordinate by coordinate: the mean of each pixel, or of each CIFAR-100 feature, over the training rows (the undrifted task's under drift, so that the same shift applies at every level; the whole training set under class-incremental training) is subtracted from every task, and MNIST is then divided by one global standard deviation. The spread stays a single scalar because over a hundred MNIST border pixels have zero training variance, so per-pixel standardisation would divide those pixels by zero (Quah, 2026). The centre is per pixel because of the gate equation: a random hyperplane through the origin splits a cloud evenly only if the cloud straddles it, and Section 6 gives the firing rates with and without centring. Every number in Section 5 is on centred inputs, with one exception noted in the caption of Table 3; Table 10 (Appendix D) repeats every arm uncentred.

5 Results

5.1 Domain-incremental

Table 2 gives the domain-incremental results. At K=100 SGE-softmax leads every control and the dense projection performs the worst. Under rotation SGE-softmax scores 0.500 / 0.596 against 0.473 / 0.561 for its static-softmax control and 0.413 / 0.494 for the dense ReLU projection: margins of 2.7 / 3.5 and 8.7 / 10.2. The dense projection is still improving at forty epochs, and at a hundred the margin over it narrows to 5.3 / 6.0 (Appendix A). Under noise, where all-tasks accuracy sits near 0.2 for every arm, the margins are 1.2 / 7.8 over the static-softmax control and 6.4 / 24.6 over the dense projection. SGE is within two points of SGE-softmax on both drifts and leads its own static control by 2.6 / 3.6 and 2.8 / 9.9. Margins over the static control either stay constant or increase from 10 to 100 epochs (Appendix A). The prototype classifier pools its class means across the five levels and falls to 0.335 / 0.408 and 0.141 / 0.337. The seven-network binary code, updated on every row of every level, reaches 0.137 / 0.177 and 0.029 / 0.071.

On rotated MNIST SGE-softmax leads every matched control on both measures: the static-gate control by 11.6 points over all tasks and 18.0 on the first, the dense ReLU projection by 5.9 and 2.8, the linear readout by 6.4 and 6.4, and the prototype classifier by 9.4 and 0.6. One-vs-all comes within 2.5 points over all tasks and trails by 7.9 on the first; the four-network binary code loses 17 points to one-vs-all over all tasks and 30 on the first, and 8 and 4 under noise. SGE trails SGE-softmax by 3.2 and 10.1.

Noised MNIST is the negative control. Additive pixel noise leaves the pixel-to-label mapping alone, so Definition 1 predicts no gating margin beyond the expansion's expressiveness gap over a linear

Table 3: Class-incremental accuracy, final K-way: split-CIFAR-100 at 40 epochs per task, split-MNIST at 100. The fixed-expansion arms update the present classes' readouts only; one-vs-all is shown under both policies. *: rate at a grid edge. Standard deviations are at most 0.005 except where given. The split-CIFAR-100 prototype cell is cosine on the features as extracted, the higher of its settings (.520 centred, Appendix F). Bold: best arm storing no data, and best with a buffer.

	Arm	split-CIFAR-100 K=100	split-MNIST K=10
fixed expansion	SGE-softmax	.468	**.858**
	SGE (Bernoulli readouts)	.425	.752
	control: static gates, Bernoulli readouts	.316	.665
	control: static gates, softmax readouts	.449	–
	control: dense random ReLU	.444	.685
	control: linear readout	.451	.823
	Prototype classifier (nearest class mean)	**.526**	.824
	One-vs-all, present classes	–	.531 ± .011*
	One-vs-all, all classes	–	.708*
	Binary code ($\lceil \log_2 K \rceil$ networks)	.098	.648
backprop	Softmax trunk	.099	–
	+ EWC (best λ)	.115	–
	+ LwF (best weight)	.270	–
	+ replay, buffer 5,000	.480	–
	+ DER++, buffer 5,000	.528	–
	SGE-softmax + replay, buffer 5,000	**.596**	.931
	SGE + replay, buffer 5,000	.560	**.936**

readout. That gap is measured i.i.d. on clean MNIST: SGE-softmax beats the linear readout by 3.8 points and SGE its static control by 4.8. Under noise the margins are 2.9 / 3.6 and 5.3 / 5.0, the size of the gap; under rotation they exceed it by 2.6 / 2.6 and 3.6 / 3.1. The task-identity oracle (Appendix D) selects the gate pattern from the task label, so it partitions by task under both drifts. Under rotation it keeps 5.5 points more of the first task than SGE (0.614 against 0.559); under noise it keeps 5.3 fewer (0.877 against 0.930).

5.2 Class-incremental

Table 3 gives the class-incremental results. At K=100 SGE-softmax reaches 0.468; the static-gate control under the same softmax rule reaches 0.449, the linear readout 0.451, and the dense random ReLU projection 0.444 at forty epochs and 0.496 at two hundred, where it is still improving and the gated arm has stopped (0.471). SGE leads its static control by 10.9 points (0.425 against 0.316) but sits 2 points below the dense projection. So at K=100 SGE-softmax, its static control, the linear readout and the dense projection all land between 0.44 and 0.50 storing no data, the gated one is not the best of them, and the prototype classifier, which trains nothing, reaches 0.526, above all of them. SGE-softmax's 1.9-point margin over its static control is about the size of its i.i.d. advantage, the 0.7 points by which it beats the linear readout when trained jointly on all hundred classes (Appendix F), which is all Section 3 predicts without regime structure. SGE's 10.9 does not fit that account, and Section 6 returns to it. The seven-network binary code keeps only its last task, 0.098 over all classes with 0.76 on the final task and at most 0.12 on any earlier one. The backpropagation-trained trunk reaches 0.099 on its own, 0.115 with EWC and 0.270 with LwF, the best rehearsal-free backpropagation baseline; four times the epoch budget moves these three by at most 1.3 points (Appendix A). With a 5,000-row buffer the trunk reaches 0.480 under plain replay and 0.528 under DER++, and SGE-softmax 0.596 (SGE 0.560); SGE-softmax leads DER++ at every buffer size we ran (Appendix C).

At K=10 SGE-softmax scores 0.858 against one-vs-all's 0.708 under the standard all-classes policy and 0.531 updating present classes only, and against the four-network binary code's 0.648. The linear readout on raw pixels is the nearest arm at 0.823 and the prototype classifier sits at 0.824, so

most of that accuracy is the fixed-representation-plus-local-readout design and the expansion adds 3.5 points over it on this data split (2.8 ± 1.0 over five, Appendix B); the static-gate and dense ReLU controls, both with Bernoulli readouts, trail by 19 and 17 points, and SGE by 10.6, so SGE leads its own static control by 8.7 (buffers: Appendix C).

Figure 1 (Appendix A) gives per-task accuracy after the sequence. SGE-softmax's retention profile lands wherever rate times budget puts it (at forty epochs it holds 13 points more of the first task than the linear readout, at two hundred the two profiles cross), since validation selects on mean accuracy, which is blind to a redistribution that sums to zero; SGE's profile shows no first-to-last trend at either budget, per-class isolation leaving nothing to overwrite, at a cost of 4.3 points of mean accuracy.

5.3 Branch count

Table 7 (Appendix A) varies B over a sixteen-fold range. The static control is flat in B to within 0.003 under both readout rules, as it must be, being one linear model whatever B (the static-gate paragraph of Section 2.4), and the gated arm rises monotonically, so the gating margin grows with B: from $+0.7$ to $+2.3$ for SGE-softmax and from $+7.2$ to $+12.1$ for SGE. At ten epochs the softmax margin is at most $+0.3$ at any B: the expansion needs a real budget before it can use its extra branches.

6 Discussion

The SGE keeps the DGN's appealing properties, a convex objective, an update with no backward pass and built-in resistance to forgetting, while requiring one network in place of K. Two of its advantages, the single expansion per input and the order-of-magnitude reduction in parameters, follow from the design (Table 1). The third, accuracy, comes from SGE-softmax: at $K{=}10$ it beats one-vs-all and the binary code under every protocol, and at $K{=}100$, where one-vs-all would take about a day of GPU across the three protocols and the binary code keeps only its last task, it is the strongest trained arm storing no data at the quoted budget and the strongest arm of any kind with a buffer.

Under class-incremental training the gating only leads to marginal gains. The static control, the linear readout and the dense random projection land within two points of SGE-softmax at $K{=}100$ with the same convexity, locality and fixed representation, and the dense projection overtakes it given budget. This agrees with two published results on frozen pretrained features, a random projection under a convex readout, RanPAC (McDonnell et al., 2024), and the prototype classifier (Janson et al., 2023): the rehearsal-free margin over the backpropagation baselines belongs to reading the frozen features with a convex readout, and the prototype classifier, which trains nothing, beats every convex readout we ran. This observation is unsurprising, considering Definition 1 , since the image-to-label mapping does not change between tasks, and SGE-softmax's margin over its static control is its i.i.d. expressiveness gap on both datasets.

Under drift the gating pays off at both dataset. SGE-softmax beats the static control under the same rule in every drift condition and the dense projection on both measures under all four drifts, and the prototype classifier, whose one mean per class averages over the regimes, falls far below it. Noised MNIST separates exploiting regime structure from plain expressiveness: the pixel-to-label mapping does not move, the margin there is the i.i.d. gap, and rotation's exceeds it; the task-identity oracle's first-task lead over SGE flips sign between the two drifts. Under regime structure, the gates of an input from a drifted task activates different branches and writes to different weights, which our dense nonlinearity of the samd width does not. Thus, the boundary of where gating pays off is regime structure. Quah (2026) placed it at $K{=}100$ on the strength of a tie under the CIFAR-100 rotation drift but the tie was the result of their preprocessing, and it disappears once the inputs are centred (Table 10).

The two readout rules differ most where the gating matters least. On centred inputs SGE-softmax wins every all-tasks comparison against SGE, at $K{=}10$ by 3 to 11 points and at $K{=}100$ by 4 under class-incremental training, and the two are within a point under drift. The softmax rule costs one population-level scalar per sample; the Bernoulli rule keeps per-class isolation, which is why SGE's retention profile is flat across tasks where SGE-softmax's is not (Figure 1).

Table 10 (Appendix D) is the basis for the preprocessing claim. At $K{=}100$ centring decides which readout rule is usable: on split-CIFAR-100 the two softmax arms, SGE-softmax and the linear readout, go from below 0.2 to above 0.45, while SGE and its static control lose 8 to 15 points; under the CIFAR-100 rotation drift every arm gains and the SGE-minus-static tie becomes a margin. The backpropagation baselines gain too, DER++ by 5 points and LwF by 4.5, which is why every $K{=}100$ comparison has both sides centred. At $K{=}10$ centring costs the static control 3 to 9 points and the gated arms nothing, which doubles the SGE-minus-static margin on rotated MNIST (from $+4.3$ / $+4.4$ to $+9.3$ / $+12.7$). The mechanism is the gate equation: a hyperplane through the origin splits a cloud evenly only if the cloud straddles it, and on the uncentred CIFAR-100 features the per-branch firing rates of the $B{=}60$ gates ran from 0.010 to 0.998 (mean 0.436), against 0.446 to 0.590 (mean 0.507) centred.

Limitations. Every number is from five seeds on one desktop GPU. The backpropagation baselines are reported at $K{=}100$ only. No trunk baseline was run on the $K{=}100$ drift protocols. Every drift is synthetic, and noise has measurable regime structure only at its first level. Replay from the smallest buffers scores below no replay for both expansion arms (Appendix C), which we have not diagnosed. The $K{=}100$ results concern frozen ImageNet features rather than raw pixels. One-vs-all is absent at $K{=}100$ for wall-clock reasons.

AI USE STATEMENT

In this work, we used generative AI tools for literature search, for revising the manuscript's phrasing and punctuation, for writing experiment and analysis code (including the ImageNet feature-extraction script and a memory optimisation of the expansion), and for drafting the baseline, control and appendix passages of this manuscript that report the continual-learning and prototype baselines. We have reviewed all AI-assisted work: every experimental design, result, claim and reference was verified against the committed result files by the author, and all AI-generated code was tested. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Every experiment runs on public benchmark datasets (MNIST, CIFAR-100) and a publicly released ImageNet backbone, on one desktop GPU; no human subjects, personal data or deployed system is involved. We are aware of no concern under the Code of Ethics.

REPRODUCIBILITY STATEMENT

The expansion, the readout rules and the continual-learning baselines are implemented in a pure-JAX package; the four protocols are single scripts (class-incremental, split-CIFAR-100, domain-incremental, CIFAR-100 drift) whose command lines, grids and seeds are recorded alongside every result file. The per-pixel centring, the validation-selection procedure and the epoch and step budgets are as stated in Section 4. Feature extraction for CIFAR-100 records the torch and torchvision versions used. Code and result files will be released with the camera-ready version.

REFERENCES

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark Experience for General Continual Learning: A Strong, Simple Baseline, October 2020.

Matteo Carandini and David J. Heeger. Normalization as a canonical neural computation. *Nature reviews. Neuroscience*, 13(1):51–62, November 2011. ISSN 1471-003X. doi: 10.1038/nrn3136.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition, December 2015.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network, March 2015.

Abhiram Iyer, Karan Grewal, Akash Velu, Lucas Oliveira Souza, Jeremy Forest, and Subutai Ahmad. Avoiding Catastrophe: Active Dendrites Enable Multi-Task Learning in Dynamic Environments. *Frontiers in Neurorobotics*, 16:846219, April 2022. ISSN 1662-5218. doi: 10.3389/fnbot.2022.846219.

Paul Janson, Wenxuan Zhang, Rahaf Aljundi, and Mohamed Elhoseiny. A Simple Baseline that Questions the Use of Pretrained-Models in Continual Learning, March 2023.

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 0027-8424, 1091-6490. doi: 10.1073/pnas.1611835114.

Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. April 2009.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998. ISSN 1558-2256. doi: 10.1109/5.726791.

Zhizhong Li and Derek Hoiem. Learning without Forgetting, February 2017.

Nicolas Y. Masse, Gregory D. Grant, and David J. Freedman. Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *Proceedings of the National Academy of Sciences*, 115(44), October 2018. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1803839115.

Mark D. McDonnell, Dong Gong, Amin Parveneh, Ehsan Abbasnejad, and Anton van den Hengel. RanPAC: Random Projections and Pre-trained Models for Continual Learning, January 2024.

Matteo Munari, Luca Pasa, Daniele Zambon, Cesare Alippi, and Nicolo Navarin. Understanding Catastrophic Forgetting of Gated Linear Networks in Continual Learning. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, July 2022. doi: 10.1109/IJCNN55064.2022.9892142.

Ian Quah. A Shared Gated Expansion for Multi-Class Dendritic Gated Networks. September 2026. doi: 10.5281/zenodo.22970238.

Eren Sezener, Agnieszka Grabska-Barwińska, Dimitar Kostadinov, Maxime Beau, Sanjukta Krishnagopal, David Budden, Marcus Hutter, Joel Veness, Matthew Botvinick, Claudia Clopath, Michael Häusser, and Peter E. Latham. A rapid and efficient learning rule for biological neural circuits, March 2021.

Gido M. van de Ven and Andreas S. Tolias. Three scenarios for continual learning, April 2019.

Joel Veness, Tor Lattimore, David Budden, Avishkar Bhoopchand, Christopher Mattern, Agnieszka Grabska-Barwinska, Eren Sezener, Jianan Wang, Peter Toth, Simon Schmitt, and Marcus Hutter. Gated Linear Networks, June 2020.

Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11(1):37–57, March 1985. ISSN 0098-3500, 1557-7295. doi: 10.1145/3147.3165.

A BUDGET SWEEPS

At ten epochs the split-CIFAR-100 branch scan gives, for the softmax rule, gated .439, .447, .454, .456, .456 and static .452, .451, .454, .454, .453 at $B \in \{15, 30, 60, 120, 240\}$: a margin of -1.3 to $+0.3$ against the $+0.7$ to $+2.3$ of Table 7 at forty. Under the Bernoulli rule the ten-epoch margin is already $+13$ to $+15.5$.

Table 4: Split-CIFAR-100 fixed-expansion arms and backpropagation baselines against epochs per task. Present classes only; five seeds, centred. Bold: best per column storing no data, and best with a buffer.

Arm	10	40	100	200
SGE-softmax	**.468**	**.468**	.458	.471
SGE	.328	.425	.446	.454
Static gates, Bernoulli readouts	.185	.316	.371	.386
Dense random ReLU	.341	.444	**.472**	**.496**
Linear readout	.454	.451	.456	.449
SGE + replay 5,000	**.557**	**.560**	**.557**	**.554**
Softmax trunk	.098	.099	–	–
+ EWC (best)	.114	.115	–	–
+ LwF (best)	.257	.270	–	–
+ replay 5,000	.480	.480	–	–
+ DER++ 5,000	.531	.528	–	–

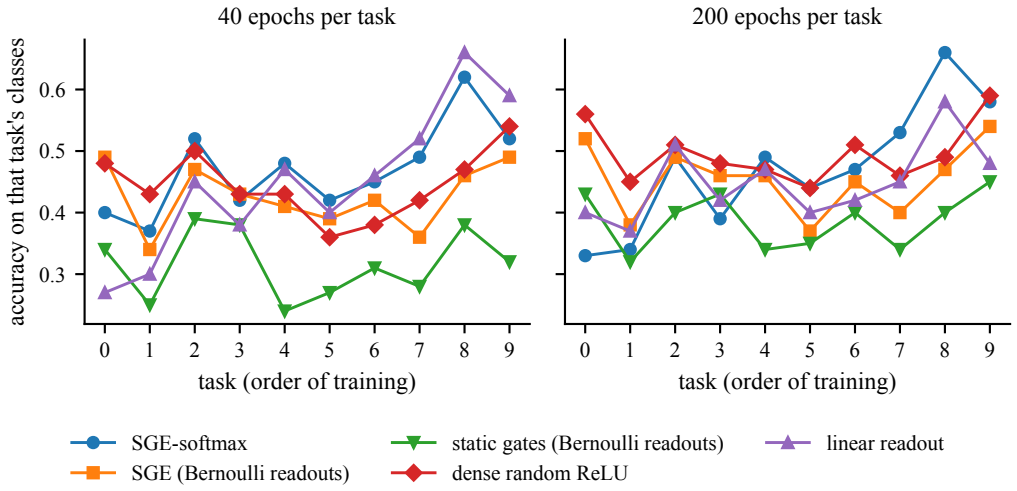

Figure 1: Split-CIFAR-100, accuracy on each task's own ten classes after the whole sequence, in training order, for the fixed-expansion arms at two budgets. Five seeds, centred features, validation-selected rates.

Table 5: K=10 against epochs per task, five seeds, centred. Rotated MNIST all tasks / first task; split-MNIST final accuracy (one-vs-all under the all-classes policy). Bold: best per column.

	rotated MNIST			split-MNIST		
Arm	10	40	100	40	100	200
SGE-softmax	**.826 / .658**	**.827 / .654**	**.828 / .660**	**.859**	**.858**	**.855**
SGE	.805 / .606	.798 / .566	.796 / .559	.749	.752	.750
Static gates, Bernoulli readouts	.712 / .479	.711 / .534	.712 / .480	.660	.665	.670
Dense random ReLU	.727 / .618	.771 / .626	.769 / .632	.693	.685	.691
Linear readout	.763 / .594	.761 / .579	.764 / .596	.817	.823	.829
One-vs-all	.779 / .604	.801 / .603	.803 / .581	.764	.708	.707

Table 6: CIFAR-100 drifts at $K=100$ against epochs per task, all tasks / first task, softmax readouts on the gated expansion and on static gates, and the dense random ReLU projection (Bernoulli readouts; not run at ten epochs). Five seeds, centred, pretrained features. Bold: best per column.

	rotation			noise		
Arm	10	40	100	10	40	100
SGE-softmax	**.461 / .563**	**.500 / .596**	**.507 / .592**	**.158 / .442**	**.189 / .530**	**.203 / .556**
Static	.437 / .547	.473 / .561	.484 / .560	.149 / .374	.177 / .452	.190 / .480
Dense random ReLU	–	.413 / .494	.454 / .532	–	.125 / .284	.141 / .335

Table 7: Split-CIFAR-100 at 40 epochs per task against the number of branches B, both update rules, each B searched at matched effective rate ($60/B$ times the base grid). Five seeds, centred. Bold: best per column.

B	15	30	60	120	240
SGE-softmax	**.454**	**.460**	**.468**	**.471**	**.473**
Static, softmax readouts	.447	.449	.449	.450	.450
SGE	.386	.412	.425	.431	.436
Static, Bernoulli readouts	.314	.317	.316	.316	.315

B DATA SPLITS

Table 8: SGE-softmax against each arm over five data splits (the split every other table uses and four more), each split five seeds at its own validation-selected rates. Level is mean $\pm$ standard deviation over splits; margins are mean points with the smallest split's margin in parentheses. Conditions with a first-task measure get a row for each measure. Static (softmax) is the linear readout at $K=10$ and on split-CIFAR-100, to which it is algebraically equal. Split-MNIST one-vs-all is under the all-classes policy.

	SGE-softmax	SGE-softmax minus each arm, points (smallest split)				
	level	SGE	static (Bernoulli)	static (softmax)	dense ReLU	one-vs-all
split-MNIST	.835 $\pm$ .016	+9.5 (7.8)	+18.0 (16.6)	+2.8 (1.2)	+15.6 (13.7)	+15.2 (13.5)
rotated MNIST						
all tasks	.818 $\pm$ .008	+2.2 (1.3)	+12.6 (11.6)	+7.4 (6.4)	+6.8 (5.9)	+2.6 (2.2)
first task	.671 $\pm$ .016	+9.3 (6.4)	+16.6 (12.9)	+8.7 (6.4)	+7.5 (2.8)	+10.2 (7.9)
split-CIFAR-100	.469 $\pm$ .001	+4.4 (4.3)	+15.4 (14.9)	+1.8 (1.6)	+2.4 (2.2)	–
CIFAR-100 rotation						
all tasks	.500 $\pm$ .001	+0.4 (0.3)	+3.0 (2.9)	+2.8 (2.7)	–	–
first task	.597 $\pm$ .001	+0.6 (0.5)	+4.2 (4.0)	+3.7 (3.5)	–	–
CIFAR-100 noise						
all tasks	.188 $\pm$ .001	+0.5 (0.2)	+3.4 (3.3)	+1.1 (0.9)	–	–
first task	.531 $\pm$ .000	+2.1 (2.0)	+14.2 (11.9)	+7.9 (7.8)	–	–

Seeds redraw the expansion and the initial weights but not the data, so the standard deviations in the main tables are gate-draw variability alone. Table 8 redraws the stratified validation split and, on MNIST, the 5,000-row training and 1,000-row test subsample. The CIFAR-100 cells barely move. The MNIST cells move by up to 0.04, and the split every other table uses is SGE-softmax's best on split-MNIST (.858) and on rotated-MNIST all-tasks accuracy (.828), so the main tables overstate those two levels by two points and one; its rotated-MNIST first-task cell (.660) sits below the five-

split mean. No margin changes sign on any split. The smallest is split-MNIST over the linear readout, $+1.2$.

C Replay buffers

Table 9: Split-CIFAR-100 at 40 epochs per task against replay buffer size, final 100-way accuracy, five seeds, centred. The zero column is each arm without a buffer. DER++ at $\alpha{=}\beta{=}0.5$; its best setting at 1,000 rows ($\alpha{=}\beta{=}0.2$) reaches .391. *: best rate at the grid edge. Bold: best per column.

Buffer (rows)	0	200	1,000	5,000
SGE-softmax + replay	**.468**	**.425**	**.491**	**.596**
SGE + replay	.425	.210*	.376	.560
Trunk + replay	.099	.177	.314	.480
Trunk + DER++	.099	.233	.372	.528

SGE-softmax leads DER++ at every buffer size (Table 9). At 200 rows both expansion arms score below their own rehearsal-free runs, and on split-MNIST so does SGE-softmax at 50 rows (.835 against .858); from 1,000 rows on, replay helps both. Each step draws as many buffered rows as the task holds, so 200 rows, two per earlier class by the last task, are each repeated about twenty times per step and carry half of every update; we suspect that weighting, but have not tested it. On split-MNIST the two rules meet once the buffer is large, SGE-softmax 0.920 and 0.931 at 1,000 and 5,000 rows against SGE's 0.920 and 0.936.

D Relation to Quah (2026)

Quah (2026) introduced the expansion, trained it with the Bernoulli readouts only, and stated Definition 1. This paper adds the softmax readouts and their half of Proposition 1, per-pixel centring and the firing-rate measurement behind it, the linear readout, the prototype classifier, the binary code, the backpropagation-trained continual-learning baselines, the noised-MNIST negative control and the transfer-gap measurement of Table 11. That report's numbers came from a different protocol: MNIST centred and scaled by global scalars and CIFAR-100 features left as extracted, so no per-pixel centring, a coarser learning-rate grid, 40 epochs per task under class-incremental training and 10 under domain-incremental, ten seeds at $K{=}10$ and three to five at CIFAR-100, and a trunk trained with one full-batch step per epoch. Its finding that the gating margin is gone at $K{=}100$ rested on a tie under CIFAR-100 rotation (0.275 / 0.297 against the static control's 0.274 / 0.290); that tie reproduces here without centring (0.275 / 0.298 against 0.274 / 0.290, Table 10) and becomes a margin with it. It also read the first-task lead of task-identity gates, selected by the ground-truth task label, as the retention a gate that knows the task can reach (0.736 against the input-inferred gates' 0.610 on rotated MNIST); centred, at a hundred epochs, that lead is 0.614 against 0.559 for the Bernoulli gates and reverses against SGE-softmax (0.660), and the oracle trails both over all tasks (0.771 against 0.796 and 0.828), so it no longer brackets what input-inferred gates can retain. The softmax readouts, which that report did not run, are the more protocol-sensitive arm: without centring, its coarser grid gives them 0.159 / 0.203 on the CIFAR-100 rotation drift where the $\sqrt{10}$-step grid gives 0.340 / 0.404, and on split-MNIST they sit below the static control (0.629 against 0.674) until the wider grid (0.737) and centring (0.859, against the static control's 0.660) lift them. Its $K{=}10$ result that input-dependent gates beat a matched static-gate control under rotation, its small margin under pixel noise and its trained-autoencoder result, which we did not rerun, carry over.

Table 10: The same arms under the earlier preprocessing (plain: MNIST centred and scaled by global scalars, CIFAR-100 features as extracted) and with per-pixel, or per-feature, centring (centred), all-tasks accuracy (first-task after the slash for CIFAR rotation and rotated MNIST). Ten epochs per task except split-MNIST at 40. Five seeds, except the plain split-CIFAR column (†: an earlier ten-seed run; its static-gate cell is ± 0.050). *: grid edge. Bold: best per column.

	split-CIFAR-100		CIFAR rotation		split-MNIST		rotated MNIST	
Arm	plain†	centred	plain	centred	plain	centred	plain	centred
SGE-softmax	.177*	**.468**	.340 / .404	**.461 / .563**	**.737***	**.859**	**.808 / .644**	**.826 / .658**
SGE	**.408**	.328	.275 / .298	.456 / .554	.727	.749	.793 / .616	.805 / .606
Static, Bernoulli readouts	.335	.185	.274 / .290	.428 / .531	.690	.660	.750 / .572	.712 / .479
Static, softmax readouts	–	.454	**.365 / .422**	.437 / .547	–	–	–	–
Dense random ReLU	.354	.341	–	–	.691	.693	.732 / .619	.727 / .618
Linear readout	.182*	.454	–	–	.731	.817	.766 / .602	.763 / .594
One-vs-all (all classes)	–	–	–	–	.480	.764*	.796 / .642	.779 / .604

E CONTINUAL-LEARNING BASELINES

All four methods run on the backpropagation-trained trunk of Section 2.4. EWC accumulates the empirical Fisher diagonal at each task's final weights and pulls every later step back toward them, in the online form (Fisher diagonals summed across tasks, anchored at the most recent task's weights); the penalty strength λ is swept over $\{1, 100, 10{,}000\}$. Experience replay trains every step after the first task on the current minibatch concatenated with an equal number of rows from a reservoir sample of the tasks seen so far; rows enter the reservoir once per task, so the buffer stays a uniform sample of the stream. LwF distils temperature-softened targets ($T{=}2$) from a frozen copy of the previous task's weights onto the new task's inputs, storing no data; the distillation weight is swept over $\{0.3, 1, 3\}$. DER++ is replay in which each buffered row also carries the logits the model assigned it when stored, and the replayed batch is trained toward those logits (mean squared error, weight α) as well as its labels (cross-entropy, weight β); $\alpha{=}\beta$ is swept over $\{0.2, 0.5, 1\}$ at a 1,000-row buffer and fixed at 0.5 elsewhere, and, as stated in Section 2.4, rows enter its reservoir once per task with end-of-task logits. Every method's learning rate is selected on the validation split like every other arm's, and each hyperparameter value is reported separately; the tables quote the best value per cell.

F TRANSFER GAP AND PROTOTYPE BASELINES

Table 11 measures Definition 1's second condition in the space the gates read. Along rotation a readout fitted at the level beats the clean readout by 11 to 22 points, so the feature-to-label mapping moves between levels while the features stay usable (within-level accuracy .41 to .65). Under noise the first step is a regime change of the same kind (.255 within against .093 transferred at $\sigma{=}0.1$), and from $\sigma{=}0.2$ the features themselves are gone: a readout trained at the level reaches .06 to .12. The joint i.i.d. ceilings on the clean features, all 100 classes at once, are .653 for the linear readout and $.660 \pm .004$ for SGE-softmax at 40 full-batch steps (.670 and .678 at 400), so on this representation the expansion adds under a point of expressiveness and the sequential arms of Table 3 retain about 70% of their own ceiling.

Table 11: Feature-space transfer gap on the pretrained CIFAR-100 features. For each drift level, a linear softmax readout trained on that level and tested on it (within) against the level-0 readout tested on it (from 0), 40 full-batch steps, rate selected on the held-out rows of the level; Euclidean nearest-class-mean prototypes alongside. Features centred on the level-0 training mean, as everywhere else.

		linear readout			prototypes		
Drift	Level	within	from 0	gap	within	from 0	gap
rotation	0°	.653	.653	.000	.516	.516	.000
	15°	.523	.413	.111	.375	.326	.049
	30°	.441	.282	.159	.342	.243	.099
	45°	.410	.193	.217	.323	.172	.151
	60°	.410	.218	.192	.316	.207	.109
noise	σ=0	.653	.653	.000	.516	.516	.000
	σ=0.1	.255	.093	.161	.207	.054	.153
	σ=0.2	.119	.028	.091	.122	.022	.101
	σ=0.3	.079	.018	.061	.091	.013	.077
	σ=0.4	.061	.015	.046	.073	.011	.062

Table 12: The prototype classifier under both distance rules, every condition of Tables 2 and 3: class means of the training rows (pooled over the five levels under drift), all tasks / first task where both exist. Split-CIFAR-100 cosine is on the features as extracted (.520 centred); every other cell is on the centred inputs.

Condition	Euclidean	cosine
split-CIFAR-100	.516	.526
CIFAR-100 rotation	.333 / .419	.335 / .408
CIFAR-100 noise	.120 / .238	.141 / .337
split-MNIST	.824	.811
rotated MNIST	.734 / .654	.732 / .623
noised MNIST	.817 / .822	.808 / .815