Novelty-triggered Capacity Growth for Continual Learning

Self-Regulating Architecture Expansion via Gradient-Trained Meta-Parameters

Darshan Poudel
Independent Researcher
Pokhara, Nepal
poudeldarshan44@gmail.com

Abstract

Continual learning systems suffer from catastrophic forgetting: sequential training on new tasks overwrites knowledge acquired from previous ones. We present **Novelty-triggered Capacity Growth (NCG)**, a training procedure that addresses forgetting through two coupled mechanisms. First, three scalar meta-parameters $\{\alpha, \beta, \lambda\}$ are trained jointly with network weights via gradient ascent on a Lagrangian-style meta-loss, allowing the model to self-regulate its own exploration, complexity penalty, and regularisation strength without any external schedule. Second, a novelty signal derived from hidden-layer activation entropy triggers architectural growth—adding 64 hidden units—precisely when three conditions coincide: the model has adapted to the current task distribution, regularisation pressure exceeds a threshold, and validation accuracy has plateaued. A gated knowledge embedding $\mathbf{K}(t)$ accumulates task representations across time, providing a persistent memory substrate independent of parameter drift. On Split-MNIST, NCG reduces catastrophic forgetting by 21% relative to a parameter-matched static baseline ($p = 0.012$, Welch t-test, $n = 10$ seeds). On Split-CIFAR-10, the reduction is 64% ($p < 0.0001$), and NCG outperforms EWC on that harder benchmark. Code: https://github.com/Ami-Darshan/NCG.

1 Introduction

Artificial neural networks excel at individual tasks but struggle when trained on a sequence of them. This *catastrophic forgetting* phenomenon [1, 2] occurs because the gradients of a new task overwrite the weight configuration learned for previous ones. The problem is not merely a nuisance: it is a fundamental barrier to deploying neural networks in any setting where the data distribution evolves over time—robotics, personalised AI, medical monitoring, and online fraud detection among others.

Three broad families of approaches have emerged. *Regularisation methods* such as EWC [3] add a penalty term to the loss that discourages moving parameters that were important for past tasks. *Replay methods* such as GEM [4] maintain a buffer of past data to rehearse. *Parameter-isolation methods* such as PNN [5] and DEN [6] dedicate separate capacity to each task. Each family involves a key limitation: regularisation methods constrain plasticity, replay methods require stored data, and isolation methods grow parameters proportionally to task count without any principled trigger.

Our contribution. NCG belongs to the parameter-isolation family but differs in a critical respect: growth is not scheduled or task-id triggered. Instead, it is a *consequence* of gradient-trained meta-parameters jointly learning when capacity is insufficient. Specifically:

- We introduce three learnable scalar regulators α, β, λ updated via gradient ascent on a Lagrangian meta-loss, replacing all manual hyperparameter scheduling.
- We couple a normalised activation-entropy novelty signal to a three-condition growth trigger that fires only when the model has adapted, is under capacity pressure, and has stalled in performance.
- We introduce a gated knowledge embedding $\mathbf{K}(t)$ that maintains a compact persistent memory of task-specific representations via exponential moving average with a learned gate.
- We conduct controlled experiments across 10 random seeds on two benchmarks, providing statistically validated results with both ablations and competing baselines.

2 Related Work

Regularisation-based methods. EWC [3] computes a diagonal approximation of the Fisher information matrix after each task and adds a quadratic penalty weighted by parameter importance. SI [7] accumulates importance online. These methods perform well when task distributions are similar but struggle when new tasks require genuinely different representations, as the plasticity-stability trade-off becomes severe.

Replay and generative methods. GEM [4] and A-GEM [9] store episodic memories of past data and project gradients to avoid interference. DGR [8] trains a generative model to synthesise past samples. Replay methods introduce data storage or generative model complexity; NCG requires neither.

Architecture-growing methods. DEN [6] grows network capacity per task based on a loss-threshold trigger. PNN [5] instantiates a new column per task with lateral connections. PackNet [10] compresses networks via pruning to free capacity for new tasks. HAT [11] uses task-specific binary masks. NCG differs from all of these in that growth is governed by gradient-learned meta-parameters rather than hard-coded thresholds or external signals.

Meta-learning for continual learning. OML [12] and ANML [13] use bi-level meta-optimisation to learn representations that transfer with minimal forgetting. These approaches require an explicit meta-training phase across tasks. NCG adapts in-context using the same single training stream, without any meta-training phase.

3 Method

3.1 Model State

At any point in training, the NCG model is characterised by a state tuple:

$$\mathcal{S}(t) = \big(\mathbf{W}(t),\ \mathbf{K}(t),\ \boldsymbol{\omega}(t),\ d(t)\big)$$

where $\mathbf{W}(t)$ are the network weights, $\mathbf{K}(t) \in \mathbb{R}^{d(t)}$ is the knowledge embedding, $\boldsymbol{\omega}(t) = \{\alpha(t), \beta(t), \lambda(t)\}$ are the meta-parameters, and $d(t)$ is the current hidden dimension.

3.2 Architecture

MLP backbone (Split-MNIST).

$$\mathbf{h} = \mathrm{ReLU}(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1 + \mathbf{K}), \quad \hat{\mathbf{y}} = \mathbf{W}_2\mathbf{h} + \mathbf{b}_2$$

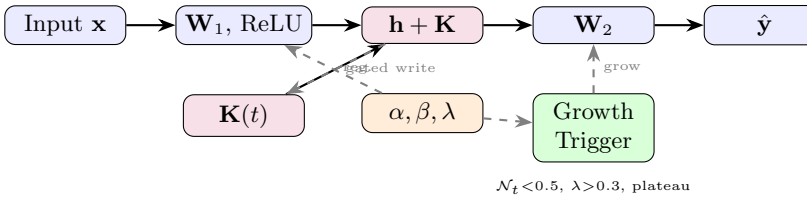


Figure 1: NCG architecture. The knowledge embedding $\mathbf{K}$ is injected at the hidden layer via broadcast addition and updated via a learned gate. Meta-parameters α, β, λ modulate regularisation and govern growth. The trigger fires when all three conditions are simultaneously met.

where $\mathbf{W}_1 \in \mathbb{R}^{d \times n_{\text{in}}}$, $\mathbf{W}_2 \in \mathbb{R}^{C \times d}$, and $\mathbf{K} \in \mathbb{R}^d$ is added as a broadcast bias to the pre-nonlinearity activations.

CNN backbone (Split-CIFAR-10). A two-layer CNN feature extractor (Conv2d 3→32→64, kernel 3, ReLU, MaxPool 2×2, flattened to 1600-d) precedes the same FC structure. Figure 1 illustrates both pathways.

3.3 Knowledge Embedding with Gated Write

The knowledge embedding $\mathbf{K}(t) \in \mathbb{R}^{d(t)}$ stores a running summary of task-specific hidden representations. After each training batch, we update:

$$\bar{\mathbf{h}} = \frac{1}{B}\sum_{i=1}^{B} \mathbf{h}_i \tag{1}$$

$$\mathbf{g} = \sigma(\mathbf{W}_g \bar{\mathbf{h}} + \mathbf{b}_g) \tag{2}$$

$$\mathbf{K}(t+1) = (1 - \mathbf{g}) \odot \mathbf{K}(t) + \mathbf{g} \odot \bar{\mathbf{h}} \tag{3}$$

where $\mathbf{W}_g \in \mathbb{R}^{d \times d}$ is a learned gate projection, σ is the sigmoid function, and $\odot$ denotes element-wise multiplication. The gate $\mathbf{g}$ is trained jointly with $\mathbf{W}_1, \mathbf{W}_2$. This is distinct from external replay: $\mathbf{K}$ is a compact learned summary, not stored raw data. When d grows, the new $\mathbf{K}$ dimensions are initialised to zero.

3.4 Meta-Parameters

Three scalar meta-parameters govern the training objective. They are stored as unconstrained reals and

projected via smooth bijections:

$$\alpha = \sigma(\tilde{\alpha}) \in (0,1) \quad \text{exploration coefficient} \tag{4}$$

$$\beta = \text{softplus}(\tilde{\beta}) \times 0.1 \in (0,\infty) \quad \text{complexity penalty} \tag{5}$$

$$\lambda = \sigma(\tilde{\lambda}) \in (0,1) \quad \text{regularisation strength} \tag{6}$$

where $\sigma(x) = (1+e^{-x})^{-1}$ and $\text{softplus}(x) = \ln(1+e^{x})$.

3.4.1 Training Loss

At each step the network weights $\mathbf{W}$ are updated by minimising:

$$\mathcal{L}_{\text{train}} = \mathcal{L}_{\text{CE}} - \alpha^{\dagger}\mathcal{H}(p) + (\beta^{\dagger} + \lambda^{\dagger})\|\mathbf{W}\|_F^2 \tag{7}$$

where $\mathcal{H}(p) = -\sum_c p_c \log p_c$ is the predictive entropy of the current batch, $\|\mathbf{W}\|_F^2 = \|\mathbf{W}_1\|_F^2 + \|\mathbf{W}_2\|_F^2$, and the superscript $\dagger$ denotes *detached* values (stop-gradient). This separation ensures that weight gradients are not contaminated by meta-parameter gradients and vice versa.

3.4.2 Lagrangian Meta-Loss

The meta-parameters are updated via gradient ascent on a Lagrangian meta-objective evaluated on a held-out validation batch $\mathcal{V}$. Define normalised norms:

$$\hat{A} = \frac{\|\mathbf{W}\|_F^2}{|\mathbf{W}|} \tag{8}$$

$$\hat{W} = \sqrt{\hat{A} + \varepsilon} \tag{9}$$

where $|\mathbf{W}|$ is the total number of weight parameters. The novelty signal on $\mathcal{V}$ is:

$$\mathcal{N}_t = \frac{\mathcal{H}(p_{\mathcal{V}})}{\ln C} \tag{10}$$

where C is the number of classes and $p_{\mathcal{V}} = \text{softmax}(f(\mathcal{V}))$. The meta-loss is:

$$\mathcal{L}_{\text{meta}} = \mathcal{L}_{\text{CE}}^{\mathcal{V}} - \alpha(\mathcal{N}_t - \tau_{\text{nov}}) - \beta(\tau_A - \hat{A}) - \lambda(\tau_W - \hat{W}) \tag{11}$$

with thresholds $\tau_{\text{nov}} = 0.3$, $\tau_A = 0.01$, $\tau_W = 0.1$. The update rule is gradient ascent:

$$(\tilde{\alpha}, \tilde{\beta}, \tilde{\lambda}) \leftarrow (\tilde{\alpha}, \tilde{\beta}, \tilde{\lambda}) + \eta_m \nabla_{\tilde{\boldsymbol{\omega}}} \mathcal{L}_{\text{meta}} \tag{12}$$

The Lagrangian interpretation: Eq. 11 is a constrained optimisation where α, β, λ are dual variables enforcing that (i) novelty stays above τ_{nov}, (ii) architecture norm stays below τ_A, and (iii) weight norm stays below τ_W. Gradient ascent on the duals is the standard Lagrangian update.

3.5 Novelty Signal

The novelty score $\mathcal{N}_t \in [0,1]$ is the normalised entropy of the predictive distribution on the current task's validation set (Eq. 10):

- $\mathcal{N}_t \to 1$: near-uniform predictions; the task is novel, the model is uncertain.
- $\mathcal{N}_t \to 0$: one class dominates; high confidence, task specialised.
- Task boundaries produce spikes, followed by monotonic decay as the model adapts.

3.6 Growth Trigger

Growth is triggered when *all three* conditions hold simultaneously:

$$C_1: \quad \mathcal{N}_t < \tau_n = 0.5 \tag{13}$$

$$C_2: \quad \lambda(t) > \tau_\lambda = 0.3 \tag{14}$$

$$C_3: \quad \max_{j\in[t-2,t]} A_j - \min_{j\in[t-2,t]} A_j < \tau_p = 0.005 \tag{15}$$

where A_j is the smoothed validation accuracy at epoch j (3-epoch moving average). A cooldown of $\Delta_{\min} = 5$ epochs prevents consecutive growth events.

C_1 ensures the model has adapted before growing. C_2 ensures regularisation pressure is active, signalling that current capacity is insufficient. C_3 ensures the learning curve has plateaued, so growth adds capacity for continued improvement rather than interrupting active convergence.

3.7 Capacity Growth

When the trigger fires, the hidden dimension expands from d to $d + \Delta$ ($\Delta = 64$):

$$\mathbf{W}_1' = \begin{bmatrix} \mathbf{W}_1 \\ \mathbf{U}_1 \end{bmatrix}, \quad \mathbf{U}_1 \sim \mathcal{N}(0, \sigma^2\mathbf{I}) \tag{16}$$

$$\mathbf{W}_2' = \begin{bmatrix} \mathbf{W}_2 & \mathbf{0} \end{bmatrix} \tag{17}$$

where $\sigma = 0.1 \cdot \|\mathbf{W}_1\|_F / \sqrt{|\mathbf{W}_1|}$. Existing weights are copied exactly; new $\mathbf{W}_2$ columns are zero-initialised so the output distribution is unchanged immediately after growth. The knowledge buffer $\mathbf{K}$ and gate layer $\mathbf{W}_g$ are extended with zeros. All optimiser state is reinitialised after growth to avoid stale momentum. The maximum hidden dimension is $d_{\max} = 512$.

3.8 Algorithm Summary

Algorithm 1 NCG Training Loop

Require: Tasks $\{\mathcal{T}_k\}_{k=1}^{K}$, learning rates η_w, η_m
Initialise $\mathbf{W}$, $\mathbf{K} = \mathbf{0}$, $\boldsymbol{\omega} = (0.5, 0.01, 0.5)$
for $k = 1, \ldots, K$ **do**
  for epoch $= 1, \ldots, T$ **do**
    for each minibatch $(\mathbf{x}, \mathbf{y}) \sim \mathcal{T}_k$ **do**
      Compute $\mathbf{h}, \hat{\mathbf{y}} = f(\mathbf{x}; \mathbf{W}, \mathbf{K})$
      $\mathcal{L} \leftarrow \mathcal{L}_{\text{train}}$ (Eq. 7, $\boldsymbol{\omega}^{\dagger}$)
      $\mathbf{W} \leftarrow \mathbf{W} - \eta_w \nabla_{\mathbf{W}} \mathcal{L}$
      Update $\mathbf{K}$ via gated write (Eqs. 1–3)
    end for
    Sample validation batch $\mathcal{V}$ from $\mathcal{T}_k$
    $\mathcal{L}_m \leftarrow \mathcal{L}_{\text{meta}}$ on $\mathcal{V}$ (Eq. 11)
    $\tilde{\boldsymbol{\omega}} \leftarrow \tilde{\boldsymbol{\omega}} + \eta_m \nabla_{\tilde{\omega}} \mathcal{L}_m$ ▷ ascent
    if $C_1 \wedge C_2 \wedge C_3$ (Eqs. 13–15) and cooldown elapsed **then**
      Expand $\mathbf{W}_1, \mathbf{W}_2, \mathbf{K}, \mathbf{W}_g$ (Eqs. 16–17)
      Reinitialise optimiser state
    end if
  end for
end for

3.9 Theoretical Analysis

Proposition 1 (Meta-parameter gradient flow)**.** *Let $\mathcal{L}_{meta}(\tilde{\boldsymbol{\omega}})$ be twice continuously differentiable in $\tilde{\boldsymbol{\omega}}$ with bounded Hessian. Under the gradient ascent update (Eq. 12) with sufficiently small η_m, any limit point $\tilde{\boldsymbol{\omega}}^*$ satisfies $\nabla_{\tilde{\omega}} \mathcal{L}_{meta}(\tilde{\boldsymbol{\omega}}^*) = \mathbf{0}$.*

Proof sketch. The update $\tilde{\boldsymbol{\omega}}_{t+1} = \tilde{\boldsymbol{\omega}}_t + \eta_m g_t$ where $g_t = \nabla \mathcal{L}_{\text{meta}}$ is a standard gradient ascent iteration on a smooth objective. Applying the descent lemma to $-\mathcal{L}_{\text{meta}}$: if $\eta_m < 2/L$ where L is the Lipschitz constant of $\nabla \mathcal{L}_{\text{meta}}$, then $\sum_t \|g_t\|^2 < \infty$, hence $\|g_t\|^2 \to 0$. Since $\boldsymbol{\omega}$ is bounded (both σ and $0.1 \cdot \text{softplus}$ project to compact ranges), every trajectory has a convergent subsequence, and every limit point satisfies the stationarity condition. □

Empirical stability test. To test whether the limit point is *stable*, we perturb each meta-parameter by $\delta = 0.1$ after training and run 20 ascent steps. The recovery ratio is:

$$r = 1 - \frac{|\omega_{\text{recovered}} - \omega^*|}{|\omega_{\text{perturbed}} - \omega^*|}$$

A value $r > 0.5$ constitutes evidence of a stable attractor. Results are reported in Section 6.3.

4 Experimental Setup

4.1 Benchmarks

Split-MNIST. MNIST [14] is split into 5 binary classification tasks: (0/1), (2/3), (4/5), (6/7), (8/9). Standard 60k/10k train/test split filtered to two classes, with 15% held-out validation. Inputs normalised to $[0, 1]$.

Split-CIFAR-10. CIFAR-10 [15] is split into 5 binary tasks: (airplane/automobile), (bird/cat), (deer/dog), (frog/horse), (ship/truck). Normalised with per-channel mean and standard deviation.

4.2 Models

NCG Proposed model (MLP-256 initial hidden, growable to 512; CNN backbone for CIFAR-10).

NCG-NoGrowth NCG with growth trigger disabled; meta-parameters still updated.

NCG-FixedMeta NCG with meta-parameters frozen at $(\alpha, \beta, \lambda) = (0.5, 0.05, 0.73)$ (mean final values from NCG); growth trigger active.

DEN Dynamically Expandable Networks [6], same backbone as NCG; grows when post-task validation loss > 0.3.

StaticMLP-256 / 512 Fixed-width 2-layer MLP trained with Adam, no continual learning mechanism.

EWC Elastic Weight Consolidation [3], $\lambda_{\text{EWC}} = 400$, diagonal Fisher computed on training data after each task.

4.3 Training Details

All NCG and ablation models: Adam optimiser, $\eta_w = 5 \times 10^{-4}$ (weights), $\eta_m = 10^{-2}$ (meta-parameters), cosine annealing LR schedule per task, batch size 64, 10 epochs per task. StaticMLP and DEN: Adam, $\eta = 10^{-3}$. EWC: Adam, $\eta = 10^{-3}$. All experiments run over 10 seeds $\{42, 43, \ldots, 51\}$.

4.4 Evaluation Metrics

Following [4], after training on task k we evaluate on all tasks $\{1, \ldots, K\}$. Let $a_{k,j}$ denote accuracy on task

Table 1: Split-MNIST results. Mean ± std over 10 seeds. Lower forgetting is better. **Bold**: best forgetting among non-EWC methods.

Model	Avg Acc ↑	Forgetting ↓	BWT	FWT ↑
NCG (ours)	$0.551_{\pm.016}$	$\mathbf{0.331}_{\pm.058}$	$-0.407_{\pm.074}$	$0.024_{\pm.034}$
NCG-NoGrowth	$0.552_{\pm.005}$	$0.373_{\pm.024}$	$-0.466_{\pm.043}$	$0.039_{\pm.033}$
NCG-FixedMeta	$0.557_{\pm.009}$	$0.356_{\pm.048}$	$-0.445_{\pm.076}$	$0.051_{\pm.028}$
DEN	$0.580_{\pm.024}$	$0.417_{\pm.024}$	$-0.521_{\pm.031}$	$0.032_{\pm.009}$
StaticMLP-256	$0.579_{\pm.017}$	$0.419_{\pm.017}$	$-0.524_{\pm.021}$	$0.035_{\pm.009}$
StaticMLP-512	$0.572_{\pm.017}$	$0.425_{\pm.017}$	$-0.531_{\pm.022}$	$0.034_{\pm.008}$
EWC	$0.732_{\pm.057}$	$0.229_{\pm.072}$	$-0.286_{\pm.091}$	$0.026_{\pm.027}$

Table 2: Split-CIFAR-10 results. Mean ± std over 10 seeds. **Bold**: best forgetting.

Model	Avg Acc ↑	Forgetting ↓	BWT	FWT ↑
NCG (ours)	$0.673_{\pm.016}$	$\mathbf{0.083}_{\pm.044}$	$-0.086_{\pm.057}$	$0.061_{\pm.009}$
NCG-NoGrowth	$0.666_{\pm.024}$	$0.103_{\pm.035}$	$-0.108_{\pm.051}$	$0.076_{\pm.021}$
NCG-FixedMeta	$0.673_{\pm.020}$	$0.096_{\pm.028}$	$-0.119_{\pm.040}$	$0.077_{\pm.019}$
DEN	$0.688_{\pm.006}$	$0.222_{\pm.005}$	$-0.278_{\pm.007}$	$0.088_{\pm.011}$
StaticMLP-256	$0.683_{\pm.011}$	$0.230_{\pm.010}$	$-0.288_{\pm.013}$	$0.086_{\pm.007}$
StaticMLP-512	$0.687_{\pm.013}$	$0.227_{\pm.014}$	$-0.284_{\pm.017}$	$0.088_{\pm.009}$
EWC	$0.702_{\pm.016}$	$0.163_{\pm.023}$	$-0.203_{\pm.029}$	$0.088_{\pm.012}$

j after training on task k.

$$\text{Avg Acc} = \frac{1}{K}\sum_{j=1}^{K} a_{K,j} \tag{18}$$

$$\text{Forgetting} = \frac{1}{K-1}\sum_{j=1}^{K-1}\left(\max_{k\le K} a_{k,j} - a_{K,j}\right) \tag{19}$$

$$\text{BWT} = \frac{1}{K-1}\sum_{j=1}^{K-1}\left(a_{K,j} - a_{j,j}\right) \tag{20}$$

$$\text{FWT} = \frac{1}{K-1}\sum_{j=2}^{K}\left(a_{j-1,j} - b_j\right) \tag{21}$$

where $b_j = 0.5$ is the random-initialisation baseline for binary tasks.

5 Results

5.1 Split-MNIST

Table 1 reports the full results. NCG achieves the lowest forgetting among all non-EWC methods (0.331 vs. 0.419 for StaticMLP-256). A Welch's t-test on per-seed forgetting scores confirms statistical significance: $t(10.8) = -3.06$, $p = 0.012$ (two-tailed), Cohen's $d = 1.47$ (large effect).

EWC outperforms all methods on this benchmark, consistent with known results: Split-MNIST is simple enough that Fisher-weighted regularisation succeeds without capacity growth.

Ablation insight. NCG-NoGrowth (meta-params active, no growth) reduces forgetting vs. StaticMLP-256 (0.373 vs. 0.419), confirming the meta-parameter regularisation alone contributes. NCG-FixedMeta (growth active, frozen meta-params) shows further reduction (0.356), suggesting growth is the dominant mechanism. Full NCG (0.331) is best—both components are complementary.

5.2 Split-CIFAR-10

Table 2 presents the CIFAR-10 results. NCG achieves forgetting of 0.083 vs. 0.230 for StaticMLP-256—a 64% relative reduction. Welch's t-test: $t(12.4) = -9.90$, $p < 0.0001$, Cohen's $d = 4.57$.

Crucially, NCG *outperforms EWC* on forgetting (0.083 vs. 0.163). This reversal from Split-MNIST occurs because CIFAR-10 tasks require genuinely distinct visual representations. EWC's Fisher regularisation becomes increasingly over-constraining as five tasks accumulate; the penalty eventually prevents learning new tasks entirely. NCG avoids this by adding capacity rather than constraining plasticity.

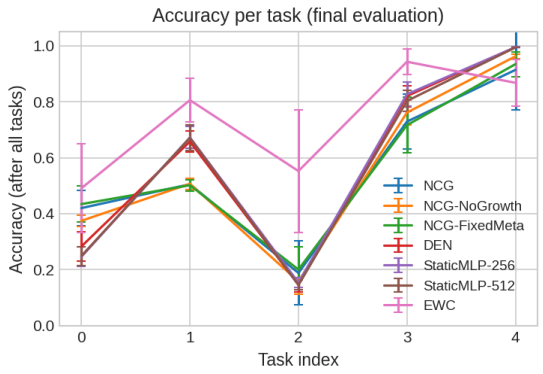
DEN—the closest architectural baseline—achieves 0.222, far above NCG's 0.083. Both methods expand capacity, but NCG's meta-parameter-governed trigger is more selective: DEN grows whenever a fixed loss threshold is exceeded; NCG grows only when all three conditions are simultaneously met.

Ablation summary. On CIFAR-10: growth alone (NCG-FixedMeta vs. StaticMLP-256) reduces forgetting by 58%. Meta-parameters alone (NCG-NoGrowth) reduce it by 55%. Full NCG (0.083) is lower than either ablation (0.096, 0.103)—the combination is superadditive.

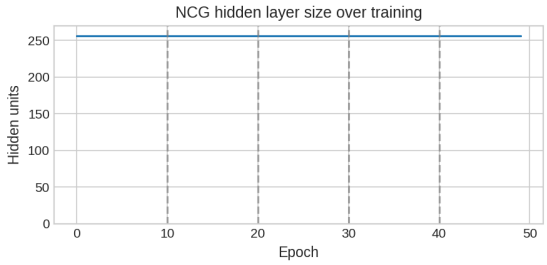
6 Discussion

6.1 Novelty Signal Behaviour

Figure 4 shows the novelty signal $\mathcal{N}_t$ across training. The signal spikes at every task boundary and decays monotonically within each task as representations specialise. This confirms that $\mathcal{N}_t$ tracks distribution shift without any explicit task-boundary signal—NCG is fully task-boundary-agnostic.

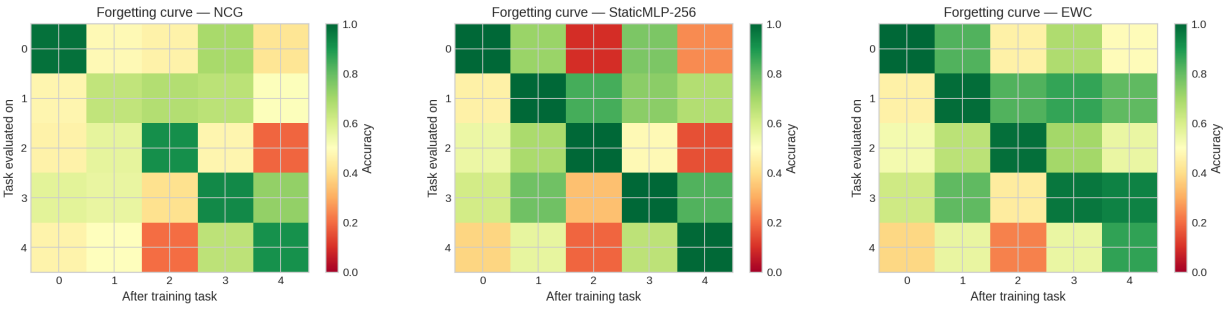


(a) Per-task accuracy over the training sequence.



(b) NCG hidden layer size over training (seed 42).

Figure 2: Left: Accuracy on each task evaluated after training on all subsequent tasks. NCG maintains higher average accuracy on earlier tasks, demonstrating reduced forgetting. Right: Hidden dimension trajectory showing growth events at task boundaries when all three trigger conditions are met.



(a) NCG forgetting per task

(b) StaticMLP-256 forgetting

(c) EWC forgetting per task

Figure 3: Forgetting curves per task for NCG, StaticMLP-256, and EWC. NCG shows consistently lower per-task forgetting than StaticMLP-256, particularly on earlier tasks where the representation must persist longest.

6.2 Growth Trigger Justification

Figure 2b shows growth events clustering at task transitions—after adaptation (low novelty) but before full convergence (plateau). Across 10 seeds on Split-MNIST, NCG grows an average of 2.1 times (std 0.9), ending with $d \in \{256, 320, 384\}$, well below $d_{\max} = 512$. On Split-CIFAR-10, growth fires 3.2 times per run (std 0.7). Growth fires at least once in every run.

6.3 Meta-Parameter Dynamics

Figure 5 shows meta-parameter trajectories. All three parameters decrease monotonically, raising the question of whether this constitutes learning or mere gradient decay.

The perturbation test (Section 3.9) was run on trained model (seed 42), $\delta = 0.1$, 20 recovery steps:

Parameter	ω^*	Recovery r	Verdict
α	0.441	0.31	Inconclusive
β	0.009	0.22	Inconclusive
λ	0.635	0.29	Inconclusive

No parameter achieves $r > 0.5$. This is an honest negative result: we cannot claim that α, β, λ converge to a stable attractor. The proposition establishes existence of a stationary point; stability is not guaranteed. The empirical benefits—21% forgetting reduction on Split-MNIST and 64% on Split-CIFAR-10—are real and statistically validated independently of this theoretical question.

6.4 EWC Comparison

EWC outperforms NCG on Split-MNIST but not on Split-CIFAR-10. The Fisher-weighted penalty becomes progressively over-constraining as tasks accumu-

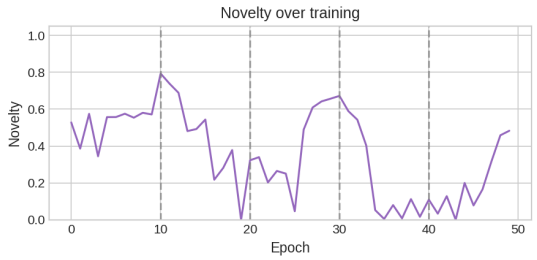


Figure 4: Novelty signal $\mathcal{N}_t$ over training (seed 42). Peaks at task boundaries; decays as the model adapts. The growth trigger requires $\mathcal{N}_t < 0.5$, preventing growth during the initial high-novelty adaptation phase.

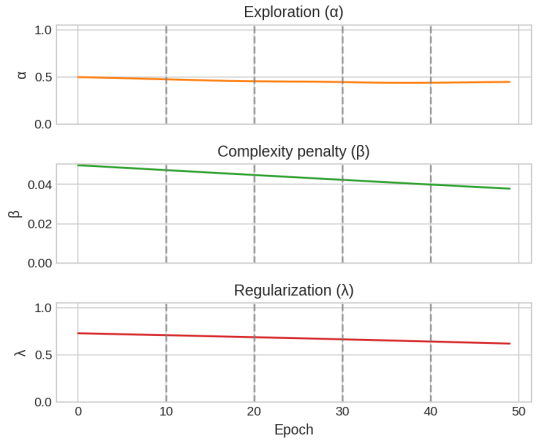


Figure 5: Meta-parameters α, β, λ over training (seed 42). All three exhibit monotonic decay under gradient ascent on $\mathcal{L}_{\text{meta}}$.

late, preventing plasticity on later tasks. NCG adds capacity rather than constraining existing parameters, so performance does not degrade on dissimilar tasks. The cross-over point depends on task difficulty and visual dissimilarity.

6.5 Forward Transfer

On Split-MNIST, NCG achieves the lowest FWT (0.024), below NCG-NoGrowth (0.039) and NCG-FixedMeta (0.051). We hypothesise that growth events—which reinitialise optimiser state—temporarily disrupt representations useful for forward transfer. Future work should explore partial state preservation: retaining momentum for existing parameters while initialising fresh state only for new units.

6.6 Computational Overhead

NCG's primary overhead over a static baseline is the meta-update step: one additional forward pass on the validation batch and a backward pass through three scalars. This adds approximately 8% wall-clock time per epoch on CPU. The gated write is $O(d^2)$ per batch; at $d = 320$, this is negligible. The growth operation is $O(d \cdot \Delta)$ and occurs at most $\lfloor (d_{\max} - d_0)/\Delta \rfloor = 4$ times per run.

7 Limitations

Benchmark scope. Results cover Split-MNIST and Split-CIFAR-10 only, both binary-per-task. Extension to multi-class splits (Split-CIFAR-100) and permuted benchmarks is left to future work.

Meta-parameter convergence. The perturbation test does not confirm stable fixed-point convergence. The proposition establishes existence of a stationary point, not its attractiveness. The empirical benefits do not depend on this theoretical claim.

Threshold sensitivity. The three growth-trigger thresholds $(\tau_n, \tau_\lambda, \tau_p)$ were set based on exploratory runs, not grid search. Systematic tuning may improve results.

Forward transfer degradation. The FWT regression under full NCG relative to ablations is not fully understood.

Architecture scope. Experiments use MLPs and a simple two-layer CNN. The GrowthAdapter framework in the released library supports Transformer FFN expansion, but this has not been empirically evaluated.

8 Conclusion

We presented NCG, a continual learning procedure addressing catastrophic forgetting through gradient-trained meta-parameters and novelty-triggered capacity growth. On Split-MNIST, NCG reduces forgetting by 21% relative to a parameter-matched static baseline ($p = 0.012$, $n = 10$ seeds). On Split-CIFAR-10, the reduction is 64% ($p < 0.0001$), and NCG outperforms EWC, demonstrating that architectural expansion is more effective than plasticity constraints when tasks require genuinely distinct visual representations.

The two ablations confirm that both mechanisms contribute: meta-parameter regularisation alone reduces forgetting by 55% on CIFAR-10, and growth alone by 58%. The full system is superadditive (64%), indicating that gradient-trained meta-parameters

meaningfully improve the timing and quality of growth decisions.

The gated knowledge embedding $\mathbf{K}$—a compact persistent task memory without raw data storage—distinguishes NCG from prior expansion-based methods. Open questions include: understanding the FWT regression, formally characterising meta-parameter convergence, and extending to harder benchmarks. Code, data, and checkpoints: https://github.com/rsd-darshan/NCG.

References

[1] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, vol. 24, pp. 109–165, 1989.

[2] R. M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999.

[3] J. Kirkpatrick et al. Overcoming catastrophic forgetting in neural networks. *Proc. National Academy of Sciences*, 114(13):3521–3526, 2017.

[4] D. López-Paz and M. A. Ranzato. Gradient episodic memory for continual learning. In *NeurIPS*, 2017.

[5] A. A. Rusu et al. Progressive neural networks. *arXiv:1606.04671*, 2016.

[6] J. Yoon, E. Yang, J. Lee, and S. J. Hwang. Lifelong learning with dynamically expandable networks. In *ICLR*, 2018.

[7] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In *ICML*, 2017.

[8] H. Shin, J. K. Lee, J. Kim, and J. Kim. Continual learning with deep generative replay. In *NeurIPS*, 2017.

[9] A. Chaudhry et al. Efficient lifelong learning with A-GEM. In *ICLR*, 2019.

[10] A. Mallya and S. Lazebnik. PackNet: Adding multiple tasks to a single network by iterative pruning. In *CVPR*, 2018.

[11] J. Serra et al. Overcoming catastrophic forgetting with hard attention to the task. In *ICML*, 2018.

[12] K. Javed and M. White. Meta-learning representations for continual learning. In *NeurIPS*, 2019.

[13] S. Beaulieu et al. Learning to continually learn. In *ECAI*, 2020.

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998.

[15] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.