EXTRACTING, REMOVING AND SHRINKING DOMAINS FROM ONE JOINTLY TRAINED NETWORK

Anonymous authors
Paper under double-blind review

ABSTRACT

Deploying language models for different sets of text domains and memory budgets usually means training a separate model for each. Dropout already trains many subnetworks inside one network but uses only their average. We ask whether single subnetworks can be deployed on their own, for one domain, several domains or a smaller budget, without a router or further training. We introduce region dropout, which assigns each domain a region of every feed-forward layer: within a region, the drop rate rises with a unit's position, so that early units form smaller models; across regions, other domains' regions are included and trained on each domain's data with a probability that rises during training, which we call exposure. Without exposure, combined models containing every unit of an arithmetic member solved 5 and 53 of 192 problems at 30M and 124M parameters that the member solved completely; with exposure, they solved all 192, and combined members at full width stayed within 0.11 nats of their single-domain members on external text and within 0.008 nats at 1.3B parameters, while the code member lost 0.06 to 0.13 nats against dense models. In a matched study, neither joint inclusion alone nor training regions on other domains' data alone kept the largest loss increase from combining below 0.10 nats. Weighting the regions of models trained without exposure by a domain posterior, as in expert mixtures, gave 0.03 to 0.12 nats lower text loss than exposure at 5.6 times the scoring time but solved at most 60 of the 192 problems. Removing a region reduced the exact answers on its task from 188 and 187 to 1 and 0 of 192 and left the other domains within 0.003 nats; retraining on the task restored most answers. A code member exported at 75% width kept 32.3M of 51.3M parameters at 0.05 nats above full width; narrowed combinations kept fewer exact answers. One network trained with region dropout thus provides models for each subset of its domains, without a domain, or for single domains at smaller width, on text from these domains.

1 INTRODUCTION

Language models are often needed for different combinations of text domains and at different sizes: a code assistant for one team, a general model for another, a smaller model where memory is short. Training a separate specialist or a separate mixture for every combination and size repeats the cost of training each time, and removing a domain from a jointly trained model requires training again (Bourtoule et al., 2021).

Dropout trains an exponential family of thinned subnetworks and deploys their average (Srivastava et al., 2014). Nested dropout removes units from the end of an ordering so that prefixes of a layer remain usable (Rippel et al., 2014), and MatFormer, Flextron and LayerDrop obtain many widths or depths from one training run (Devvrit et al., 2023; Cai et al., 2024; Fan et al., 2019). These methods vary capacity along one axis of a network.

Modular language models separate domains instead. DEMix trains one feed-forward expert per domain and weights experts by an estimated domain posterior (Gururangan et al., 2022); Branch-Train-MiX learns a router in a further finetuning stage (Sukhbaatar et al., 2024); FlexOlmo trains each expert next to a frozen public model so that experts can later be combined through a router (Shi et al., 2025). Combination there relies on weights, a router or an anchor.

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.

We ask whether individual members of a dropout family, single domain regions at full or reduced width and their unions, can be deployed as standalone models without further training, and how the mask distribution relates to which members work. The answer is not obvious. Dropout is used for its average and was introduced to keep units from co-adapting, whereas a union of regions has to work as one network; and a region trained on its own domain has never been active on the data of the domains it is combined with.

We study this with region dropout, which treats dropout as a distribution over subnetworks and structures it at two levels. Every feed-forward layer is divided into one region per domain and a shared pool. Within each region, the drop rate rises with a unit's index. Over regions, the regions of other domains are included, on data from one domain, with a probability that we keep at zero, fix, or raise during training, which we call exposure. A member is exported by copying its units into smaller matrices. For example, one network trained on three text sources and code yields a code model, a text-and-code model and narrower versions of each (Figure 1); a region is assigned to its domain but can also be updated on other domains' data.

Our contributions are:

- Region dropout, a dropout scheme over domain regions whose members can be exported as standalone models of different domain sets and widths.
- An examination of combined members trained with and without exposure, on development data and a frozen external panel at 30M, on a task with exact answers at 30M and 124M, and on held-out test data at 1.3B parameters.
- An examination of which narrower members work, comparing the rising drop rate with a flat rate, with random units and with a common post-training ranking, and single-unit draws with whole-prefix draws.
- An examination of what removing a region suppresses and whether relearning undoes it.

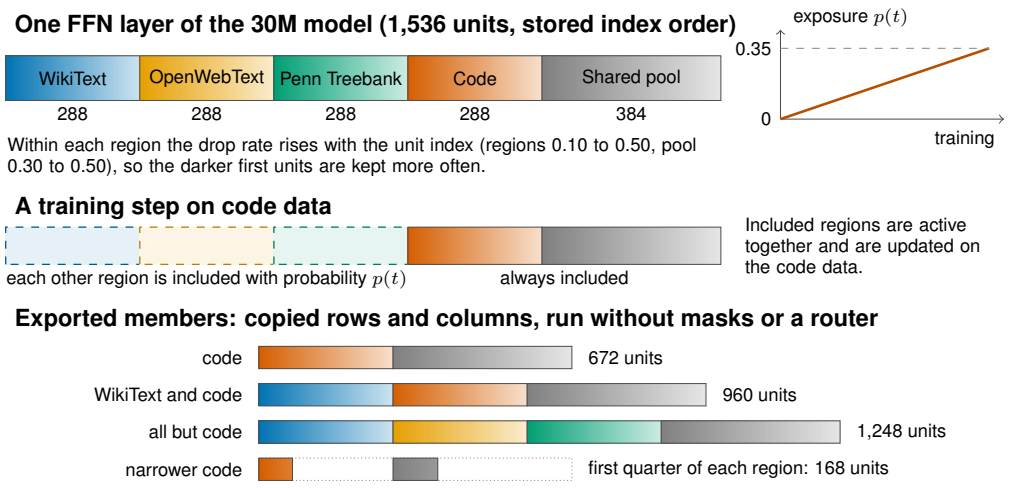


Figure 1: Region dropout for one FFN layer of the 30M model. Shading marks the drop rate rising with the unit index; the inset shows the exposure schedule of the ramp. On a batch from one domain, each other region is included with probability $p(t)$, and all included regions are updated on that batch. Members are exported by copying rows and columns, drawn to scale in FFN units per layer. The figure specifies architectures, not their quality.

2 Related work

Dropout and ordered capacity. Dropout trains an exponential family of thinned networks and approximates their average at test time with a single network (Srivastava et al., 2014). Nested dropout removes nested sets of units from the end of an ordering, so that earlier units carry more of the representation (Rippel et al., 2014); ordered dropout uses the same principle to serve clients of different

capacity from one model (Horvath et al., 2021). Slimmable and once-for-all networks, MatFormer and Flextron obtain many widths from one training run, and LayerDrop many depths (Yu et al., 2019; Cai et al., 2020; Devvrit et al., 2023; Cai et al., 2024; Fan et al., 2019). Structured dropout also drops groups of units at once, such as entire feature maps (Tompson et al., 2015) or contiguous blocks of a feature map (Ghiasi et al., 2018), and curriculum dropout changes the retention probability over training (Morerio et al., 2017). MatFormer additionally mixes granularities across layers to extract models that were not trained explicitly. These methods vary capacity along one axis of one network or drop groups of units for regularisation; none of them assigns parts of a layer to domains.

Domain experts and their combination. DEMix replaces each FFN layer with one expert per domain, updates each expert only on its own domain, and mixes experts at inference with weights from an estimated domain posterior (Gururangan et al., 2022). Branch-Train-Merge trains expert language models separately and ensembles or averages them (Li et al., 2022); Branch-Train-MiX merges their FFNs into a mixture of experts and learns token-level routing in a further finetuning stage (Sukhbaatar et al., 2024). FlexOlmo trains each expert next to a frozen public expert, so that independently trained experts can later be combined through a domain-informed router (Shi et al., 2025). In these methods, combination relies on output weights, a router, an anchor or further training. Here, a combined member sums the outputs of its regions without weights, as a wider FFN. Parameters shared across domains also have a cost: multilingual models trade positive transfer against capacity dilution (Conneau et al., 2020), and negative interference lowers performance on high-resource languages (Wang et al., 2020).

Localising and removing capabilities. Gradient routing masks gradients by data point so that chosen subregions of a network learn chosen data, which can then be removed by ablation (Cloud et al., 2024). Selective gradient masking lets examples of a target domain update only their dedicated parameters (Shilov et al., 2025). After unlearning, finetuning on other, accessible facts can recover most of the removed accuracy (Deeb & Roger, 2024). In a related test, we retrain a model on the removed task itself after removing a region.

Position of this work. Prior work varies capacity along one axis of one network, or combines domain experts through weights, a router or an anchor. It does not determine which members of a single dropout family, over domain regions and widths, can be deployed without either, because networks of variable width have no domain regions, and expert mixtures combine their experts through weights, a router, an anchor or further training rather than as one unweighted layer. We test this by training one network whose drop-rate distribution covers both the index within each region and the inclusion of other domains' regions.

3 METHOD

Dropout as a distribution over subnetworks. A dropout mask m keeps a subset of units and thereby selects a subnetwork $f_{\theta,m}$. On data from domain d at training step t, training follows the gradient of

$$\mathbb{E}_{x\sim P_d}\,\mathbb{E}_{m\sim\mathcal{D}_{d,t}}\big[L\big(f_{\theta,m}(x)\big)\big], \tag{1}$$

so the mask distribution $\mathcal{D}_{d,t}$ determines how often, and on which data, each subnetwork is trained. Standard dropout keeps every unit independently with the same probability, so all subnetworks with the same number of units are equally likely, and the trained network is used as an approximation of their average (Srivastava et al., 2014). Here, $\mathcal{D}_{d,t}$ is structured at two levels, within a region of units and over entire regions, so that the distribution places its training on chosen subnetworks, the members. We call this scheme region dropout.

Regions and members. Before training, the units of a layer are divided into one region per domain and a shared pool; here, these are the hidden units of the feed-forward (FFN) layers of a transformer. A region is assigned to its domain but not exclusive to it, because other domains can also update it. Units keep fixed indices within each region. A member $M(C,k)$ is specified by a set of domains C and a prefix length k_r for the pool and for the region r of each domain in C; it keeps the first k_r units of these regions, with the same indices in every layer. A member that combines several regions therefore contains every unit of each single-domain member it combines, and leaving out a region

yields the member of the remaining domains. The regions of a member contribute to the FFN output without input-dependent weights, and a member can be exported as a smaller dense network.

Within a region: profile. Unit i of region r is kept with probability $q_r(i)$, which does not increase with i, so earlier units are kept more often than later ones. A flat profile makes the units of a region interchangeable, as in standard dropout; an index-dependent profile distinguishes them by index, as in nested dropout (Rippel et al., 2014). The profile fixes how often each unit is trained.

Within a region: coupling. The profile fixes only the probability with which each unit is kept. Mask distributions with the same probabilities can still differ in two ways. Units can be drawn independently, or through one uniform variable U per region that keeps all units with $q_r(i) > U$; this draw always keeps a prefix, so whole prefixes are drawn instead of single units, as in nested dropout (Rippel et al., 2014). A draw can also be made for every token and layer, or shared across the tokens of a batch and across layers; a draw shared across layers keeps the same indices in every layer, as a member does.

Over regions: exposure. On data from domain d, the pool and the region of d are always eligible. Each other region is included entirely with probability $p(t)$, its exposure, and an included region receives the gradients of domain d; a unit of such a region is kept with probability $p(t)\, q_r(i)$. At $p = 0$, each region is trained only on its own domain, next to the shared pool, and no training step lets two domain regions be active together. At $p > 0$, the units of a single-domain member are also trained on other domains. Exposure may change over training. Profile and coupling determine how often and how jointly the units of a region are trained; exposure determines whether regions are trained on the same data.

4 Experimental setup

Models and data. The main model has six layers, hidden width 384 and FFN width 1,536 (about 30M parameters), with a 384-unit shared pool and four 288-unit regions. It is trained on WikiText-103, OpenWebText, Penn Treebank and Python source code with a nominal 20M tokens per domain; each optimiser update accumulates one equally weighted microbatch from every domain. A variant widens the FFN to 16 times the hidden width (51.3M parameters), so that the FFN holds most of the parameters and physical export changes model size more; with generated two-digit arithmetic as a fifth domain, this variant carries the removal experiments. A loss difference alone does not show whether a member still solves a task; a second four-domain mixture of WikiText-103, a larger OpenWebText, CodeParrot and generated arithmetic therefore allows exact-answer scoring, at 30M and 124M parameters with a nominal 800M tokens. To ask whether combined members behave alike at a much larger width, one pair of 1.3B-parameter models was trained on web, science, question-answer and code data. Appendix A lists the settings of the 30M to 124M models, Appendix U those of the 1.3B pair, and Table 2 maps every experiment to its recipe, size, number of models and data.

Mask distribution. The profile is linear in the unit index. In a region of n units,

$$q_r(i) = 1 - a - (b - a)\,\frac{i-1}{n-1}, \qquad i = 1, \ldots, n, \tag{2}$$

so the drop rate rises from a at the first unit to b at the last, with $(a, b) = (0.10, 0.50)$ in domain regions and $(0.30, 0.50)$ in the pool, giving mean keep probabilities of 0.70 and 0.60. These endpoints were set by hand early in development and held fixed in all reported models; the pool fraction, region sizes and token budgets were likewise carried over from early development runs. By default, units are drawn independently for every token, unit and layer. Each foreign region receives one exposure gate per layer, shared over the examples and tokens of that layer. Kept units are scaled by $1/q_r(i)$, given that their region is included, so that expected activations match the full network (Srivastava et al., 2014).

Exposure values. Constant exposure at $p = 0.35$ was the first nonzero setting we trained. It is the constant setting in the 30M studies, and every other exposed model uses a linear ramp from 0 to 0.35 over training; Appendix N compares $p = 0$, 0.35, 0.70 and 1 at 30M with one model per value. After

the constant setting showed a cost for single-domain members, three schedules were compared on development data under a protocol dated before any of these runs finished: a decreasing ramp from 0.35 to 0, holding 0.35 for three quarters of training before decreasing to 0, and the increasing ramp from 0 to 0.35, planned as a control arm. The increasing ramp was selected after this comparison and then evaluated with further seeds; we call it the ramp; scheduling a dropout probability over training follows curriculum dropout (Morerio et al., 2017). Training without exposure ($p = 0$) serves as the reference recipe for combined members; it trains each region only on its own domain, as the experts of DEMix are trained (Gururangan et al., 2022).

Compared variants. Prefixes of a region are compared with the same number of random units and with the units of largest outgoing-weight norm, a variant of the weight-norm rankings used to remove whole units after training (Li et al., 2017). A second 30M model uses a flat profile with drop rates 0.3 in the domain regions and 0.4 in the pool, the means of the curves. For coupling, five mask distributions with the same keep probabilities are compared: independent draws for every token; independent draws shared across the tokens of a batch; prefix draws shared across the tokens of a batch; and the last two shared also across layers (Appendix K). A further recipe draws one prefix length per region for all tokens and layers of a microbatch from a fixed set of fractions; its keep probabilities differ from Eq. 2 (Appendix O). For models trained without exposure, we also multiply the outputs of the domain regions by a fixed factor, chosen on the first validation half from 0.1, 0.2, 0.3, 0.5 and 0.7, a set fixed before the evaluation, by the mean loss over the four domains, and evaluated on the second half (Appendix X); in a second comparison, we weight the region outputs at each token by the posterior of each domain given the preceding tokens, computed from the single-domain members as in DEMix, and with constant weights of 1/4 (Appendix X.1). Two further arms separate the two effects of exposure: foreign regions active but not updated on other domains' data, or updated without joint activity (Appendix X.2).

Export and measured quantities. At evaluation, stochastic masks and scaling are disabled. A member is exported by copying the selected input-weight rows and output-weight columns of every FFN layer into smaller matrices, so it runs without masks or a router. Export shrinks only the FFN; embeddings, attention and the output head are copied unchanged. Numerical agreement between masked and exported members is checked for the benchmarked members (Appendices F and G). Loss is mean next-token cross-entropy in nats. The composition penalty of domain d in a member with domain set $C \ni d$ is

$$R_d(C) = L_d\big(M(C)\big) - L_d\big(M(\{d\})\big), \tag{3}$$

where both members use the same shared and own-region indices. A penalty near zero means that adding regions barely changes a domain's loss relative to its single-domain member; it says nothing about the absolute loss. We count cases with $R_d > 0.10$ nats, a descriptive tolerance carried over from our earlier development analyses, not a statistical or universal bound.

Evaluation data. Validation streams of the main mixture were split into two token-position halves, both of which were used during method development; they are development data, not an untouched test set. We therefore also scored a panel of 512 documents from Wikipedia, C4 web, C4 news-like and CodeParrot, routed to the four trained regions by a rule given in Appendix Q, with checkpoints, members and documents fixed before scoring. Arithmetic members are scored by exact answers on 192 equations whose combinations of operands and operator never occur in training; the task and its equations were chosen within the same validation setting, and for 85 of the 89 additions among them the reversed operand order does occur (Appendix M). The 1.3B models are scored after full training on a held-out test split of about 10M tokens per domain, assigned by source group and screened against training and development data by exact hashing and near-duplicate detection; whole documents are scored in windows of at most 1,024 tokens. Their members are scored with deterministic masks, and nine of them are also exported and compared with the masked model. For this pair, the primary quantity is the mean positive penalty, the equally weighted mean of $\max(R_d(C), 0)$ over all 28 cases (C, d) with $d \in C$ in combinations of two or more regions.

Replication and uncertainty. The replication unit is the trained model. Intervals over evaluation sequences or documents condition on the trained models and do not measure variation between training runs. Results from 30M to 124M parameters come from one to five models per setting; the

1.3B results come from one model per recipe after full training. Two further screens are reported in the appendix: adding a domain to a trained model with replay (Appendix W) and the recipe on a convolutional network (Appendix R).

5 RESULTS

5.1 COMBINED MEMBERS

A combined member contains every unit of its single-domain members, and can still fail on their tasks. Unless a paragraph states otherwise, each comparison here holds architecture, drop-rate curves, data and token budget fixed and changes only exposure. The composition penalty of a domain is its loss in a combined member minus its loss in the single-domain member with the same shared and own indices.

Development data. In the 30M model trained without exposure, code lost 8.428 nats when all four regions ran at full width, relative to the code member. With constant exposure $p = 0.35$ the code penalty was 0.011 nats, and with the ramp it was 0.025 to 0.033 nats over five seeds. Over all four domains, the largest penalty with the ramp was 0.040 nats.

An arithmetic region in combination. The second mixture includes generated two-digit arithmetic, so that members can be scored by exact answers on 192 held-out equations (Table 32 in Appendix V). Without exposure, the arithmetic member answered all 192 at 30M and at 124M parameters, and the all-region member of the same model answered 5 at 30M and 53 at 124M. With the ramp, both members answered 192 in all four models, and the code penalty was at most 0.009 nats, against 9.872 and 11.521 nats without exposure. In an exploratory diagnostic that placed Python text before each equation, the arithmetic member answered 0 or 1 of 192 equations with the ramp and 136 and 134 without exposure; the ramp outputs continued the Python text (Appendix Y).

Frozen external panel. Three 30M models per recipe were scored on 512 documents from Wikipedia, C4 web, C4 news-like and CodeParrot, routed to the four trained regions, for all 15 full-width combinations; checkpoints, combinations and documents were fixed before scoring (Table 1). The largest composition penalty was 7.877 nats without exposure and 0.108 nats with the ramp. Of 84 cases in combinations of two or more regions, 59 exceeded 0.10 nats without exposure and 2 with the ramp.

Table 1: Composition penalties of 30M models on the frozen external panel. Top: largest penalty of each trained model over all 15 full-width combinations, the largest over the three models of a recipe, and the number of cases above 0.10 nats; the counts include the 12 single-domain cases, whose penalty is zero by definition, so over the 84 cases in combinations of two or more regions they are 59 and 2. Penalties in nats. The table does not show reduced widths. Document-level results and intervals are in Appendix Q.

Recipe	Parent 0	Parent 1	Parent 2	Maximum	> 0.10
No exposure	7.500	7.877	7.432	7.877	59/96
Ramp	0.091	0.108	0.105	0.108	2/96

1.3B parameters. One model per recipe was trained at 1.3B parameters on web, science, question-answer and code data and evaluated after its full 99,183 updates on a held-out test split for the 28 cases of the 11 combinations of two or more regions. The largest composition penalty was 0.549 nats without exposure and 0.008 nats with the ramp; 18 of 28 cases exceeded 0.10 nats without exposure and none with the ramp. The primary endpoint, the mean positive penalty over the 28 cases, was defined before training and adopted for this pair after a development trial and before any test data were scored; it was 0.200 nats without exposure and 0.001 nats with the ramp, a paired difference of -0.199 nats (95% bootstrap interval $[-0.200, -0.197]$ over 37,532 source groups, conditional on one pair of trained models). Single-domain members of the ramp model were 0.007 to 0.015 nats above those of the model without exposure (Appendix U).

Rescaling without exposure. The factor chosen for the region outputs was 0.2 in the main 30M model and 0.1, the smallest value in the set, in both models of the second mixture. On the second validation half, it reduced the code penalty from 8.448 to 0.937 nats in the main model and from 9.879 to 1.033 and from 11.576 to 3.020 nats at 30M and 124M, against 0.022, 0.006 and 0.005 nats for the ramp models on the same data; the rescaled all-region member answered none of the 192 equations at either size. Scaling only the regions of the domains other than the evaluated one by 0.1, which requires knowing the input's domain, gave code penalties of 0.032, 0.067 and 0.796 nats and 192 answers at both sizes.

Weighting by a domain posterior. Weighting the regions of the models trained without exposure by a domain posterior computed from the preceding context reached 0.032 to 0.118 nats lower loss than the unweighted ramp members on the four sources of the external panel (three models per recipe), at five forward passes per token instead of one and 5.6 times the scoring time. On the 192 equations, these weighted members answered 60 and 57 at 30M and 124M, against 192 for the unweighted ramp members. Constant weights of 1/4 gave 0.281 to 0.792 nats higher loss than the posterior weighting and 0.221 to 0.674 nats higher than the unweighted ramp members on all four sources.

Level and schedule of exposure. With constant exposure, $p = 0$, 0.35, 0.70 and 1 gave code members of 2.161, 2.316, 2.295 and 4.468 nats and code penalties of 8.428, 0.011, -0.043 and -2.077 nats, with one model per value. Two further seeds at $p = 0.35$ gave code members of 2.277 and 2.345 nats, a range that includes the model at $p = 0.70$. The schedule protocol used two criteria: a code member below the 2.310 nats of a model trained on code alone, and a code penalty of at most 0.10 nats. The ramp met both on its two initial seeds and on three further seeds evaluated after its selection, with code members of 2.222 to 2.279 nats. The decreasing ramp met both on two seeds (penalties 0.074 to 0.082 nats); holding 0.35 before decreasing missed the code criterion on one of two seeds.

Joint activity and foreign updates. With the same three seeds, initial states and data order in all four arms, the largest penalty was 0.032 to 0.035 nats with the ramp, 0.285 to 0.335 with joint activity only, 0.546 to 0.642 with foreign updates only and 7.94 to 8.97 without exposure; within this matched study, penalties below 0.10 nats occurred only with both (Appendix X.2).

Cost for single-domain members. The ramp code member reached 2.222 to 2.279 nats over five seeds, 0.06 to 0.13 nats above three unpartitioned models trained on the same data and budget (2.153 to 2.165). Against five models trained on the same 20M code tokens alone, the ramp code member was 2.251 ± 0.026 against 2.288 ± 0.020 nats, a difference of -0.037 nats with Welch 95% interval $[-0.071, -0.003]$, computed after the results were known; total training compute was not matched. With constant exposure, the corresponding difference was -0.003 ± 0.028 nats over three seeds per cell (Appendix N). With four times the code tokens and one seed, a model trained on code alone reached 1.711 nats and a code member extracted from a model trained without exposure 1.744; no ramp model was trained at this budget.

5.2 Narrower members

Order within a region. The code member of the 30M model with constant exposure was cut to equal fractions of its pool and code region, keeping the first units, those with the largest outgoing-weight norm, or random units (mean of five draws). The first units gave 2.359, 2.502, 2.976 and 3.534 nats at 504, 336, 168 and 67 units per layer; random units gave 2.438, 2.795, 3.380 and 3.775 nats. The largest outgoing weights gave losses 0.001 to 0.057 nats above the first units. On WikiText, the first units were 0.075 to 0.349 nats below random units.

Flat drop rate. The 30M model with a flat drop rate was otherwise trained identically. Cutting only the code region and averaging over cut points, the first units were 0.009 nats above random units in this model and 0.059 nats below them in the model with the curve. Cutting the code region to 5 to 15% of its units gave $+0.005$ against -0.118 nats, and cutting both regions of the code member gave $+0.015$ against -0.222 nats (first minus random units; one random subset per cut, shared by both models).

Two widths per member. Figure 2 in Appendix C varies the retained fractions of the shared pool and of the code region independently. Of the 25 cells, 14 were Pareto-optimal, meaning that no cell with the same or fewer units had lower loss, and 11 of these kept partial width in both regions. At 336 units, half of each region gave 2.502 nats; at 384 units, a quarter of the pool and the full code region gave 2.676 nats.

Physical export. In the model with the FFN widened to 16 times the hidden width (51.3M parameters), a member exported with 75% of its shared pool and 75% of its code region had 32.3M parameters. Over five seeds it ran at 1.530 to 1.553 times the full model's batch-16 forward throughput on an RTX 2080 Ti and 1.57 to 1.58 times on an A100, and was 0.042 to 0.049 nats above the full-width code member. On the A100, all exported members ran at 0.99 to 1.07 times the full network at batch 1 and 0.92 to 1.01 times in generation without a key-value cache. At 75% width, the composition penalty was 0.097 to 0.116 nats for all four regions and at most 0.015 nats for code and WikiText (Appendix G.1). In two models with an arithmetic region, exported at 75% width, the arithmetic member answered 186 and 191 of 192 equations (191 and 192 at full width) and the all-region member 116 and 145 (188 and 187; Appendix M.1).

Prefix coupling at equal marginals. Five mask distributions with the same retention probability for every unit were trained with the ramp, three models each. Among masks shared across the tokens of a batch, drawing one prefix per region instead of single units lowered the loss of the 168-unit code member by 0.238 nats (95% t interval $[-0.350, -0.127]$ over three model pairs) and raised that of the 672-unit member by 0.059 nats ($[+0.042, +0.076]$). A prediction dated before these runs finished expected 0.05 to 0.30 nats at 168 and 384 units, which the results met, and at most 0.02 nats from sharing masks across tokens, which they did not ($+0.140$ at 672 units). Drawing prefixes instead of single units changed the code composition penalty by at most 0.011 nats.

Prefix coupling at full budget. A second comparison drew one prefix length per region for all tokens and layers, with a retention distribution that differs from the single-unit recipe (Appendix O), on two seeds and all four domains. Against the single-unit recipe it changed the loss by -0.39 to -0.59 nats at 168 units, -0.21 to -0.34 at 384 units and $+0.02$ to $+0.08$ at 672 units; all 24 paired intervals excluded zero. Composition penalties changed by at most 0.010 nats.

5.3 Members without a region

Removing the arithmetic region. In the main mixture with arithmetic added as a fifth domain, two models with the widened FFN were trained with the ramp. Their all-region members answered 188 and 187 of the 192 held-out equations, and their arithmetic members 191 and 192. Removing the arithmetic region left 1 and 0 correct answers, while the loss on each of the four other domains changed by at most 0.003 nats; only 45 and 24 of the 192 outputs were still well-formed integers, so the failure includes the answer format (Appendix M).

Relearning. Training the member without the arithmetic region on arithmetic alone restored 177 and 179 correct answers after 2,002,944 tokens. Over the same training, the loss on the four other domains rose by 0.56 to 1.53 and 0.35 to 1.33 nats in the two models.

Loss of the removed domain. In the main mixture, removing a domain's region raised that domain's loss by 0.523 to 1.576 nats over fourteen seed and domain pairs of five ramp models. In the 1.3B ramp model after full training, removal raised the removed domain's loss by 0.132 to 0.372 nats, and the other domains changed by less than 0.011 nats.

6 Discussion

Which members can be deployed. Dropout is usually deployed through its average and was introduced against co-adaptation of units. In our models, combined members worked with equal weights when training had let their regions be active together on each other's data; neither part alone kept the largest penalty below 0.10 nats in a matched comparison. Without exposure, the all-region member failed on code and on arithmetic although it contained every unit of each single-domain member; one factor on all regions did not restore it, and damping the other domains' regions

restored the equations only with the input's domain known, the information that domain-weighted mixtures and routers supply. Regions of one model can thus be combined in two ways: by training them together, which lets them work with equal weights in one pass, or by weighting them at each token with a domain posterior, which gave lower loss on external text for models trained without exposure at five forward passes per token and recovered fewer exact answers. With exposure rising to 0.35, the largest penalty was at most 0.108 nats in every evaluation of full-width members when all domains were trained from the start. The price was paid by single-domain members: the ramp code member ended 0.06 to 0.13 nats above dense models of the same budget, and between $p = 0$ and $p = 0.35$ single-domain loss rose while the penalty fell, the trade-off of transfer and capacity dilution in multilingual models (Conneau et al., 2020).

Which narrower members work. With a flat drop rate, prefixes of a region did not beat random units; with the rising curve they did, most at deep cuts. We read this as the curve creating the order, although each arm is one trained model. Separate widths for the pool and each region added members between proportional cuts. Drawing whole prefixes instead of single units, at equal marginals, moved quality from wide to narrow members and changed the composition penalty by at most 0.011 nats. With single units, prefixes carried although a particular member was almost never drawn during training, like models that MatFormer extracts without training them explicitly (Devvrit et al., 2023); drawing whole prefixes improved narrow members further. Both observations point to which units were active together, not to whether a member was drawn (interpretation). At 75% width, an arithmetic member kept 186 and 191 of 192 answers, the all-region member 116 and 145; in the four-domain catalogue, all-region penalties at that width reached 0.097 to 0.116 nats.

Removal and relearning. Removing a region disrupted its task and left other domains within 0.003 nats. Relearning on the removed task alone for 2M tokens restored 177 and 179 of 188 and 187 answers and raised the loss on the other domains by up to 1.53 nats. We read this as a stronger attack on removal than relearning that also keeps the other domains; after unlearning, finetuning on other facts also recovered most removed accuracy (Deeb & Roger, 2024). Whether removal holds under training that also protects the other domains, for example with replay (Rolnick et al., 2019), was not tested; a replay screen for adding a domain is in Appendix W.

Boundaries of the recipe. In a diagnostic designed after the main results, ramp arithmetic members continued Python text placed before an equation instead of answering it; members trained without exposure answered most (Appendix Y). On a CIFAR convolutional network, prefixes were below arbitrary units of the same size (Appendix R), a boundary of our recipe, not of ordered dropout, reported for such networks under its own recipe (Horvath et al., 2021). With four times the code tokens, training on code alone overtook the extracted member; in multilingual pretraining, added data from other languages also helped at small target budgets and hurt at larger ones (Chang et al., 2024). Training after export answers a different question: without further training, exported members were 0.73 to 3.09 nats better than activation-informed pruning of a dense model to the same width, whereas with 1M tokens of code training the pruned model was 0.078 to 0.419 nats better at all four widths (Appendix H.2). Export is therefore a useful option when no further training is possible, in the comparisons tested (token budgets by number of members in Appendix F.1).

Limitations. Models range from 30M to 124M parameters with one to five models per setting, and one 1.3B model per recipe, whose exported members exceeded the FP32 tolerance in 2 of 26 checks and agreed with the masked model in FP64 in all 26. After the primary endpoint was adopted and before the final 1.3B evaluation was frozen, the first 32 test documents per domain were scored for the full member and the member without the code region, without composition. Cached generation was not measured. Apart from program synthesis, on which 1.3B members at an intermediate checkpoint solved at most 7 of 257 problems (Appendix U.1), exact answers were tested only on generated arithmetic, and removal on one such task with one relearning budget. Prefix coupling and posterior weighting were not tested at 1.3B, and no further schedules were searched.

What the results support. One network trained with region dropout yielded standalone members for each subset of its domains, without a domain, and for single domains at smaller width. The results do not show that combined members solve tasks that mix their domains, that removal prevents relearning, or that narrowed members combine as well as full-width ones.

ETHICS STATEMENT

Reconfigurable models can support selective deployment, but the choice of capabilities made available to a recipient is also a governance question. Our experiments distinguish structural region removal, measured suppression of the removed task and its recovery under relearning; relearning on the removed task alone restored most of its answers, and combining removal with an unlearning objective (Appendix I) did not prevent recovery. Deployments should communicate the tested objectives, retained utility and attack budgets rather than imply irreversible deletion from a corpus-loss score. The marginal-risk framing of Kapoor et al. (2024) motivates assessing such outcomes in their deployment context. No new human-subjects data were collected. The experiments use existing public corpora and synthetically generated text; public web text may contain personal information.

REPRODUCIBILITY STATEMENT

The source package contains data preparation, training and evaluation scripts, manuscript sources, configuration records, and anonymised result files behind the reported numbers. Verification scripts recompute reported quantities and check their comparison references and manuscript entries. Checkpoints are not included in the supplementary archive; their identities and training recipes are recorded for reproduction. The external evaluation additionally records its frozen protocol, source revisions, screening rules, per-document losses and selected indices (Appendix Q); raw external text and reconstructive token caches are not redistributed. Table 26 is generated with a checkpoint-provenance check. Candidate-specific envelope reanalysis stores source hashes and matching checks separately (Appendix D); incomplete historical cross-file provenance is marked. Reproducing the historical prefix checkpoint requires the explicit legacy layerwise sampler and marginal scaling, not the corrected training defaults. The original configuration catalogue is retained in Appendix T; its ordering flags and normalised gap columns are descriptive diagnostics.

Ramp checkpoints store the initial exposure, $p = 0$, in their saved configuration; the ramp to 0.35 is given by the recorded training commands. The rescaling and Python-context analyses (Appendices X and Y) record their committed protocols, manifests and checkpoint hashes.

STATEMENT ON LLM USAGE

LLM assistants from Anthropic and OpenAI were used under author direction for editing and revising the text for clarity and consistency, for literature search, including the identification of related work in adjacent fields, and for checking derivations and proofs. They also assisted with hypothesis development, experiment design, implementation, numerical and statistical checks, and consistency checks between text and result files. All reported numbers were produced by the authors' analysis code from versioned result files. The authors reviewed all AI-assisted content and take full responsibility for it.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. arXiv:2108.07732.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy*, pp. 141–159, 2021. doi: 10.1109/SP40001.2021.00019.

Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://arxiv.org/abs/1908.09791.

Ruisi Cai, Saurav Muralidharan, Greg Heinrich, Hongxu Yin, Zhangyang Wang, Jan Kautz, and Pavlo Molchanov. Flextron: Many-in-one flexible large language model, 2024. arXiv:2406.10260.

Tyler A. Chang, Catherine Arnett, Zhuowen Tu, and Ben Bergen. When is multilinguality a curse? language modeling for 250 high- and low-resource languages. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 4074–4096, 2024. doi: 10.18653/v1/2024.emnlp-main.236.

Alex Cloud, Jacob Goldman-Wetzler, Evzen Wybitul, Joseph Miller, and Alexander Matt Turner. Gradient routing: Masking gradients to localize computation in neural networks, 2024. URL https://arxiv.org/abs/2410.04332. arXiv preprint arXiv:2410.04332.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8440–8451, 2020. doi: 10.18653/v1/2020.acl-main.747.

Aghyad Deeb and Fabien Roger. Do unlearning methods remove information from language model weights?, 2024. URL https://arxiv.org/abs/2410.08827. arXiv preprint arXiv:2410.08827.

Devvrit, Sneha Kudugunta, Aditya Kusupati, Tim Dettmers, Kaifeng Chen, Inderjit Dhillon, Yulia Tsvetkov, Hannaneh Hajishirzi, Sham Kakade, Ali Farhadi, and Prateek Jain. Matformer: Nested transformer for elastic inference, 2023. URL https://arxiv.org/abs/2310.07707. NeurIPS 2024.

Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout, 2019. arXiv:1909.11556.

Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V. Le. DropBlock: A regularization method for convolutional networks. In *Advances in Neural Information Processing Systems*, 2018.

Suchin Gururangan, Mike Lewis, Ari Holtzman, Noah A. Smith, and Luke Zettlemoyer. DEMix layers: Disentangling domains for modular language modeling. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5557–5576, 2022. doi: 10.18653/v1/2022.naacl-main.407.

Samuel Horvath, Stefanos Laskaridis, Mario Almeida, Ilias Leontiadis, Stylianos I. Venieris, and Nicholas D. Lane. FjORD: Fair and accurate federated learning under heterogeneous targets with ordered dropout. In *Advances in Neural Information Processing Systems*, 2021.

Sayash Kapoor, Rishi Bommasani, Kevin Klyman, Shayne Longpre, Ashwin Ramaswami, Peter Cihon, Aspen K. Hopkins, Kevin Bankston, Stella Biderman, Miranda Bogen, et al. Position: On the societal impact of open foundation models, 2024. URL https://proceedings.mlr.press/v235/kapoor24a.html. Proceedings of the 41st International Conference on Machine Learning (ICML), PMLR 235:23082-23104.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient ConvNets. In *International Conference on Learning Representations*, 2017.

Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A. Smith, and Luke Zettlemoyer. Branch-train-merge: Embarrassingly parallel training of expert language models, 2022. URL https://arxiv.org/abs/2208.03306. arXiv preprint arXiv:2208.03306.

Nathaniel Li, Alexander Pan, Anjali Gopal, Summer Yue, Daniel Berrios, Alice Gatti, Justin D. Li, Ann-Kathrin Dombrowski, Shashwat Goel, Gabriel Mukobi, et al. The wmdp benchmark: Measuring and reducing malicious use with unlearning, 2024. URL https://proceedings.mlr.press/v235/li24bc.html. Proceedings of the 41st International Conference on Machine Learning (ICML), PMLR 235:28525-28550.

Pietro Morerio, Jacopo Cavazza, Riccardo Volpi, Rene Vidal, and Vittorio Murino. Curriculum dropout. In *2017 IEEE International Conference on Computer Vision*, pp. 3564–3572, 2017. doi: 10.1109/ICCV.2017.383.

Oren Rippel, Michael Gelbart, and Ryan Adams. Learning ordered representations with nested dropout, 2014. URL https://proceedings.mlr.press/v32/rippel14.html. ICML 2014, PMLR 32(2):1746-1754.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Greg Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, 2019.

Weijia Shi, Akshita Bhagia, Kevin Farhat, Niklas Muennighoff, Pete Walsh, Jacob Morrison, Dustin Schwenk, Shayne Longpre, Jake Poznanski, Allyson Ettinger, Daogao Liu, Margaret Li, Dirk Groeneveld, Mike Lewis, Wen tau Yih, Luca Soldaini, Kyle Lo, Noah A. Smith, Luke Zettlemoyer, Pang Wei Koh, Hannaneh Hajishirzi, Ali Farhadi, and Sewon Min. FlexOlmo: Open language models for flexible data use, 2025. arXiv:2507.07024.

Igor Shilov, Alex Cloud, Aryo Pradipta Gema, Jacob Goldman-Wetzler, Nina Panickssery, Henry Sleight, Erik Jones, and Cem Anil. Beyond data filtering: Knowledge localization for capability removal in llms, 2025. URL https://arxiv.org/abs/2512.05648. arXiv preprint arXiv:2512.05648.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Jha, Sachin Kumar, Li Lucy, Xinxi Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew Peters, Abhilasha Ravichander, Kyle Richardson, Zejiang Shen, Emma Strubell, Nishant Subramani, Oyvind Tafjord, Evan Walsh, Luke Zettlemoyer, Noah Smith, Hannaneh Hajishirzi, Iz Beltagy, Dirk Groeneveld, Jesse Dodge, and Kyle Lo. Dolma: an open corpus of three trillion tokens for language model pretraining research. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15725–15788, 2024. doi: 10.18653/v1/2024.acl-long.840.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

Sainbayar Sukhbaatar, Olga Golovneva, Vasu Sharma, Hu Xu, Xi Victoria Lin, Baptiste Rozière, Jacob Kahn, Daniel Li, Wen tau Yih, Jason Weston, and Xian Li. Branch-Train-MiX: Mixing expert LLMs into a mixture-of-experts LLM, 2024. arXiv:2403.07816.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2306.11695.

Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient object localization using convolutional networks. In *2015 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 648–656, 2015. doi: 10.1109/CVPR.2015.7298664.

Zirui Wang, Zachary C. Lipton, and Yulia Tsvetkov. On negative interference in multilingual models: Findings and a meta-learning treatment. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 4438–4450, 2020. doi: 10.18653/v1/2020.emnlp-main.359.

Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://arxiv.org/abs/1812.08928.

TERMS USED IN THE APPENDICES

Several appendices keep the terms of an earlier version of this paper. A *block* or *private block* is a domain region; a *product* or *export* is a member; a *specialist* or *singleton* is a single-domain member; a *parent* is a trained model; p_{sub} is the exposure p; the *factorized* recipe is training without exposure; and *composition regret* is the composition penalty. Wherever $p_{\text{sub}} = 0.35$ appears for a model other than the 30M constant-exposure models, it denotes the end point of the ramp.

OVERVIEW OF EXPERIMENTS

Table 2 maps each reported experiment to its recipe, model size, number of trained models and evaluation data. "Main" is the mixture of WikiText-103, OpenWebText, Penn Treebank and Python code; "second" is the mixture of WikiText-103, a larger OpenWebText, CodeParrot and generated arithmetic.

Table 2: Experiments, recipes and data. Exposure: none ($p = 0$), constant ($p = 0.35$) or ramp ($0 \to 0.35$). Models: trained models per recipe. Dev: validation halves of the main mixture, used during development.

experiment	section	size	exposure	models	data	appendix
combined members	5	30M	none, constant, ramp	1, 3, 5	main, dev	N
external panel	5	30M	none, ramp	3, 3	512 external documents	Q
exact answers	5	30M, 124M	none, ramp	1, 2 per size	second, 192 equations	V
1.3B pair	5	1.3B	none, ramp	1, 1	held-out test split	U
domain-posterior weighting	5	30M, 124M	none (ramp as reference)	3; 1 per size	external panel; 192 equations	X.1
joint activity vs. foreign updates	5	30M	ramp, none and two arms separating them	3 per arm	main, dev	X.2
constant region weights	5	30M	none (ramp as reference)	3	external panel	X.1
export catalogue	5.2	51M	ramp	5	main, dev half B; A100	G.1
narrower arithmetic exports	5.2	51M	ramp	2	192 equations	M.1
rescaling	5	30M, 124M	none	1 per size	dev half B, 192 equations	X
exposure level and schedule	5	30M	four values of p; three schedules	1 per p; 2 to 5	main, dev	N
Python context	5	30M, 124M	none, ramp	1, 2 per size	second, 192 equations	Y
order within a region	5.2	30M	constant	1	main, dev	H
flat drop rate	5.2	30M	constant	1 per profile	main, dev	R
two widths per member	5.2	30M	constant	5	main, dev halves A and B	C
physical export	5.2	51M	ramp	5	main, dev	F, G
prefix coupling	5.2	30M	ramp	3 per sampler	main, dev	K
prefix coupling, full budget	5.2	30M	constant	2 per recipe	main, dev	P
removal and relearning	5	51M	ramp	2	main plus arithmetic	M, S
loss of removed domain	5	30M, 1.3B	ramp	5; 1	main, dev; 1.3B test	N, U
adding a domain with replay	–	30M	none, ramp	1 parent	web, science, Q&A, code	W
convolutional network	–	CIFAR-10 CNN, MLP	block partition	see appendix	CIFAR-10	R
program synthesis	–	1.3B	none, ramp	1, 1	MBPP sanitized test	U.1

A REGIONAL TRAINING AND EXPORT RECIPE

This appendix specifies the two sampling implementations and the full-budget comparison's training recipe. Both place the regional mask after GELU and before the FFN output projection. Attention, residual-stream width, embeddings, LayerNorm and output head remain unpartitioned. The region order is fixed before training; training learns weights that use this imposed order.

Partition and retention profiles. For FFN width U and shared fraction s_f, allocate $n_S = \mathrm{round}(s_f U)$ units to the shared pool. Split the remaining units into contiguous domain blocks, assigning the integer remainder to the last block; private blocks precede the shared pool in storage. The reference geometry has $U = 1536$, $s_f = 0.25$, $n_S = 384$ and four 288-unit private blocks. For a region of $n > 1$ units with dropout endpoints (a, b), the independent implementation uses

$$p_r(i) = 1 - a - (b - a)\frac{i - 1}{n - 1}, \qquad i = 1, \ldots, n. \tag{4}$$

A one-unit region uses $1 - a$. Private endpoints are $(0.10, 0.50)$ and shared endpoints are $(0.30, 0.50)$, giving mean conditional keep probabilities 0.70 and 0.60.

Two samplers, one export interface. On data from domain d, its block and the pool are visible; each foreign block is independently exposed with probability $p_{\text{sub}} = 0.35$. Let z_r indicate region visibility. The following table describes one domain microbatch, before gradient accumulation.

	ordered independent	coordinated regional prefixes
Foreign-block draw	Independently in each layer; shared across examples and tokens in that layer	Once for all layers, examples and tokens
Within visible region	Independent Bernoulli draws with probability $p_r(i)$ for every example, token and layer	One prefix length K_r, drawn independently for each visible region; the resulting index vector is shared across examples, tokens and layers
Activation multiplier	$z_r m_{r,i}/p_r(i)$	$z_r \mathbf{1}\{i \le K_r\}/q_r(i)$, where $q_r(i) = \Pr(K_r \ge i \mid z_r = 1)$
Foreign gradients	All exposed parameters can receive the current domain's gradients	Same update policy

Zero-probability entries receive multiplier zero. Conditional scaling divides by within-region retention, without an additional $1/p_{\text{sub}}$ factor for exposed foreign blocks. For the coordinated implementation, a fraction A_r determines $K_r = \max\{1, \text{round}(A_r n_r)\}$ in a nonempty visible region. The evaluated `direct_full50` law is

$$A_r \sim \tfrac{1}{2}\delta_1 + \tfrac{1}{2}\,\text{Unif}\{1, 0.75, 0.50, 0.25, 0.10\},$$
$$q_r(i) = \tfrac{1}{2} + \tfrac{1}{10} \sum_{a \in \{1,.75,.5,.25,.1\}} \mathbf{1}\{i \le \max(1, \text{round}(a n_r))\}. \tag{5}$$

Thus each visible region is full-width with probability 0.60; this is not a probability of full-network training. Its conditional mean retention is approximately 0.76, with exact counts set by rounding. The ordered-independent ramp is replaced by this prefix distribution in the coordinated arm. At evaluation, stochastic masks and inverted scaling are disabled; the selected regional indices are applied identically in every FFN layer, then physically sliced for export. The layer-local prefix implementation with marginal foreign scaling is specified separately in Appendix O.

Full-budget training configuration. The following settings are explicit configuration values or source defaults used by both 80M-token comparison arms; they are not generic command-line defaults for every experiment.

setting	value
Model	Six pre-LayerNorm transformer layers, six attention heads, model width 384, FFN width 1536, GELU, learned positions, tied token embedding/output weights; 50,257-token GPT-2 vocabulary; ordinary attention dropout zero
Training samples	Context 256, batch 24 per domain; random contiguous windows sampled with replacement from each training token cache
Domain balance	WikiText, OWT, PTB, code, in that order; one microbatch each per update; each microbatch's mean token cross-entropy divided by four before backward
Optimiser	AdamW on all parameters, weight decay 0.1, $\epsilon = 10^{-8}$, $\beta_2 = 0.95$; global gradient norm clipped to 1 after accumulation
Schedule	Two-phase cosine OneCycleLR: maximum LR 6×10^{-4}, first 5% of updates increasing; initial LR 2.4×10^{-5}, final target 2.4×10^{-9}
Momentum schedule	Default OneCycleLR momentum cycling: β_1 moves from 0.95 to 0.85 and back to 0.95, inversely to LR; scheduler advances after each optimiser update
Budget and seeds	3,255 updates, 79,994,880 base tokens, 19,998,720 per domain; independent training seeds 1 and 2; no auxiliary objective or extra training tokens
Runtime	PyTorch 2.7.1+cu128, Python 3.13.5, RTX 2080 Ti; float32 training without autocast

Table 3: Loss reversals under nested masks. Compositional violation rate over 287 nested mask pairs, the worst single violation in nats, and retrained-reference gap closure for target-block deletion and total ablation. Transformer rows are per seed; MLP CVRs average three seeds, with 50 pairs per seed.

substrate	training	**CVR**	worst violation	removal	total ablation
transformer, 6×384	naive	**0.70**	8.514 nats	173.2%	41.4%
	naive	**0.71**	9.126 nats	182.4%	42.0%
	naive	**0.71**	8.319 nats	165.5%	40.6%
	joint	**0.20**	0.133 nats	12.3%	31.4%
	joint	**0.18**	0.122 nats	11.7%	30.7%
transformer, 8×256	naive	**0.70**	8.551 nats	n/a	n/a
	joint	**0.22**	0.302 nats	n/a	n/a
CIFAR-10 MLP†	naive	**0.51**	0.410	92.3 / 115.2 / 83.5%	n/a
	joint	**0.13**	0.111	8.1 / 39.2 / 15.7%	n/a

The optimiser constructor specifies $(\beta_1, \beta_2) = (0.9, 0.95)$, after which OneCycleLR sets and cycles β_1. Its other schedule arguments retain PyTorch defaults: `div_factor=25`, `final_div_factor=10000`, `cycle_momentum=True`, `three_phase=False`. The token count is $4 \times 24 \times 256$ per update, with $\lfloor 80{,}000{,}000/(4 \times 24 \times 256) \rfloor$ updates. The coordinated arm therefore draws 13,020 whole-network masks. The separate arithmetic objective's extra passes and accounting are given in Appendix I.

Evaluation mapping. Training-script monitoring uses 24 sequences per domain from the full validation caches with draw seed 1234. The reported full-budget comparison instead uses 128 sequences per domain from token-stream half B, draw seed 20260916, inference batch 16, and the nine regional allocations from $\{25\%, 50\%, 100\%\}^2$ (Appendix P). Both arms use identical inputs and targets; the loss is mean next-token cross-entropy in nats. The pilot uses 48 sequences per domain on half B with seed 20260913. The historical 25-cell surfaces and the separately selected pruning controls have their own evaluation records (Appendices D and H.1).

Executable configuration. The implementation is in `scripts/transformer_selectivity.py` and `src/frankenstein/graded_partition.py`; evaluation is in `scripts/revision_evaluate.py`. The two full-budget configurations are `configs/revision_confirmation_seed1.json` and `configs/revision_confirmation_seed2.json`. Coordinated sampling is selected by `delivery-mode=global_conditional`, the five `delivery-levels`, and `delivery-full-prob=0.5`; both arms use `foreign-updates=all`. Paths in this paragraph are relative to the source package root.

B THE AGGREGATE VIOLATION RATE

An aggregate measure, for robustness. For nested masks $A \subseteq B$, a vendor may require $L(B) \leq L(A) + \delta$ as an operational policy. This is not a theorem about fixed neural weights: adding units changes the function, and can increase expected loss even when both masks were sampled during training. A violation neither proves lack of training exposure nor certifies presence or absence of target knowledge. Over every nested pair of the grid, the **compositional violation rate** is

$$\mathrm{CVR} = \frac{\#\{(A,B) : A \subset B,\ L(B) > L(A)\}}{\#\{(A,B) : A \subset B\}}. \tag{6}$$

This definition uses zero tolerance ($\delta = 0$). It needs no retrained reference or removal percentage and is bounded in $[0, 1]$. Two hundred and eighty-seven nested pairs arise from the grid cells and the single-selection arms.

Table 4: Target loss in nats over (pool kept, block kept), with delivered FFN units. Joint-trained transformer, target `code`; the full delivery $S \cup B$ is 2.316 nats at 672 units. **Bold** marks the Pareto-optimal cells, those nothing else beats at equal or smaller size.

pool \ block	100%	75%	50%	25%	10%
100%	**2.316**/672	**2.330**/600	2.365/528	2.471/456	2.633/413
75%	**2.344**/576	**2.359**/504	**2.404**/432	2.537/360	2.724/317
50%	2.432/480	**2.444**/408	**2.502**/336	**2.676**/264	**2.895**/221
25%	2.676/384	2.679/312	**2.750**/240	**2.976**/168	**3.227**/125
10%	2.958/326	2.949/254	3.025/182	**3.273**/110	**3.534**/67

Naive composition produces large discrepancies between nested-mask losses. It violates the condition on roughly seven nested pairs in ten. In the first listed seed, the largest loss reversal is **8.5 nats**, indicating severe performance sensitivity to the chosen masks. Joint training cuts the rate to roughly one in five and brings the worst violation to **0.13 nats**, a factor of 65. *Severity separates the two conditions far more sharply than incidence does in this seed*; joint training reduces both measures while retaining a nonzero violation rate.

Dependence and uncertainty. It is a descriptive rate over a fixed grid, and its pairs are strongly dependent: the same mask appears in many comparisons, so the 287 pairs are not 287 independent samples and we attach no interval to them. Uncertainty here comes from independent training seeds: transformer rows are reported per seed, while the MLP rows average three seed-specific rates, listed below. A bootstrap over evaluation sequences within a seed would give a second, narrower source and is not run.

Two replications. A second transformer geometry, 8 layers by 256 against the main 6 by 384, gives 0.70 naive against 0.22 joint with a worst violation of 8.551 against 0.302 nats, so the effect is not a property of one depth and width. The four naive transformer runs land at $0.70, 0.71, 0.71, 0.70$, which is a tighter agreement than we expected across two architectures. †The MLP row is computed over its 50 single-selection pairs rather than the full grid, since its grid was built with a superseded capacity cut (Section 5.2); it gives 0.51 naive against 0.13 joint, per-seed $0.44/0.54/0.54$ and $0.08/0.22/0.08$. The criterion transfers across substrates; its calibration does not.

Accuracy and reference effects. Evaluated on accuracy over the same grid, naive composition looks *better*: its degradation from the full model to a quarter width is smaller. That is a floor effect. The naive network sits near 0.56 across the whole grid where joint training reaches 0.858 at full width and 0.644 at a quarter, better in 10 of 12 cells, so it has further to fall. A model reading an off-distribution mask still returns a number; it takes per-mask losses, retained-domain utility and explicit references to interpret the change.

The accuracy comparison carries the confound stated in Section 3: with no subset sampling a domain never sees another domain's block, so the naive network is also a smaller effective one, and part of the accuracy gap is capacity rather than validity.

C THE CAPACITY SURFACE, EXACT VALUES

Per-checkpoint replication. Repeating the surface for all four domains on five checkpoints, evaluation only, the interior cells are always the large majority of the Pareto front. The similar-size spread is the largest loss range among cells in any 60-unit-wide band, with lower bounds 60, 90, . . . , 690 units per layer. These comparisons allow unequal widths:

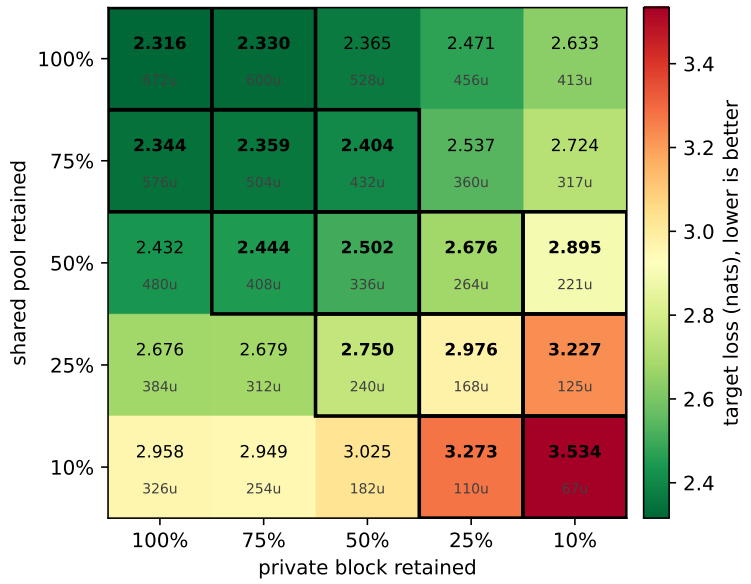


Figure 2: Code loss of members that keep different fractions of the shared pool and of the code region, in the 30M model trained with constant exposure; FFN units per layer beneath each cell. Outlined cells are Pareto-optimal: no member with the same or fewer units has lower loss. Diagonal cells keep equal fractions of both regions. The figure shows one domain and one model; exact values follow below.

checkpoint (four domains each)	Pareto points	interior	similar-size spread
joint, 6×384	14–16	11–13	0.429–0.479 nats
joint, 6×384, seed 1	14–16	11–12	0.427–0.542
joint, 8×256	14–17	11–13	0.556–0.759
naive, 6×384	14–15	10	0.471–0.705
naive, 8×256	14–17	10–13	0.628–0.838

Allocation among nearby widths.

widths compared	lower-loss allocation	higher-loss allocation	spread
336 / 326	2.502 (pool 50%, block 50%)	2.958 (pool 10%, block 100%)	$+\mathbf{0.456}$
432 / 413	2.404 (pool 75%, block 50%)	2.633 (pool 100%, block 10%)	$+0.230$
504 / 480	2.359 (pool 75%, block 75%)	2.432 (pool 50%, block 100%)	$+0.073$

The paired entries use nearby, unequal widths, so their loss differences describe local quality–size choices. The exact surface also contains strict dominance examples: 336 units at 2.502 nats improve on 384 at 2.676, and 432 at 2.404 improve on 456 at 2.471.

The proportional reference. A fixed retained fraction applied to both regions corresponds to the equal-fraction diagonal of each grid. On 48 of the 52 recorded complete singleton grids (checkpoints $\times$ domains, covering the joint, naive and both-geometry checkpoints and both training seeds), every equal-fraction cell is Pareto-optimal: no cell of equal or smaller delivered width has lower loss, so free regional allocation gains nothing over the proportional rule at the proportional budgets. The only exceptions are the four grids of the rejected $p_{\text{sub}} = 1$ checkpoint, where a skewed cell beats the 50%/50% diagonal by at most 0.211 nats. Between proportional budgets the independent controls add products; the best of them improves on the largest proportional cell that fits by up to 0.364 nats on the $4\times$ joint grids and up to 1.898 nats on the $16\times$ grids. No equal-width proportional reference exists at those intermediate budgets, so these are granularity gains, not matched-width comparisons.

Selection and evaluation here share the validation stream that produced each surface, so all of these comparisons are in-sample.

Selection across validation segments. Selecting the envelope on one contiguous half of the validation stream and evaluating it on the other token-position half, over four domains and eight budget bands on the joint 6×384 checkpoint:

domain	mean selection regret	worst	Pareto front A/B/shared	Spearman ρ
code	0.0000	0.0000	15/14/14	0.999
wikitext	0.0000	0.0000	16/16/16	1.000
owt	0.0000	0.0000	16/15/15	0.998
ptb	0.0038	0.0240	15/14/13	0.993

Selection regret is what the procedure costs: the loss on half B of the cell chosen on half A, minus the loss of the cell half B would have chosen at the same budget. Three of four domains cost exactly zero and the weakest rank agreement is $\rho = 0.993$. The Pareto fronts differ slightly more than the choices do, which is expected: neighbouring cells on a flat part of the surface swap order under resampling without changing the budget-constrained choice. Selection regret is not an optimism bound on the chosen loss, and the token-position split does not establish document-disjoint generalisation.

D CANDIDATE-SPECIFIC REANALYSIS AND PROVENANCE

The corrected envelope is computed from each recorded candidate's exact retained indices, its retained domain set, and its per-domain losses. Singleton candidates are evaluated on their target and have zero composition regret by definition. All-domain candidates are evaluated on all retained domains and compared with singletons at matching regional capacities. An all-domain score is never attached to a singleton's size. No unlearning reference or total-ablation quantity enters these support decisions.

checkpoint	code singleton	smallest units	all-domain	all-domain, $\delta = 0.02$
naive, 6×384	5/25	480	0/25	0/25
joint, 6×384	6/25	432	8/25	7/25
joint, seed 1	6/25	432	8/25	7/25
joint, 8×256	6/25	288	8/25	8/25
joint, $16\times$ FFN	6/25	1728	5/25	5/25

Quality tolerance is 0.10 nats relative to the full-width product with the same retained domain set. All-domain composition tolerance is 0.10 nats against matching singleton products. The third and fourth numeric columns differ only in whether capacity reversals are reported diagnostically or also restricted to 0.02 nats. The singleton counts above are unchanged by that policy. Units are FFN units retained per layer, not total parameters or hardware cost.

The accompanying reanalysis manifest, under `results/revision-20260913/legacy-envelopes/summary.json`, covers 13 historical groups and 52 target grids. Per-product records retain source-file hashes and the index and shared-score checks used for matching. Historical cross-file matches lack complete evaluation provenance; the counts are descriptive results conditional on these recorded grids and sample matches, not statistically confirmed acceptance probabilities. The reanalysis introduces no newly trained checkpoint and does not update the historical loss or throughput measurements.

Independent-data boundary. A data-only audit reconstructs the current WikiText validation and test token streams from preserved source rows. It also finds the complete 3,946-token test article *The Hustler (film)* in the current training tensor. We therefore do not treat the unfiltered official test cache as training-excluded. This audit computes no model losses and does not establish historical test non-use, cross-domain exclusion or absence of partial and near duplicates. The historical cache-based product-quality results remain development evidence; the source-group audit and its limitations accompany the release. Appendix Q separately evaluates a frozen external-distribution

panel, without claiming an untouched in-distribution test. A newly recovered, pinned OpenWebText preparation reconstructs the current primary caches byte for byte, but its validation-first token split cuts through a document. Reconstructibility does not establish source-group separation or historical non-use of those data.

Revision record. The reanalysis separates singleton utility from multidomain composition, replaces a purported all-unit-ablation ceiling with a diagnostic reference, and identifies the layer-coordination and scaling confounds in the historical prefix experiment. Repeated active-mask timing replaces earlier near-zero overhead estimates. An earlier deletion row combined two checkpoints; its normalised figure changed from 49.3% to 32.8% when evaluated with one provenance. Earlier exploratory claims about a conservation law, additive regional contributions and architecture predicting removability are not premises of the present analysis. The main text reports the final definitions and supported comparisons.

D.1 Interpreting ablation losses

Ablation losses measure changes in predictive performance. Interpreting them as suppression requires a named reference, retained-domain utility, and a specified evaluation or attack protocol.

Total ablation is a diagnostic, not a ceiling. For a fixed trained network, set inclusion alone does not order expected losses. Added units can interfere: for a binary example, a correct-class logit of 2 has cross-entropy 0.127 nats, while adding a unit contributing -3 changes it to -1 and loss to 1.313 nats. Both masks can be present during training. Consequently, ablating all FFN units is a useful reference but cannot upper-bound the loss or reference-gap closure of another deletion mask.

Our reported deletion/reference pairs are 11.3%/43.7%, 12.3%/31.4%, 95.4%/114.2%, and 133.2%/31.5%, where the second number is total ablation. The last discrepancy calls for investigation, not a declaration of impossibility. A loss above the retrained reference can reflect miscalibration or damage; the reference is not a maximum attainable loss and exceeding it does not quantify additional forgetting.

Exposure and ordering are insufficient certificates. Paired bootstraps over 200 sequences put superset-minus-subset loss at $+0.133$ $[+0.116, +0.152]$, $+0.241$ $[+0.223, +0.260]$, and $+5.543$ $[+5.208, +5.879]$ nats on three historical checkpoints. These establish loss reversals on the sampled data. They do not prove that masks were unseen, that a model is invalid, or that target knowledge is absent. Suppression assessment also needs retained-domain utility, a named retrained reference, task-level evaluations, and specified adaptation attacks.

Reference choice changes normalised scores. The same untuned RMU run closes **92.8%** against a retrained reference and **183.6%** against unigram. The reference therefore belongs to the definition of the score. This change is not a uniform multiplier across checkpoints: it is $1.978\times$ for that RMU run, but $2.381\times$ for partition plus RMU (110.9% versus 264.1%). For original loss L_0, retrained loss L_R and unigram loss L_U, the multiplier is $(L_R - L_0)/(L_U - L_0)$. Here $L_R = 7.734808$ and $L_U = 4.979599$ nats, while L_0 is 2.161505 for RMU and 2.984169 for partition plus RMU. Table 26 reports the reference and checkpoint provenance, with a generation check that rejects cross-checkpoint rows.

E Controlling prediction through the shared route

A historical MNIST experiment sweeps the shared region's dropout profile across six settings. The MLP has 200 units per hidden layer, five class groups, shared fraction 0.45, private dropout endpoints $(0.10, 0.50)$ and foreign-block exposure 0.35. Each setting trains for 12 epochs with seeds 0, 1 and 2. Regional widths are fixed at 90 shared units and 22 units per private block in each hidden layer. The shared profile's mean keep probability ranges from 0.90 to 0.15.

For each checkpoint, evaluation retains either the shared regions alone or the target's private blocks alone, while leaving the rest of the architecture as specified by that model. Let $A_r(X)$ be target-group accuracy for retained set X in seed r, using restricted argmax over digits 0 and 1 on 2,115

Table 5: Functional control at fixed widths: MNIST digit-0/1 prediction on 2,115 examples. Scores are normalised per seed: 100 is the extracted model and 0 is the model trained without the target group. Entries are mean ± sample standard deviation over three training seeds. The full six-setting sweep follows.

Mean shared keep	Shared-only score	Private-only score
0.60	99.93 ± 0.20	99.98 ± 0.11
0.15	30.87 ± 9.49	100.00 ± 0.00

MNIST test examples. This is not ten-class accuracy. Let A_r^{ref} be the accuracy of its reference trained without the target group. The reported normalised target score is

$$C_r(X) = 100 \frac{A_r(X) - A_r^{\text{ref}}}{A_r(S \cup B_d) - A_r^{\text{ref}}}. \tag{7}$$

Normalisation is performed per seed before taking the mean. It measures predictive performance relative to these two references; it is not a percentage of stored information.

Table 6: Shared availability during training changes accessible target performance at fixed regional widths. Scores are means ± sample standard deviations across three training seeds; these are not confidence intervals. Dropout endpoints increase linearly within the shared region.

mean shared keep	shared dropout endpoints	shared-only score	private-only score
0.90	0.05–0.15	100.00 ± 0.00	99.25 ± 1.06
0.75	0.20–0.30	99.95 ± 0.09	99.91 ± 0.08
0.60	0.35–0.45	99.93 ± 0.20	99.98 ± 0.11
0.45	0.50–0.60	99.90 ± 0.11	99.93 ± 0.01
0.30	0.65–0.75	95.12 ± 3.92	100.00 ± 0.07
0.15	0.80–0.90	30.87 ± 9.49	100.00 ± 0.00

Reducing mean shared keep from 0.60 to 0.15 lowers shared-only target score from 99.93 to 30.87, while private-only score remains approximately 100. Shared-only raw target accuracy falls from 99.87% to 54.93%. Thus training availability controls what is predictively accessible through the shared route, beyond choosing an export width. This demonstrates functional control on this MNIST setting, not a measured information allocation law for arbitrary domains.

The routes are redundant at the lighter dropout settings: both can separately predict the target. Their scores are therefore not complementary shares, and a decline in shared-route performance cannot be interpreted as an equal quantity of information moving into private units. In particular, $100 - C_r(S)$ is not a measurement of $C_r(B_d)$. The private label denotes the assigned domain region; foreign-block exposure and the remaining architecture preclude interpreting it as exclusive knowledge. The operational controls are regional capacity, training availability and the predictive performance of the resulting retained products.

The six source files are in `results/ladder_small/`, named `mnist-mlp-g5-h200-t0-k`K`.json` for $K \in \{90, 75, 60, 45, 30, 15\}$. Each contains the partition settings, all three seeds, retained-set accuracies and retrained references. No new training or evaluation was performed for this table.

F Hardware cost of Pareto-optimal exports

Parameters, half-precision bytes and transformer-body matrix-multiplication FLOPs per token for every Pareto-optimal cell of Figure 2, on the joint 6×384 checkpoint with target code. The full model is 30.04M parameters and 23.6M body FLOPs per token; the tied embedding is 19.4M of the parameters, and attention, norms and the head are untouched by an FFN partition. The narrowest Pareto export retains 77% of the parameter count. The analytic FLOP estimate includes attention and

FFN matrix multiplications but excludes the vocabulary projection, normalisation and elementwise operations. Weight tying saves storage, not output-projection computation: the projection alone adds $2|V|d$ FLOPs per token. Reported FLOP fractions therefore concern the transformer body, whereas the timings below include the full model forward.

Table 7: Hardware cost of the fourteen Pareto deliveries. FFN width and parameter count are different quantities: parameters and body FLOPs include costs outside the sliced FFN.

pool/block	units	loss	params	of full	fp16 MiB	body FLOPs/token (of full body)
10/10	67	3.534	23.27M	77.4%	44.4	10.1M (42.6%)
10/25	110	3.273	23.46M	78.1%	44.8	10.5M (44.3%)
25/10	125	3.227	23.53M	78.3%	44.9	10.6M (44.9%)
25/25	168	2.976	23.73M	79.0%	45.3	11.0M (46.6%)
50/10	221	2.895	23.98M	79.8%	45.7	11.5M (48.6%)
25/50	240	2.750	24.06M	80.1%	45.9	11.6M (49.4%)
50/25	264	2.676	24.18M	80.5%	46.1	11.9M (50.3%)
50/50	336	2.502	24.51M	81.6%	46.7	12.5M (53.1%)
50/75	408	2.444	24.84M	82.7%	47.4	13.2M (55.9%)
75/50	432	2.404	24.95M	83.0%	47.6	13.4M (56.9%)
75/75	504	2.359	25.28M	84.2%	48.2	14.1M (59.7%)
75/100	576	2.344	25.62M	85.3%	48.9	14.7M (62.5%)
100/75	600	2.330	25.73M	85.6%	49.1	15.0M (63.4%)
100/100	672	2.316	26.06M	86.7%	49.7	15.6M (66.2%)

Against the widest Pareto cell, the narrowest trades 1.218 nats for 2.79M parameters and 5.6M body FLOPs per token. In this geometry, the unsliced embedding, attention and output layers limit the parameter reduction available through FFN slicing. Appendix G reports the separate FFN-dominated geometry. A retained-width ratio alone does not specify a parameter, computation or measured-throughput saving.

Forward timing of physical exports. Batch 16×256 tokens, each slice verified against its masked equivalent before timing. These are full-sequence forward measurements, with no autoregressive decoding loop or key/value-cache benchmark.

FFN units kept	of FFN width	parameters	body FLOPs/tok	measured	speedup
1536 (full)	100%	30.04M	100%	26.85 ms	1.00×
672 (a domain)	43.8%	26.06M	66.2%	23.42 ms	**1.15**×
125 (deep prefix)	8.1%	23.53M	44.9%	21.15 ms	**1.27**×

Repeated active-mask and export benchmarks. A repeated control reuses the historical joint checkpoint, reconstructs its partition settings, and verifies that every partition layer changes its activation in training mode and is the exact identity when disabled. On an RTX 2080 Ti, batch 16 and context 256, three paired repetitions alternate the order of partitioned and unpartitioned runs. Each arm starts from identical weights and AdamW state, uses five warmup updates and measures 20 synthetic cross-entropy forward/backward/optimiser updates. Mean times are 99.57 ms unpartitioned and 105.46 ms partitioned, a 5.91% increase; the paired timing interval is $[5.64, 6.19]\%$. This measures one domain microbatch per update. It excludes the full balanced accumulation loop, adversarial auxiliary passes and data loading, and the interval is not training-seed uncertainty.

Inference timings use the physically sliced model with the source model removed from GPU memory, three repetitions of 20 full-context forwards after five warmups, and include vocabulary logits. The maximum slice-versus-mask difference is 1.34×10^{-5}. Median speedups are:

product	units/layer	params	batch 1	batch 16
full model	1536	30.04M	1.000×	1.000×
code specialist	672	26.06M	0.969×	1.139×
code pool/block 75%	504	25.28M	0.985×	1.181×

The full model takes 3.171 ms at batch 1 and 26.419 ms at batch 16. Slicing improves batched forward throughput here but provides no observed batch-1 speedup. This is not a key/value-cached decoding benchmark. GPU memory denotes peak PyTorch allocated memory for the slice and input, excluding CUDA context. Two control jobs ran on separate GPUs of the same host, so host-level contention was not independently eliminated. The source artifact is `results/revision-20260913/controls/benchmark_standard/tf_g_sf25_t35.json`, which records checkpoint/script hashes, raw repetitions and mask probes.

F.1 Training tokens by number of members

Table 8 compares the training tokens needed to obtain a number of members of the 30M main mixture by three routes. The entries are computed from the budgets reported in the main text, not measured in new runs. A member is a set of domains at a given width. Separate training trains each member from scratch on the 20M tokens of each of its domains; dense training followed by pruning trains one unpartitioned model on all 80M tokens and then prunes and repairs each member with 1M tokens; region dropout trains one partitioned model on the same 80M tokens and exports each member without further tokens. Wall-clock time, the extra forward cost of calibration for pruning, and any training overhead of the masks are not included.

Table 8: Training tokens (millions) to obtain N members, computed from the reported budgets, and the measured quality reference of each route on code. $N = 1$: the code member; $N = 4$: the four single-domain members; $N = 15$: all combinations of domains at full width. The table does not show wall-clock time or members of reduced width.

route	$N = 1$	$N = 4$	$N = 15$	code quality (nats)
separate training	20	80	640	2.310 for a model trained on the 20M code tokens
dense, prune, repair	81	84	95	0.078 to 0.419 below exported members at four widths after 1M code tokens
region dropout	80	80	80	ramp code member 2.222 to 2.279; dense model of the same budget 2.153 to 2.165

Separate training grows with the number of domains summed over all members, so all fifteen combinations of four domains need 32 domain-trainings of 20M tokens. Pruning with repair adds a fixed budget for every member and gave the best narrow members in our comparison. Region dropout pays once, through the loss of its single-domain members relative to a dense model of the same budget, and adds no tokens for further members.

G The FFN-dominated geometry

The main model is embedding-dominated, so even its narrowest tested export keeps 77% of the parameters. To test a model in which FFN slicing removes a larger share of the weights, we increase FFN width from $4\times$ to $16\times$ the hidden width. The resulting model has 51.3M parameters, with 55% in the FFNs and 38% in the embedding. We train three conditions on the same 80M tokens: unpartitioned, factorized, and domain-subset exposure at constant $p_{\text{sub}} = 0.35$. The exposed and unpartitioned conditions have two seeds each; the factorized condition has one. This gives 1.6 training tokens per parameter, compared with 2.7 in the main model.

Parameter and FLOP reductions. A delivery keeping 4% of the FFN width retains $\mathbf{47.0}\%$ of the parameters (24.09M) and $\mathbf{17.7}\%$ of the body matrix-multiplication FLOPs per token, against 77.3% and 42.4% at $4\times$; a 25%-width delivery is 30.04M parameters, the size of the entire main model.

Composition and regional allocation.

full-width domain regret	code	wikitext	owt	ptb
factorized	$+\mathbf{8.184}$	$+1.408$	$+1.131$	$+1.975$
subset exposure	$+0.017$	$+0.017$	$+0.004$	$+0.004$

The factorized code regret is 8.184 nats, compared with 8.428 at $4\times$; both comparisons show substantially lower regret with subset exposure. The capacity surface for the subset model has 16 Pareto points of which **13 are interior**, and the loss difference between the 50%/50% and 10%/100% pool/block allocations is **1.101 nats** at 1,344 versus 1,306 units. The same allocations differ by 0.456 nats at 336 versus 326 units in the $4\times$ model. These are nearby-width comparisons, not equal budgets. Over two subset seeds against two unpartitioned controls the specialist costs $+0.137$ **to** $+0.171$ **nats**; the sets do not overlap, with within-condition spreads of 0.015 and 0.019 against a smallest gap of 0.137, and the range overlaps the $4\times$ range of 0.112–0.163.

A100 forward benchmark. The same slice-and-verify protocol as the $4\times$ bench (slice-vs-mask $\leq 1.3\text{e}{-5}$ before timing), on the exact index sets the surface scored, one A100, context 256, which is this model's trained context:

delivery	units	params	loss	tokens/s (batch 1 / 16)		speedup
full model	6144	51.31M	2.298	71,167	116,650	1.00×
specialist ($S \cup B_d$)	2688	35.36M	2.281	104,440	173,420	**1.49**×
pool/block 75%	2016	32.26M	2.323	105,258	186,211	1.60×
pool/block 50%	1344	29.16M	2.548	106,864	203,510	1.74×
pool/block 25%	672	26.06M	4.037	106,616	229,668	**1.97**×

Speedup is batch-16 full-sequence forward throughput against the full model; peak inference memory falls from 555 to 458 MiB at batch 1 for the full model versus the 25%/25% export. These historical peaks include the resident source model; the repeated control below removes it. Estimated fp16 weight storage falls from 98 to 50 MiB. Batch-1 latency stays near 2.4 ms below the specialist's size, while batch-16 throughput keeps improving. This is not a measurement of autoregressive decode latency or key/value-cache use. The $1.97\times$ delivery increases loss by 1.74 nats relative to the full model. The repeated benchmark below measures active partition masking explicitly.

Repeated RTX 2080 Ti control. The corrected protocol of Appendix F, on the same FFN-$16\times$ checkpoint and surface index sets, verifies slice-versus-mask differences at most 2.10×10^{-5} and removes the source model from the GPU before timing each slice. Three repeated full-context forward timings give:

product	params	resident MiB	batch-1 speedup	batch-16 speedup
full model	51.31M	206.34	1.000×	1.000×
code specialist	35.36M	143.87	1.272×	1.416×
code pool/block 75%	32.26M	132.45	1.149×	1.530×

The full-model medians are 3.993 ms at batch 1 and 45.235 ms at batch 16. Resident memory is PyTorch allocated memory at batch 16 and excludes the CUDA context; it is distinct from peak memory during the forward. Batch-1 repetitions are variable: the 75%/75% slice takes 3.079–3.925 ms, so these medians do not establish precise latency gains across machines.

With masking active, mean training microbatch times are 168.08 ms unpartitioned and 184.27 ms partitioned, a 9.63% increase with a paired timing interval of $[7.40, 11.42]\%$. As in the standard-FFN control, these three pairs measure synthetic cross-entropy forward/backward/AdamW updates, exclude adversarial passes and the full domain accumulation loop, and do not measure retraining uncertainty. Both control jobs used separate GPUs of one host. The complete source, including raw repetitions, mask probes and checkpoint/script hashes, is `results/revision-20260913/controls/benchmark_ffn16/tf_f16_sub35.json`.

Capacity reversals across seeds. Capacity-reversal magnitudes vary across seeds: worst violations per domain run 0.033/0.047/0.035/ **0.134** on seed 0 and 0.005/0.000/0.033/**0.082** on seed 1, so two domains drop to near zero while **ptb exceeds the $4\times$ observed maximum of 0.040 in both seeds**. Grid-worst regret widens to 0.083–0.173. Factorized composition regret remains much larger than these capacity reversals. The small capacity reversals observed on the $4\times$ grids do not generalise to every domain in this geometry.

G.1 Export catalogue

The five ramp models with the widened FFN of Section 5.2 were exported into fixed members, with the first units of the pool and of each included region at the same fraction: code alone at 100, 75, 50 and 25%, code and WikiText at 100 and 75%, and all four regions at 100 and 75%. Losses are on the second validation half (200 windows per domain), which was also used during development. Costs were measured on one A100 80GB GPU with the source model moved off the GPU, context 256, FP32 without TF32: forward throughput at batch 1 and 16 (median of ten timed passes after three warm-up passes), peak allocated memory at batch 16 excluding the CUDA context, and greedy generation of 128 tokens at batch 1 without a key-value cache.

The protocol and code were committed before the first forward pass (25 September 2026, 18:28 UTC). Its export check required the FP32 logits of every export to match the masked model within 2×10^{-5}; this check failed on the A100 at 2.0 to 4.2×10^{-5} before any measurement. An amendment committed at 18:43 UTC, after this failure and before any result, replaced it by two conditions: exact identity of the selected weights and agreement of FP64 logits within 10^{-9}. All 272 checks passed these conditions (largest FP64 difference 5.9×10^{-14}); 24 of them exceeded the original FP32 bound, with a largest difference of 4.2×10^{-5}. All measurements were then run in FP32.

Table 9: Export catalogue over five ramp models with the widened FFN. Parameters, peak memory at batch 16 (MiB), speed relative to the full network at batch 1 and 16 and in uncached generation, largest composition penalty over the included domains, and largest loss above the full-width single-domain member, in nats; ranges over five models. The full network has 51.31M parameters. Losses are on development data.

member	width	params	memory	batch 1	batch 16	generation	penalty	above single
code	100%	35.36M	943–944	0.99–1.05	1.45	0.92–1.00	–	0.000
code	75%	32.26M	933	1.05	1.57–1.58	1.00	–	0.041–0.046
code	50%	29.16M	921	1.06–1.07	1.72	1.00	–	0.297–0.372
code	25%	26.06M	907	1.05–1.06	1.91–1.93	1.00–1.01	–	2.263–3.460
code + WikiText	100%	40.68M	962	1.03–1.04	1.26	1.00	0.010–0.017	0.010–0.017
code + WikiText	75%	36.25M	947–949	1.04–1.05	1.41–1.42	1.00	0.008–0.015	0.054–0.061
all four	100%	51.31M	1004–1005	1.00	1.00	1.00	0.038–0.048	0.038–0.048
all four	75%	44.22M	977	1.02–1.03	1.15	1.00	0.097–0.116	0.143–0.161

H Index selection against pruning with repair

H.1 Frozen selection and matched repair objectives

These controls compare the historical reference partitioned checkpoint with its unpartitioned baseline; they do not compare the new sampler checkpoints in Appendix P. Both reference checkpoints were trained on 80M tokens at the same 30M-parameter geometry. To separate selection from evaluation, we freeze the best regional allocation within each width budget using 100 code sequences from the first token-position half of validation, then score every method on 100 sequences from the second half. The chosen allocations are 100/100, 75/75, 50/50 and 25/25 for 672, 504, 336 and 168 units per layer. These halves are token-disjoint, not document-disjoint. The baseline retains the units with largest outgoing-weight column norms, independently in each layer. Delivered widths and parameter counts match the selected products.

Repair minimises cross-entropy either on code alone or by cycling through the four domains. It uses AdamW with learning rate 10^{-4}, betas $(0.9, 0.95)$, no weight decay, batch 24 and context 256. A separate random stream per domain ensures matching prefixes of each domain's training examples. Requested budgets of 200k, 1M and 2M execute 196,608, 995,328 and 1,996,800 tokens. At 1M,

the all-domain run sees 245,760 code tokens and the code-only run sees 995,328. Repair runs in the masked full-size architecture; the delivered parameter count is computed after physical slicing. Our selected products receive no repair.

Table 10: Pruned-and-repaired code loss minus the frozen selected product's loss, evaluated on validation half B. Positive favours slicing. Brackets are 95% paired bootstrap intervals over 100 evaluation sequences (2,000 resamples); they do not include checkpoint or repair-seed variation. Budgets are requested total repair tokens.

repair data	budget	672u	504u	336u	168u
		$+1.417$	$+2.189$	$+2.824$	$+3.335$
none	0	$[+1.325, +1.509]$ $+0.330$	$[+2.084, +2.298]$ $+0.653$	$[+2.694, +2.958]$ $+1.128$	$[+3.224, +3.446]$ $+1.293$
all domains	200k	$[+0.303, +0.360]$ $+0.054$	$[+0.604, +0.706]$ $+0.097$	$[+1.047, +1.210]$ $+0.127$	$[+1.184, +1.405]$ $+0.011$
all domains	1M	$[+0.041, +0.068]$ $+0.019$	$[+0.081, +0.112]$ $+0.030$	$[+0.106, +0.150]$ -0.007	$[-0.024, +0.048]$ -0.258
all domains	2M	$[+0.004, +0.034]$ $+0.102$	$[+0.015, +0.045]$ $+0.254$	$[-0.024, +0.012]$ $+0.513$	$[-0.282, -0.233]$ $+0.689$
code only	200k	$[+0.086, +0.119]$ -0.051	$[+0.225, +0.285]$ -0.036	$[+0.463, +0.566]$ -0.062	$[+0.614, +0.768]$ -0.307
code only	1M	$[-0.068, -0.035]$ -0.107	$[-0.053, -0.020]$ -0.104	$[-0.080, -0.043]$ -0.171	$[-0.334, -0.278]$ -0.490
code only	2M	$[-0.128, -0.088]$	$[-0.123, -0.085]$	$[-0.192, -0.151]$	$[-0.520, -0.461]$
selected slice loss		2.379	2.421	2.558	3.009

The repair objective changes the conclusion. After 200k requested code-only tokens, slicing still leads by 0.102–0.689 nats. At 1M, code-only repair leads at all four widths: 0.051, 0.036, 0.062 and 0.307 nats. All four paired intervals exclude zero. With all-domain repair at 1M, slicing's margins are 0.054, 0.097, 0.127 and 0.011 nats, and the narrowest interval crosses zero. The 2M all-domain baseline leads at 168 units by 0.258 nats; code-only repair at 2M leads by 0.107, 0.104, 0.171 and 0.490 nats across the four widths. These controls support slicing as a zero-repair option, not a claim that a specialist needs millions of repair tokens to match it. They use one pair of base checkpoints and one repair seed, so the crossover is specific to this recipe and sampled budgets.

Sources are the seven `results/revision-20260913/controls/pruning/code_*.json` files for the conditions above. Each records checkpoint/script hashes, exact selected and pruned indices, per-domain token exposure and paired sequence losses. The recorded selection surface hash is verified against `results/revision-20260913/controls/selection_half_a/dom_code.json`, whose evaluation metadata specifies half A and seed 20260913. Some schema-2 control artifacts omit these fields in their copied selection metadata; the hashed source supplies them. Every control uses half B and evaluation seed 20260913, with repair seed 20260914.

H.2 Activation-informed structured pruning

A stronger structured heuristic ranks each FFN unit by $\sqrt{\mathbb{E}[h_j^2]}\,\|W_{\text{out}}[:, j]\|_2$, where h_j is the post-GELU input to the output projection in the unpruned model. Activation RMS is calibrated on 32,768 code *training* tokens (128 contexts, seed 1729) without gradients; the ranking is fixed and reused at every width. This uses the weights-and-activations idea of Wanda (Sun et al., 2024), but ranks whole units independently per layer rather than individual weights per output. It is not a reproduction of Wanda. Calibration adds forward-pass cost beyond the repair-token budgets below.

The checkpoints, half-A selected products, half-B evaluation sequences and code-only repair streams are identical to the magnitude controls. With no repair, selected slices lead by 0.731–3.086 nats. At 200k requested repair tokens, the margins are $+0.005, +0.070, +0.129, +0.187$ nats at 672, 504, 336 and 168 units; the 672-unit paired interval is $[-0.012, +0.022]$, so this comparison does not resolve a quality difference at that width. At 1M tokens, the activation-informed baseline leads by 0.084, 0.078, 0.144 and 0.419 nats, and at 2M by 0.131, 0.138, 0.217 and 0.551 nats. All eight

corresponding paired intervals exclude zero. Figure 3 compares all three pipelines at the sampled budgets; connecting lines do not identify an exact crossover between them.

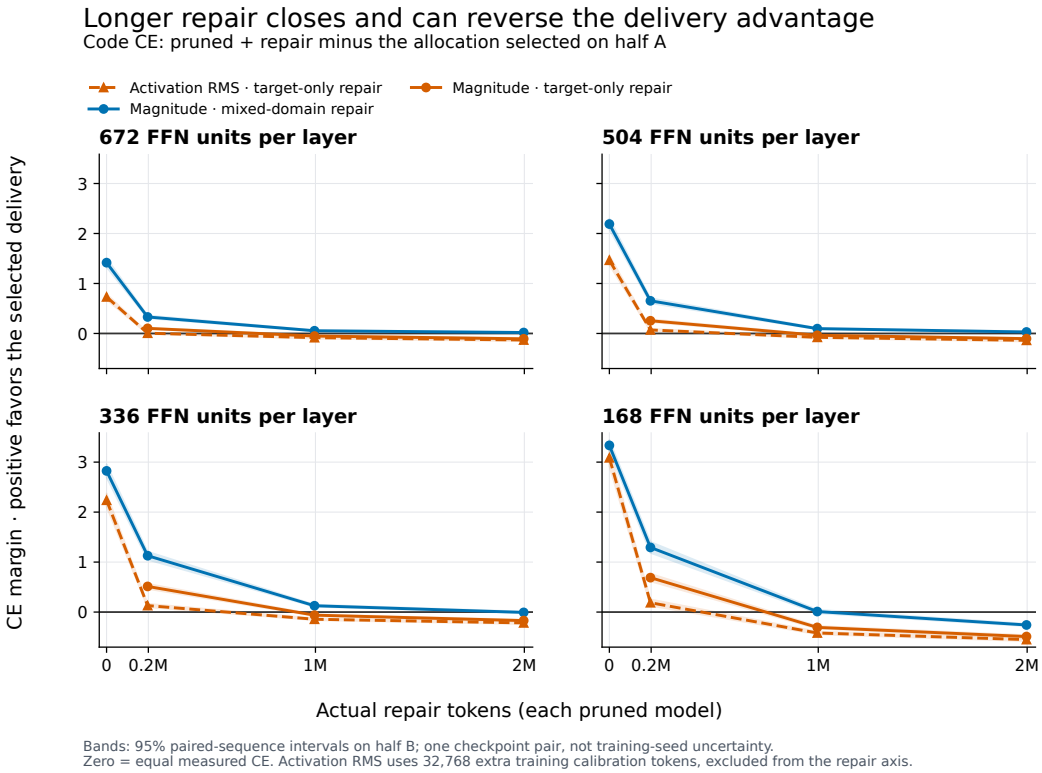


Figure 3: Repair objective and unit-ranking criterion change the comparison with zero-repair slices. Each panel holds delivered width fixed. The horizontal axis counts actual repair tokens; activation calibration adds 32,768 forward tokens. Bands are paired evaluation-sequence intervals, not training-seed uncertainty.

The four activation-control artifacts are `results/revision-20260913/supplement/pruning/code_activation_target_*.json`. They record calibration samples and scores, exact masks, repair exposures, per-sequence losses and hashes. The figure's source manifest is `results/revision-20260913/figures/manifest_review.json`.

H.3 Historical comparison on a shared evaluation sample

Both models see 80M base tokens at the same geometry. Our evaluated product is a code specialist selected from the partitioned model. The baseline is an ordinary joint model, pruned by outgoing-weight magnitude to the same FFN width and repaired on the balanced four-domain mixture. Repair tokens are total tokens across that mixture, while the table scores only code. The frozen-selection target-only controls above supersede these historical rows for the specialist comparison. The historical sweep is retained to show the effects of repair budget and allocation on the original evaluation sample.

The two blocks differ only in how our side spends its budget: block-first fills the budget from the 288-unit private block, then takes any remainder from the pool. At 336 units this is pool 12.5% / block 100%, where the selected allocation is 50% / 50%. The allocation penalty is 0.000 nats at 672 units, 0.042 at 504, 0.399 at 336 and 0.270 at 168, and it changes the apparent crossover in this historical comparison.

At 168 delivered units the baseline improves from 6.765 nats at no repair to 2.709 at 2M, while our selected delivery sits at 2.976 throughout. The crossover is bracketed rather than located: we sampled four budgets and did not fit a curve.

Table 11: Margin in nats, pruned+repair minus ours, on target `code`. Positive means index selection is better. Ours receives no repair at any row, so the columns are a fixed reference and only the baseline moves. These are point estimates without a comparison-specific confidence interval or independent training-seed replication; uncertainty from other experiments cannot supply that interval.

repair tokens	672u	504u	336u	168u
delivered by the allocation the envelope procedure selects				
0	$+1.619$	$+3.048$	$+3.874$	$+3.789$
200k	$+0.315$	$+0.655$	$+1.197$	$+1.306$
1M	$+0.048$	$+0.099$	$+0.118$	-0.015
2M	$+0.018$	$+0.027$	-0.019	$\mathbf{-0.267}$
delivered by the fixed block-first allocation rule				
0	$+1.619$	$+3.006$	$+3.475$	$+3.519$
200k	$+0.315$	$+0.613$	$+0.798$	$+1.036$
1M	$+0.048$	$+0.057$	$\mathbf{-0.281}$	$\mathbf{-0.284}$
2M	$+0.018$	-0.016	$\mathbf{-0.417}$	$\mathbf{-0.536}$

The selected allocation is read off the scored surface and re-scored here, and these are *not* independent samples: both use the same evaluation batch, and the re-scored losses reproduce the surface exactly. The disjoint-sample check of Section 5.2 reports 0.001 nats of mean selection regret. That measures stability of the chosen allocation; it does not bound optimism of the reported losses or bias in this baseline comparison.

The margin table gives no absolute levels, so the 200k row is repeated below with the losses each margin is a difference of, for both of our arms against the one baseline they share.

	ours, no repair			margin	
units	selected	block-first	pruned + 200k	selected	block-first
672	**2.316**	2.316	2.631	$+0.315$	$+0.315$
504	**2.359**	2.401	3.014	$\mathbf{+0.655}$	$+0.613$
336	**2.502**	2.900	3.699	$\mathbf{+1.197}$	$+0.798$
168	**2.976**	3.246	4.282	$\mathbf{+1.306}$	$+1.036$

The ordering, at a fixed allocation. Within one trained checkpoint, holding (k_S, k_B) at the diagonal cells and varying only which units each region contributes. Magnitude is the top k by outgoing-weight norm *within* each region, so it solves an easier problem than the baseline above, which also had to choose an allocation. Fixing the allocation isolates within-region unit selection. Random is the mean of five draws.

pool/block	units	prefix	magnitude	random	vs magnitude	vs random
100/100	672	2.316	2.316	2.316	$+0.000$	$+0.000$
75/75	504	2.359	2.360	2.438	$+0.001$	$+0.079$
50/50	336	2.502	2.535	2.795	$+0.033$	$+0.293$
25/25	168	2.976	3.033	3.380	$+0.057$	$+0.404$
10/10	67	3.534	3.589	3.775	$+0.055$	$+0.241$

The same pattern holds on wikitext ($+0.075$ to $+0.349$ against random, $+0.003$ to $+0.049$ against magnitude) and on the historical, confounded prefix-sampling checkpoint of Appendix O: $+0.898$ nats against random at 67 units, $+0.076$ against magnitude.

Table 12: Constant-exposure partitioned pipelines and an unpartitioned RMU baseline under the same local optimisation and nominal 2M-token relearning recipe. Higher code loss indicates stronger target suppression; lower retained loss indicates better utility. Retained loss is the unweighted mean over WikiText, OpenWebText and Penn Treebank. Individual-domain results and paired intervals are reported below.

	RMU	Delete + RMU	Delete + shrink + RMU
Parameters (M)	30.045	28.716	27.719
Code CE after RMU	7.334	8.254	7.992
Code CE after relearning	2.179	2.333	2.334
Retained CE after RMU	4.425	4.594	4.673
Retained CE after relearning	4.924	5.432	5.518

I Composing structural removal with optimisation

This appendix reports optional interventions on exported weights, their target and retained-domain losses, and the adaptation protocols. Reference-gap scores describe corpus loss; functional task evaluations and relearning tests supply distinct evidence.

Export followed by optimisation. For an optional target intervention $\mathcal{U}_R$, the final weights are

$$\theta_{C,\mathbf{k},R} = \mathcal{U}_R(\mathrm{Slice}[\theta, M(C,\mathbf{k})])\,. \tag{8}$$

The identity intervention gives the immediate exports of Section 5.2. For suppression, $\mathcal{U}_R$ is RMU applied after removing the target block. Slicing precedes optimisation, so the optimiser acts on the standalone model.

RMU on structurally edited models. Table 12 compares complete pretrained pipelines using constant exposure in the partitioned arms and the same RMU and relearning recipe. The deletion-plus-RMU pipeline uses 1.33M fewer parameters than the unpartitioned RMU baseline and has post-relearning code loss 2.333 versus 2.179 nats; the paired difference is $+0.154$ with a 95% interval $[+0.142, +0.167]$. The partitioned pipeline has higher retained-domain losses at delivery and after relearning, as shown alongside the target result. RMU's incremental normalised retained-loss increase is 0.565% versus 0.453% (definition in Table 26); the pretrained starting points also differ.

The same interface combines removal, shrinking and RMU. A 1,032-unit export retains 27.72M parameters, with post-relearning code loss 2.334 versus 2.333 for the 1,248-unit combination. Its extra retained-utility cost concentrates in Penn Treebank, as quantified below.

With ramped exposure, deletion plus RMU leaves mean post-relearning code loss 0.113 nats higher and retained-domain loss 0.443 nats higher than the same single RMU baseline. The five seeds vary the partitioned pretraining; baseline variability is not estimated (Appendix N).

Historical corpus-loss reference comparisons.

target and method	delivered loss	reference loss	collateral
arith, slice after adversarial training	**5.150**	5.263	-0.6%
whole domain, index-only	2.842	7.735	0.2%
whole domain, RMU alone (Li et al., 2024)	7.334	7.735	0.5%
whole domain, index + RMU	8.254	7.735	0.6%

One structural interface, optional optimisation. Selection and regional prefixes produce standalone models without post-hoc gradients. Suppression objectives can be imposed during training or applied after export: the narrow arithmetic experiment follows the first route, while the code RMU experiment follows the second. The positive combined-method result is that RMU applied after code-block deletion raises target loss more than RMU applied to the full plain model, including after the evaluated longer relearning budgets (Appendix S). This comparison evaluates the complete pipelines, whose pretraining structures differ.

Direct comparison and reference sensitivity. The matched RMU pair reaches code loss 8.254 versus 7.334 at delivery and 2.333 versus 2.179 after the nominal 2M-token relearning schedule. These direct contrasts are the primary evidence for the combined pipeline. Separately reported gaps to a model trained without code use an attack-floor control with different batching and evaluation draws; those gaps are descriptive and should not be interpreted as a fully matched causal estimate of leaked information.

Matched RMU recipe, utility and cost. The direct pair uses the same unlearning script, nominal 2M-token forget budget, layer 3, steering strength 20, retained-loss weight 1 and learning rate 5×10^{-5}. Each starts from a separately trained nominal-80M-token checkpoint. RMU's incremental normalised retained collateral is 0.453% on the plain model and 0.565% after partitioning and deletion. The latter checkpoint starts with worse retained utility: after RMU its WikiText, OpenWebText and Penn Treebank losses exceed plain RMU's by 0.224, 0.176 and 0.109 nats. Thus the pipelines are not matched in retained utility.

The source defaults imply 488 RMU updates at batch 16 and context 256, processing 1,998,848 forget and 1,998,848 retain tokens, with additional frozen-model retain forwards. The nominal 2M-token attack rounds to 2,002,944 tokens. These exact counts are inferred from the archived recipe and source defaults; the historical result records store nominal budgets rather than a complete runtime token ledger. Both direct comparison arms use the same recipe. The paired target-loss difference is $+0.920$ nats at delivery (95% interval $[+0.816, +1.015]$) and $+0.154$ after relearning ($[+0.142, +0.167]$), conditional on these trained models and evaluation samples.

Performance under total FFN ablation. Ablating *every* partitioned unit in the transformer, not the target's block but all of them, still retains **49.8–84.4%** on the normalised target-loss measure over 17 historical configurations. This establishes residual performance under total ablation, but does not bound the performance of other retained sets.

Predictive performance of retained regions. Evaluate the exact retained set to determine whether the resulting product still performs the target. A pool-only score is informative about that smaller model, but set inclusion does not transfer it into a monotonic bound on a larger retained set. On an MLP at mean shared keep probability 0.60, the shared pool *alone* reaches a normalised target score of 99.9 (Appendix E), and measured gap closure after target-block deletion is 0.1–0.5%. Stronger shared dropout reduces the pool-only score to 30.9 on the same normalisation and is associated with greater gap closure. A CNN whose partitioned layer *is* the only path still closes only 0.5–3.6% of the reference gap after target-block deletion.

Controlled residual-path intervention. We vary routing within one MLP whose partitioned layer is the only path from stem to head. The treatment adds an *identity skip* around that layer. No parameters are added, task, width, optimiser, seeds and epochs are identical, and the routing applies to the unpartitioned controls too. With the partitioned layer fully ablated, across-batch output variance is 0.0000 without the skip and 0.1896 with it.

With the partitioned layer fully ablated, the reported normalised target score is $24.8 \pm 2.2\%$ without the skip and $99.9 \pm 0.0\%$ with it, at every tested mean shared keep probability (0.90 through 0.15, three seeds). **A residual path around the declared units raises what survives total ablation from 24.8 to ≈100.** This is an extreme intervention: with one partitioned layer and a full-width skip the stem alone solves MNIST. The causal conclusion concerns this MLP. Similar residual paths make bypass a plausible mechanism in the transformer, without identifying its contribution there.

Generated arithmetic and its suppression objective. The arithmetic delivery has cross-entropy 5.150 nats against 5.263 for the retrained-without-arithmetic reference. Its 0.113-nat remaining gap corresponds to **95.4%** gap closure, with -0.6% collateral. This is a normalised corpus-loss comparison, not arithmetic answer accuracy, generalisation to new formats, or a measurement that 95.4% of arithmetic skill has been removed.

The checkpoint comes from a six-domain run (WikiText, OWT, PTB, code, generated arithmetic and citation text) with a nominal 120M-token budget. After 20% of optimiser updates, each update

minimises

$$\frac{1}{6}\sum_{d=1}^{6} L_d(m_d) + 3\max\{5 - L_{\text{arith}}(M_{\text{delete}}), 0\} + \frac{1}{5}L_{r_t}(M_{\text{delete}}), \tag{9}$$

where r_t cycles through retained domains. The two delete-mask terms use separate extra microbatches and disable the stochastic training partition, exposing the exact delete mask. The hinge pushes the deleted model's arithmetic loss up to 5 nats, and the retain term discourages collateral damage. The historical nominal token counter counts only the six base microbatches; it omits these two extra passes after warmup. Before step rounding this adds about 32M tokens to the 120M base budget, for about 152M total. Consequently, this result cannot be attributed to index selection or corpus narrowness alone, nor compared as compute-matched with an untreated 120M-token run. It uses no *post-hoc* repair at delivery.

Equation overlap in the historical validation cache. The original arithmetic stream was generated with replacement. Parsing complete equations finds 3,058,905 occurrences and 1,917,815 distinct ordered $(a, \text{operator}, b)$ identities in the training cache. Validation contains 62,390 complete occurrences; 40,012 (64.13%) have an identity already in that training cache. Among the 61,730 distinct validation identities, 39,593 are shared. The audit omits incomplete boundary fragments when counting complete equations. The trailing training fragment is compatible with 1,000 identities, of which 349 are absent from the complete training equations; these are excluded from any subsequently constructed unseen set as well. This is an overlap audit of the generated arithmetic cache only, not a search of the other text corpora. The frozen protocol and token-cache hashes are stored in `results/revision-20260913/arithmetic-protocol-cpu-audit/protocol.json`.

The composed product: removal, shrinking and RMU. The same interface also combines code-block deletion, regional shrinking and RMU. The archived `drop-b075` mask retains 75% of the *concatenated* retained-private indices, rather than 75% of each block. Its exact regional widths are WikiText 288, OpenWebText 288, Penn Treebank 72, code 0 and shared 384: 1,032 units per layer. It ships 27.719M parameters rather than the 1,248-unit combination's 28.716M. After the same RMU-script relearning schedule, code losses are 2.3341 and 2.3331 nats. The compact product therefore combines all three operations with nearly unchanged measured target suppression: their paired difference is $+0.001008$ nats with a 95% sequence interval $[-0.000183, +0.002180]$. Relative to plain RMU, the compact product retains a target-loss difference of $+0.1554$ nats ($[+0.1426, +0.1679]$) after the same attack. The allocation has a retained-utility cost: Penn Treebank loss rises from 3.6772 to 3.9102 at delivery and from 4.6026 to 4.8713 after relearning. Composition is operationally supported; preserving every domain's quality is an allocation and validation problem.

I.1 Answer-level arithmetic evaluation

The frozen protocol samples 192 ordered equation identities per scenario, uniformly without replacement (seed 20260915). Native-format conditions use addition, subtraction and multiplication with operands in $[0, 999]$: one sample occurs in the arithmetic training cache and another is excluded against all complete equations and compatible trailing fragments. The alternative question format uses the same unseen identities with prompts such as `What is 1 + 187?`, a newline, and `Answer:`; the larger-operand condition uses prompts with operands in $[1000, 1999]$. The latter two are distribution shifts. Exclusion covers the arithmetic cache only, not potential exposure elsewhere in the mixed text corpora.

We evaluate greedy continuation of one equation at a time, stopping at newline, end-of-text, or eight generated tokens. Exact correctness requires the decoded numeric string to equal the answer; an additional score requires observed termination. The two correctness counts agree for every arm here. Teacher-forced answer-token cross-entropy includes the leading answer space and excludes the prompt and terminator. It is distinct from whole-stream language-model loss and therefore cannot be substituted into the historical 95.4% corpus-gap figure.

The joint and adversarial models establish little answer-level competence under these prompts before deletion. On unseen native-format equations, both full models and both deleted products score 0/192; the unpartitioned model scores 2/192 and the arithmetic-only control 5/192. A Wilson 95%

Table 13: Answer-token cross-entropy in nats / number of exact answers out of 192. All arms use the same scenarios and decoding protocol. Specialist/delete arms are static masks of their indicated checkpoint. These are individual historical checkpoints, not training-seed estimates.

checkpoint	product	seen native	unseen native	question	larger operands
joint	full	3.956 / 0	3.756 / 0	7.297 / 0	6.738 / 0
joint	specialist	3.969 / 0	3.790 / 0	6.595 / 0	6.769 / 0
joint	delete	5.907 / 0	5.656 / 0	7.396 / 0	6.683 / 0
adversarial	full	4.005 / 1	3.806 / 0	7.360 / 0	6.826 / 0
adversarial	specialist	3.947 / 1	3.768 / 0	6.894 / 0	6.904 / 0
adversarial	delete	6.052 / 0	5.782 / 0	7.386 / 0	6.662 / 0
unpartitioned	full	3.381 / 1	3.185 / 2	7.773 / 0	6.987 / 0
no arithmetic	full	7.311 / 0	6.915 / 0	8.123 / 0	7.547 / 0
arithmetic only	full	2.941 / 6	2.784 / 5	11.174 / 0	8.061 / 0

interval for 0/192 has upper endpoint 1.96%; for 5/192 the interval is $[1.12, 5.95]\%$. Every arm scores zero in the changed question format and with larger operands. These outcomes prevent a useful estimate of an arithmetic-skill removal rate. They do not establish absence of arithmetic ability under all prompts or decoding methods.

Loss still changes: deleting the arithmetic block raises unseen-native answer-token loss from 3.806 to 5.782 in the adversarial checkpoint, while the no-arithmetic reference is 6.915. The effect depends on format; on larger operands the same loss changes from 6.826 to 6.662. Generation often degenerates after deletion: 146/192 adversarial-delete and 144/192 joint-delete unseen-native continuations hit the eight-token cap, compared with 0/192 for adversarial-full and 1/192 for joint-full. Extending the cap or varying the prompt would be another experiment.

All nine product artifacts, their per-equation answers, cap/termination diagnostics, checkpoint hashes and frozen task list are in `results/revision-20260913/supplement/arithmetic/`. The protocol hash matches the independent CPU-audited protocol exactly. The reference checkpoints have different domain inventories and budgets (the no-arithmetic and arithmetic-only runs use 100M and 20M base tokens); the adversarial objective has the extra passes described above. These are explicit diagnostic references, not a compute-matched comparison of training methods.

J THE SAME CONSTRUCTION AT FOUR TIMES THE SIZE

Penn Treebank holds 1.17M training-cache tokens, so increasing its training budget entails substantial repetition. The scale study replaces it with generated arithmetic, keeps WikiText-103 unchanged, extends OpenWebText and uses CodeParrot in place of the primary study's Python corpus. The nominal 800M-token budget is balanced over four domains. Cache passes are not distinct arithmetic problems: this generator has only 20,000 ordered identities. The earlier results retain their own mixture; this change does not establish broader natural-language domain coverage. `repro/prepare_scale_data.py` prepares the larger OpenWebText and code caches by splitting concatenated tokens, which does not establish document independence. A post-training audit verified an equivalent reconstruction of all 220,000,000 training and 1,100,000 validation arithmetic tokens. The original generation command and environment remain unrecovered; logical token equality does not imply identical serialized files or establish untouched test data. Exact reconstruction arguments and logical hashes appear in `repro/POST_V33_ARITHMETIC_CACHE.md`.

Both rungs use the same data and nominal budget, with two exposure seeds and one factorized seed per size. The exposure arms ramp p_{sub} from 0 to 0.35; the factorized arm never reveals foreign blocks.

model	code loss	composition	products	arithmetic	with all blocks
30M, exposure, seed 0	1.878	+0.007	119/375	192/192	192/192
30M, exposure, seed 1	1.867	+0.005	123/375	192/192	192/192
30M, factorized	1.871	+9.872	48/375	192/192	5/192
124M, exposure, seed 0	1.669	+0.007	118/375	192/192	192/192
124M, exposure, seed 1	1.669	+0.009	110/375	192/192	192/192
124M, factorized	1.663	+11.521	48/375	192/192	53/192

Code loss is the extracted specialist's; composition is what the same domain loses when every block runs at full width, relative to its singleton; its 0.009-nat maximum is not a bound across domains and widths. Products counts grid cells meeting the 0.10-nat tolerances out of 375; arithmetic counts exact answers on held-out identities for the arithmetic specialist product, and the last column repeats that measurement with all blocks active.

The mechanism holds at both sizes, and the arithmetic domain shows the failure it repairs at the level of a task rather than a loss: without exposure the arithmetic specialist answers all 192 evaluated equations, whereas the same weights with all blocks active answer 5/192 (2.6%) at 30M and 53/192 (27.6%) at 124M. With exposure the full models retain 192/192. These are task-level composition results on the generated validation task.

This is a factor of four in parameters on larger token caches, not evidence about billion-parameter models. The corpora differ from the four-domain mixture used elsewhere in the paper, so these numbers replace nothing there; they test whether the construction survives a larger model on the stated changed mixture and larger token caches.

K What the mask distribution contributes

Which aspects of mask coordination change the compact/full-width trade-off? The complete-recipe comparison changes mask dependence, sharing and mean retention together. This five-arm study holds conditional unit retention q_i fixed while varying sharing, with three parents per arm at the standard geometry and ramped foreign exposure.

arm	unit draw	shared across tokens	shared across layers
T: token independent	Bernoulli	no	no
A: batch independent	Bernoulli	yes	no
B: batch nested	nested prefix	yes	no
C: layer independent	Bernoulli	yes	yes
D: layer nested	nested prefix	yes	yes

A nested draw uses one uniform U per region and keeps unit i when $U < q_i$. Decreasing q_i gives a prefix, including an empty prefix when required, with marginal q_i. Retained units are scaled by $1/q_i$ conditional on exposure. Each foreign block receives one exposure gate per layer with probability p, giving own/shared activity q_i and foreign activity pq_i. Sharing the conditional unit draw across layers does not share those layer-local gates.

Correction and controlled reruns. The released forward path applied the exposure gate twice in C and D, giving foreign activity $p^2 q_i$ instead of pq_i. At $p = 0.35$, the gate probability was 0.1225 instead of 0.35. Sampler-only tests missed this complete-forward-path error. The earlier C/D code composition penalties, 0.176 ± 0.016 and 0.084 ± 0.011 nats, were confounded recipe results; we withdraw their causal interpretation as an effect of layer sharing. T, A and B were unaffected.

Integrated regression tests now check own/shared/foreign activity, scaling and cross-layer dependence at $p \in \{0, 0.1, 0.35, 1\}$. Six corrected C/D parents were trained from scratch, using seeds 0–2 and the original 80M nominal-token budget. Each processed 79,994,880 tokens in 3,255 updates; the six reruns processed 479,969,280 training tokens, with no auxiliary training passes. The recipe, endpoints and analysis were fixed before these results. Nine historical T/A/B controls were reused after recovered configuration/source checks and exact CPU/CUDA forward-and-RNG replay of the unaffected paths. A dedicated data-sampling generator prevents changed mask consumption

from changing sampled training windows. Original launch-time source/cache hashes and historical initial-state records remain incomplete: recovered identities support this comparison but are not a cryptographic proof of the original training bytes. No historical provenance was overwritten.

Fixed development panel. All fifteen parents were evaluated on the same 200 cached validation windows per domain (sampling seed 99), using deterministic masks. The 168/384/336/672-unit code products retain shared/private fractions 25/25, 25/100, 50/50 and 100/100%, respectively. Thus the 384-unit product does not contain the 336-unit product. The primary compact endpoint is 168 units; 384 is the other originally specified compact endpoint and 336 is additional. Code composition penalty is all-domain full-width code loss minus the 672-unit code singleton loss with exactly the same shared and code-private indices. The table includes that absolute all-domain code loss as well as the specialist losses. Entries are mean ± sample standard deviation across three parents, in nats.

Arm	168	384	336*	672	All-domain	Comp.
T	2.864 ± 0.053	2.566 ± 0.040	2.438 ± 0.029	2.249 ± 0.031	2.278 ± 0.035	0.029 ± 0.004
A	2.836 ± 0.053	2.591 ± 0.035	2.522 ± 0.035	2.389 ± 0.027	2.420 ± 0.027	0.032 ± 0.004
B	2.598 ± 0.024	2.482 ± 0.027	2.504 ± 0.024	2.448 ± 0.025	2.471 ± 0.024	0.023 ± 0.002
C	2.732 ± 0.039	2.508 ± 0.032	2.497 ± 0.031	2.394 ± 0.028	2.422 ± 0.030	0.029 ± 0.002
D	2.558 ± 0.016	2.477 ± 0.016	2.495 ± 0.016	2.450 ± 0.016	2.468 ± 0.015	0.018 ± 0.004

Mean ± sample SD over three parents. Code loss in nats; *336 units is exploratory. Composition is all-domain code loss minus the matching 672-unit singleton loss.

Seed-level results. The independent replication unit is the trained parent, not the evaluated product or window. Every planned seed is shown below; corrected C/D rows replace the defective weights in the controlled comparison. The archived defective results remain in the supplementary evidence.

Arm	Seed	168	384	336*	672	All-domain code	Composition
T	0	2.8442	2.5452	2.4376	2.2500	2.2782	0.0282
T	1	2.8234	2.5412	2.4090	2.2180	2.2433	0.0253
T	2	2.9246	2.6121	2.4670	2.2790	2.3124	0.0334
A	0	2.8685	2.6029	2.5272	2.3861	2.4223	0.0362
A	1	2.7751	2.5517	2.4841	2.3637	2.3921	0.0284
A	2	2.8649	2.6190	2.5533	2.4166	2.4470	0.0304
B	0	2.5856	2.4710	2.4921	2.4382	2.4595	0.0213
B	1	2.5823	2.4620	2.4869	2.4295	2.4552	0.0257
B	2	2.6253	2.5125	2.5315	2.4770	2.4987	0.0217
C	0	2.7705	2.5391	2.5249	2.4141	2.4435	0.0294
C	1	2.6924	2.4755	2.4646	2.3616	2.3879	0.0263
C	2	2.7343	2.5102	2.5029	2.4057	2.4361	0.0304
D	0	2.5423	2.4642	2.4817	2.4391	2.4543	0.0153
D	1	2.5560	2.4705	2.4902	2.4433	2.4657	0.0224
D	2	2.5745	2.4949	2.5129	2.4680	2.4834	0.0155

Code loss in nats; *336 units is exploratory. Composition is all-domain code loss minus the matching 672-unit singleton loss.

Prespecified contrasts. A−T changes token sharing; B−A and D−C change nesting; C−A and D−B change layer sharing. Differences pair the records by seed, subject to the recovered-control limitation above. Negative values favor the first arm. The intervals use a Student t distribution with two degrees of freedom across the three parent differences. They describe seed variation, not document uncertainty; they are not simultaneous intervals across endpoints. An interval containing zero does not establish equivalence.

Contrast	Endpoint	Δ_0	Δ_1	Δ_2	Mean	95% t interval
$A-T$	168	$+0.0243$	-0.0482	-0.0598	-0.0279	$[-0.1412, +0.0854]$
$A-T$	384	$+0.0577$	$+0.0105$	$+0.0068$	$+0.0250$	$[-0.0455, +0.0955]$
$A-T$	336*	$+0.0897$	$+0.0750$	$+0.0863$	$+0.0837$	$[+0.0646, +0.1027]$
$A-T$	672	$+0.1360$	$+0.1457$	$+0.1376$	$+0.1398$	$[+0.1268, +0.1528]$
$A-T$	Composition	$+0.0080$	$+0.0031$	-0.0030	$+0.0027$	$[-0.0111, +0.0165]$
$B-A$	168	-0.2829	-0.1929	-0.2396	-0.2384	$[-0.3504, -0.1265]$
$B-A$	384	-0.1319	-0.0897	-0.1065	-0.1094	$[-0.1622, -0.0565]$
$B-A$	336*	-0.0352	$+0.0029$	-0.0218	-0.0180	$[-0.0659, +0.0299]$
$B-A$	672	$+0.0521$	$+0.0657$	$+0.0604$	$+0.0594$	$[+0.0424, +0.0764]$
$B-A$	Composition	-0.0149	-0.0027	-0.0087	-0.0087	$[-0.0239, +0.0064]$
$D-C$	168	-0.2282	-0.1364	-0.1598	-0.1748	$[-0.2933, -0.0563]$
$D-C$	384	-0.0748	-0.0051	-0.0152	-0.0317	$[-0.1254, +0.0619]$
$D-C$	336*	-0.0432	$+0.0256$	$+0.0100$	-0.0025	$[-0.0921, +0.0871]$
$D-C$	672	$+0.0249$	$+0.0817$	$+0.0623$	$+0.0563$	$[-0.0154, +0.1280]$
$D-C$	Composition	-0.0141	-0.0039	-0.0149	-0.0110	$[-0.0262, +0.0043]$
$C-A$	168	-0.0980	-0.0827	-0.1306	-0.1038	$[-0.1646, -0.0430]$
$C-A$	384	-0.0638	-0.0761	-0.1088	-0.0829	$[-0.1407, -0.0251]$
$C-A$	336*	-0.0024	-0.0195	-0.0504	-0.0241	$[-0.0846, +0.0364]$
$C-A$	672	$+0.0281$	-0.0021	-0.0109	$+0.0050$	$[-0.0458, +0.0558]$
$C-A$	Composition	-0.0069	-0.0021	$+0.0000$	-0.0030	$[-0.0117, +0.0057]$
$D-B$	168	-0.0433	-0.0262	-0.0508	-0.0401	$[-0.0714, -0.0088]$
$D-B$	384	-0.0067	$+0.0085$	-0.0176	-0.0053	$[-0.0377, +0.0272]$
$D-B$	336*	-0.0104	$+0.0033$	-0.0186	-0.0086	$[-0.0360, +0.0189]$
$D-B$	672	$+0.0009$	$+0.0139$	-0.0091	$+0.0019$	$[-0.0266, +0.0304]$
$D-B$	Composition	-0.0060	-0.0033	-0.0062	-0.0052	$[-0.0092, -0.0012]$

Paired differences in nats (first arm minus second); *336 units is exploratory. Intervals use three parent differences, are pointwise and normality-based, and do not establish equivalence.

Batch nesting lowers the 168-unit code loss by 0.238 nats relative to batch independence, but raises the 672-unit singleton loss by 0.059 nats. Token independence improves that full-width loss by 0.140 nats relative to batch independence. With corrected gating, layer sharing lowers 168-unit loss by 0.104 nats for independent draws (C−A) and 0.040 nats for prefix draws (D−B). Their mean 672-unit differences are $+0.005$ and $+0.002$ nats, with intervals spanning zero; these full-width effects remain unresolved.

The large historical layer-sharing composition penalty does not persist: corrected C and D have mean code penalties 0.029 and 0.018 nats, compared with 0.032 and 0.023 for A and B. Across all fifteen parents and four domains, the largest observed all-domain full-width penalty is 0.0366 nats (WikiText, A, seed 2). This covers sixty domain measurements, not all domain subsets or widths, and is not a population-wide bound. Each parent's absolute full-width losses and worst affected domain are retained in the released report. The findings support recipe-specific compact/full-width trade-offs, not a general necessity for token-independent masks or a universal benefit from coordination.

Historical predictions and evaluation limits. The historical protocol says it was recorded before runs finished; that does not establish that it was frozen before any outcomes were observed. Its compact endpoints were 168 and 384 units; an earlier manuscript substituted 336. The B−A means at 168 and 384 agree with its predicted 0.05–0.30-nat improvement, but the 336-unit difference is only -0.018 nats. The prediction that token sharing changes every endpoint by at most 0.02 nats fails, including the $+0.140$ full-width difference. The original layer-sharing predictions could not be tested with the defective C/D runs. Their corrected replacements do not support the earlier claim of a large composition cost from sharing across layers.

This is development evidence, not independent test confirmation. Cached windows do not establish document independence. Nine representative physical exports per parent also passed the frozen CUDA float32 checks (135 checks in total; four fixed input rows per product), with absolute/relative tolerances $10^{-4}/10^{-5}$. The maximum logit discrepancy was 3.433×10^{-5}; the maximum loss difference was 9.537×10^{-7} nats. These are numerical spot checks, not physical export of every quality evaluation or product in the broader grid. The corrected report, individual evaluations, work

Table 14: Foreign-block exposure reduces domain-composition regret. *Left*: capacity reversals, over 200 nested capacity pairs at fixed domain set with a 0.02-nat tolerance. *Right*: full-width all-domain loss minus the matching full-width singleton loss, in nats for each domain.

	capacity	domain-composition regret			
training	violation rate	wikitext	owt	ptb	**code**
naive	0.00	+1.847	+1.535	+1.694	**+8.428**
joint	0.00	+0.008	+0.001	+0.000	**+0.011**
joint, seed 1	0.00	+0.007	+0.001	+0.001	+0.011
naive, 8×256	0.00	+1.533	+1.372	+1.594	+8.325
joint, 8×256	0.01	+0.011	+0.002	+0.007	+0.012

ledgers and original confounded results accompany the supplement. The tables are generated from the complete corrected report.

L EVERY DOMAIN SUBSET ON THE WIDTH GRID

Historical full-width and capacity comparison. The constant-exposure recipe is labelled joint and the no-exposure recipe naive in this table.

Beyond singleton and four-domain products, the delivery interface accepts intermediate subsets. This audit evaluates all of them on a specified width grid. For each of the fifteen non-empty subsets of the four text domains, and for each combination of five shared retention levels (100, 75, 50, 25, 10%) with the same five levels applied equally to every retained private block, one deterministic mask is applied and every retained domain is scored: 375 products per checkpoint. Two references come from the same grid, so no comparison imports a number from another run: quality loss is measured against the full-width product of the *same* subset, and composition penalty against each retained domain's singleton at *identical* shared and own-block indices.

checkpoint	supported	singletons	pairs	triples	all four	worst penalty
ramp, seed 0	96/375	24/100	37/150	29/100	6/25	+0.078
ramp, seed 1	94/375	24/100	36/150	28/100	6/25	+0.090
ramp, seed 2	93/375	23/100	36/150	27/100	7/25	+0.086
ramp, seed 3	97/375	23/100	36/150	30/100	8/25	+0.084
ramp, seed 4	97/375	23/100	38/150	30/100	6/25	+0.085
constant exposure	99/375	25/100	37/150	29/100	8/25	+0.042
factorized	29/375	21/100	4/150	4/100	0/25	+8.428

Supported means every retained domain of that product stays within 0.10 nats of the same subset's full-width product and within 0.10 nats of its own singleton, the historical tolerances. The counts are an audit of the declared grid, not a selection: choosing winners from these cells would need a further confirmation sample.

Two readings follow. First, the measured composition penalties remain small as domains are added in these ramp parents. On the first ramp parent at full width the worst composition penalty is 0.000 nats for singletons, 0.009 to 0.021 for the six pairs, 0.020 to 0.029 for the four triples and 0.033 for the four-domain product; across all 375 products and all five ramp parents it never exceeds 0.090. The factorized recipe reaches 8.428, and its supported products are almost entirely singletons: 21 of its 29, with none of the 25 four-domain products and only 4 of 150 pairs. Exposure enlarges the supported multidomain catalogue on this declared grid.

Second, the binding constraint on the ramp catalogue is width reduction. Around a quarter of the grid is supported for each ramp parent (93–97/375), compared with 29/375 for the factorized parent, and the ramp failures are concentrated at the aggressive retention levels, where a domain leaves its own full-width reference by more than the tolerance. Reported product families should therefore name

their retained subsets *and* their widths; the tolerances themselves are declared analysis choices, not community standards.

The audit evaluates the recorded validation protocol described in the study design, so it inherits that protocol's limits: it is a complete sweep of the declared grid on these checkpoints, not an untouched test of unseen documents. It does not cover independent widths for individual private regions or physically export all 375 products. Mask/export agreement checks in the hardware appendices apply only to their listed products and numerical settings.

Additional export checks. A post-audit numerical check uses ramp seed 0 at 4× and 16× FFN width. For each checkpoint, nine products cover full-width and compact singleton/pair/triple/all-domain sets plus a compact code specialist. Deterministic masking and physically sliced weights are compared on one fixed 256-token window per domain, in CPU float32 with training masks and scaling disabled. All 18 checks pass the prespecified absolute/relative tolerances $10^{-4}/10^{-5}$. The largest absolute logit discrepancies are 5.15×10^{-5} and 3.34×10^{-5} for the two geometries; loss discrepancy is at most 4.77×10^{-7} nats. These are numerical checks on the listed inputs, not fresh quality estimates or physical export of the complete grid. Exact indices, inputs and identities are recorded in the supplementary `evidence/post_v33/export_checks` records.

M DELETING A SKILL THE MODEL HAS

Appendix I.1 reports that the historical arithmetic experiment could not measure skill removal: its models answered none of 192 held-out equations before deletion, so the 95.4% corpus-loss figure describes suppression from a low-competence starting point. This appendix supplies the missing starting competence and repeats the deletion against it.

Why we changed the arithmetic task. The historical task draws operands up to 999 with addition, subtraction and multiplication, and writes them in a form GPT-2 byte-pair encoding segments inconsistently. Eight pilot runs, one configuration each at 20M tokens on a separately trained, unpartitioned 30M arithmetic specialist, compare task and formatting choices by exact answers on hash-disjoint held-out identities:

operators	operands	number form	exact answers
add, subtract, multiply	≤ 999	as written (historical)	8/192
add, subtract, multiply	≤ 999	spaced digits	5/192
add, subtract, multiply	≤ 99	as written	111/192
add, subtract, multiply	≤ 99	spaced digits	142/192
add, subtract	≤ 999	spaced digits	163/192
add	≤ 999	spaced digits	185/192
add, subtract	≤ 99	spaced digits	**191/192**
add	≤ 99	spaced digits	191/192

At this budget, removing multiplication from the spaced three-digit task changes 5/192 to 163/192; additionally restricting operands to two digits gives 191/192. Spacing digits at the same two-digit, three-operator setting changes 111/192 to 142/192. These one-run pilots guided task selection; they do not prove that the historical corpus is unlearnable.

Competence gate. The recorded competence gate requires the extracted specialist to answer at least 90% of held-out equations. The competent task is addition and subtraction of two-digit operands with spaced digits, 24M generated training tokens, and validation identities separated by a hash of the ordered equation. This excludes the evaluated ordered identities under the generator, but not equivalent problems: 85 of the 89 evaluated additions have a reversed counterpart assigned to training. The complete task has only 20,000 possible ordered identities; 24M cache tokens and repeated cache passes are not that many distinct problems. Task and pilot decisions used this validation setup, so these 192 equations are development evidence, not an untouched confirmation set or a test of new arithmetic families.

Deletion against demonstrated competence. Two five-domain geometries, with two seeds each, add this task to the four text domains and train with the delivered ramped-exposure recipe. The standard geometry assigns arithmetic 232 private units plus 384 shared units, giving a 616-unit specialist from 1,536 FFN units; both seeds fail the gate. The FFN-dominated geometry assigns 924 private plus 1,536 shared units, giving a 2,460-unit specialist from 6,144; both seeds pass. Products are deterministic index masks of one checkpoint; *delete* keeps the shared pool and the other four blocks.

	$4\times$ FFN (gate failed)		$16\times$ FFN (gate passed)	
product	seed 0	seed 1	seed 0	seed 1
specialist	74/192	78/192	**191/192**	**192/192**
full model	54/192	64/192	188/192	187/192
deletion	0/192	0/192	**1/192**	**0/192**
shared pool only	0/192	0/192	0/192	0/192

The same-parent deletion comparisons are **188/192 → 1/192** and **187/192 → 0/192**. The specialists' 99.5% and 100% are separate extraction results. Shared-only accuracy is zero, showing that this route alone does not produce correct answers; it does not show an absence of relevant information in shared computation. Full-parent arithmetic loss increases from 1.087 and 1.088 to 3.289 and 3.200 nats, respectively.

paired outcome, $16\times$ parents	seed 0	seed 1
Full → delete → relearn, correct/192	$188 \to 1 \to 177$	$187 \to 0 \to 179$
Valid integer outputs, full → delete	$192 \to 45$	$192 \to 24$
WikiText CE, delete minus full	-0.000292	$+0.000522$
OpenWebText CE, delete minus full	-0.002882	-0.000952
Penn Treebank CE, delete minus full	$+0.000981$	-0.001137
Code CE, delete minus full	-0.000124	-0.001608

Retained-domain differences use the saved evaluation sample per domain, with the same examples before and after deletion; all are below 0.003 nats in absolute value. No uncertainty over new documents or training seeds is implied by their decimal precision. Relearning here means 2,002,944 actual target tokens. A retrospective audit of the saved generated strings counts a valid integer as one or more single digits separated by spaces, optionally preceded by a minus sign and a space. The drop to 45/192 and 24/192 valid outputs shows that functional disruption includes output-format failure; it does not identify erasure of arithmetic knowledge.

The skill can be relearned. Deletion is structural, and what remains can learn the task again. Training the deletion product on the target's own data, with the removed units held off, recovers it:

attack tokens	seed 0	seed 1
0	1/192	0/192
28,672	2/192	0/192
102,400	3/192	10/192
401,408	93/192	149/192
2,002,944	**177/192**	**179/192**

At 102,400 tokens the two deletion products answer 1.6% and 5.2% correctly; by 2,002,944 tokens they answer 92.2% and 93.2%. Arithmetic cross-entropy recovers from 3.20–3.29 nats to about 1.10, against 1.087–1.088 for the full parents. This recovery curve demonstrates that deletion does not prevent relearning. Without a matched acquisition or recovery control it cannot establish that deletion imposes an unusual delay.

This is a narrow, generated task with an exact answer, chosen because removal can be checked rather than inferred. It supports a claim about that task, not about mathematical reasoning, and the two geometries show the claim depends on the capacity the domain receives: the same deletion on a model that never reached competence measures nothing about skill.

Raw product measurements and answer strings are in `results/revision-20260913/arith_removal/arith2big_rev_s0.json` and the seed-1 counterpart; the corresponding `arith2big_relearn_s0/s1.json` files contain the recovery measurements.

M.1 Exact answers of narrower exports

The two models with the widened FFN and an arithmetic region were exported with the first units of the pool and of each included region at 100, 75, 50 and 25%, for the arithmetic member and the all-region member, with the same 192 equations, prompts and greedy decoding (at most 12 tokens) as above, under the protocol of Appendix G.1. The full-width counts reproduce those above.

Table 15: Exact answers (of 192) of exported members at reduced width, for the two models (seed 0 / seed 1), with the number of outputs that are well-formed integers, parameters and batch-16 forward speed relative to the full network (A100).

member	width	params	correct	integers	speed
arithmetic	100%	34.31M	191 / 192	192 / 192	1.49
	75%	31.47M	186 / 191	192 / 192	1.62
	50%	28.63M	80 / 123	189 / 192	1.76–1.77
	25%	25.80M	0 / 0	0 / 1	1.95
all five	100%	51.31M	188 / 187	192 / 192	1.00
	75%	44.22M	116 / 145	190 / 192	1.14
	50%	37.12M	16 / 36	149 / 140	1.38
	25%	30.04M	0 / 4	128 / 140	1.68

N Foreign-block exposure and candidate-specific support

These historical checkpoints differ in `subset_prob`; shared fraction, within-region keep ranges, geometry and nominal base-token budget are identical. Exposure changes both the available training capacity and the distribution of masks, so this is not an intervention on coverage alone.

Choice of exposure values. Constant exposure at $p = 0.35$ was the first nonzero setting we trained. It is the constant setting in the 30M studies, and every other exposed model uses a linear ramp from 0 to 0.35 over training. The sweep below adds $p = 0$, 0.70 and 1 at 30M with one model per value. At $p = 0.35$ with three foreign regions per layer, a layer includes 1.05 foreign regions on average, none in 27.5% and all three in 4.3% of draws. After the constant setting showed a cost for single-domain members, three schedules were compared under a protocol dated before any of these runs finished; the increasing ramp from 0 to 0.35 was planned as a control arm, selected after the comparison and then evaluated with further seeds (last paragraphs of this appendix).

p_{sub}	specialist	family price	full-width regret	singleton	all-domain
0.00	2.161	-0.001	$+8.428$	5/25	0/25
0.35	2.316	$+0.154$	$+0.011$	6/25	7/25
0.70	2.295	$+0.133$	-0.043	5/25	7/25
1.00	4.468	$+2.306$	-2.077	4/25	8/25

Specialist loss and full-width regret concern code. Family price uses the seed-0 unpartitioned control at 2.162 nats; two further control seeds reach 2.165 and 2.153. The last two columns use candidate-specific reanalysis with a 0.10-nat quality tolerance on each retained domain, 0.10-nat composition tolerance, and a 0.02-nat capacity-reversal policy. The singleton column concerns code; the all-domain column concerns a product retaining and evaluated on all four domains. Appendix D documents the recorded grids and provenance checks.

At $p_{\text{sub}} = 1$, foreign blocks are revealed at every layer, while deterministic specialist performance is substantially worse. Its negative regret means that adding the other blocks helps this domain. Relative support is consequently permissive: the singleton reference is 4.468 nats, versus 2.161–2.316 in the other rows. The counts therefore describe flexibility around different quality levels,

not an absolute ranking of the training methods. Earlier mixed-product envelope counts and their common-bar ranking are withdrawn; they imposed another product's constraints on singletons. One checkpoint per exposure setting does not establish an optimal exposure probability.

Matched-iteration specialists. The controls above are unpartitioned four-domain models. A specialist trained on nothing but its own corpus adds the comparison that controls the passes over the target data. With 20M code tokens each (0.85 passes), the single-domain specialist reaches 2.310 nats. The factorized extraction beats it by $+0.149$, $+0.156$ and $+0.120$ nats over three seeds (mean $+0.142$), and the unpartitioned controls at 2.153–2.165 beat it as well: the foreign corpora train units the specialist never trains. The exposed extraction at 2.316 has a similar measured loss (-0.003 ± 0.028 over three seeds). Thus constant exposure improves composition while reducing this specialist advantage. The larger-budget control reverses the comparison: at 80M code tokens each (3.40 passes), the specialist reaches 1.711 nats, ahead of a 320M-token unpartitioned model at 1.739 by 0.028 nats and of the code product carved from an equally trained factorized partition at 1.744 by 0.033. The transfer advantage observed at 20M target tokens does not persist at 80M in these runs. The 320M comparisons use one seed and do not establish a saturation threshold.

The exposure schedule. The preceding comparisons use constant exposure; here we vary p_{sub} during training. A protocol dated before run completion records three schedules, two seeds each, otherwise the reference recipe: the decreasing ramp $0.35 \rightarrow 0$, 0.35 held for three quarters of training then ramped to 0, and the ramp $0 \rightarrow 0.35$. Its timing does not establish that predictions were frozen before any results were observed. The recorded criteria were code extraction below the original 2.310-nat from-scratch specialist and full-width composition regret within 0.10 nats; the recorded expectation that no arm reaches both was wrong. The increasing schedule reaches both on the two initial seeds, and three further seeds also pass: extraction 2.222–2.279 nats over five seeds against 2.263–2.310 for five from-scratch specialists, 21 of 25 pairwise comparisons won, mean margin 0.04 nats. These are all cross-recipe pairings of five runs per recipe, not 25 independent trials. Full-width composition regret is 0.025–0.033 on code and within 0.040 on every domain, against 0.011 for the constant recipe. Without the optional capacity-reversal constraint, the candidate-specific singleton envelope stays at 6/25 on three of the five ramp seeds and 5/25 on the other two, and the all-domain envelope reaches 6/25 on each of the three evaluated seeds under descriptive pairing, against 8/25 for the constant recipe. The $0.35 \rightarrow 0$ direction also passes both bars, more weakly (extraction 2.270–2.273, regret 0.074–0.082); holding 0.35 for three quarters gives extraction loss 2.273–2.311, so one of its two seeds misses the original specialist threshold. The increasing schedule has slightly higher full-width composition regret than constant exposure, while remaining below the 0.10-nat tolerance. The separate capacity-reversal policy compares widths at a fixed domain set. The specialist bar exists only for code, the one domain with from-scratch checkpoints.

Extraction, export and removal with ramped exposure. The remaining operations were also evaluated under the increasing schedule. Extraction reaches 2.251 ± 0.026 nats over the five ramp seeds against 2.288 ± 0.020 over the five specialists, Cohen's d 1.6 (means and sample standard deviations; ten training runs). The geometry replicates: five $16\times$ ramp seeds ship 32.3M of 51.3M parameters at 1.530–$1.553\times$ measured batch-16 throughput for 0.042–0.049 nats against their full-width specialists; the worst full-width code regret over all ten ramp seeds of both geometries is 0.040. Deleting a domain's block costs the deleted domain 0.523 to 1.576 nats over fourteen seed–domain deletions (code on all five seeds, the other domains on seeds 0, 1 and 4; mean 0.94), more than the constant recipe's 0.32 to 0.98. This measures greater sensitivity to block deletion, without identifying where domain information is stored. The removal pipeline transfers and replicates: deleting the code domain's block before the same RMU and relearning recipe, on each of the five ramp seeds, leaves code loss 0.082 to 0.140 nats higher after the 2M-token attack than the same single unpartitioned RMU baseline (mean $+0.113$), at 0.416 to 0.475 nats mean retained-domain cost (mean $+0.443$), on 28.7M parameters. The constant-recipe pipeline table (Table 12, Appendix I) supplies the three-way comparison. The five ramp seeds vary partitioned pretraining only; these differences do not estimate variation in the RMU baseline or isolate deletion's effect from the different pretraining procedures. The RMU and relearning random seed is fixed across these runs. Reproduction commands and the recorded schedule protocol are in `repro/SCHEDULE.md` and `evidence/revision/psched/PROTOCOL.md` in the source package.

O HISTORICAL PREFIX-SAMPLING COMPARISON

The historical checkpoint labelled `g_deliv` draws retained fractions from $\{1.0, 0.75, 0.5, 0.25, 0.1\}$ and applies a prefix within each revealed region. Its measured losses remain useful observations about that recipe, but two implementation details prevent interpreting it as a controlled comparison with training the actual exported products.

Layer coordination. Each transformer layer independently draws both its domain subset and its regional widths. Exported products instead use the same index vector at every layer. At $p_{\text{sub}} = 0.35$, the probability of drawing a particular singleton product with specified shared and private widths in all six layers during a relevant-domain forward is $[(1 - 0.35)^3/25]^6 \approx 1.76 \times 10^{-12}$. Sampling a prefix in every layer therefore does not establish exposure to that whole-network product.

Foreign-block scaling. The independent-dropout arm scales units by their retention probability conditional on the foreign block being exposed. The historical prefix arm divides by marginal retention, including the exposure probability. At $p_{\text{sub}} = 0.35$ this gives an exposed foreign block an additional multiplier of $1/0.35 \approx 2.86$. The two recipes therefore differ in activation scale as well as mask dependence and prefix structure.

	specialist loss		worst elastic reversal	
domain	independent	historical prefix	independent	historical prefix
code	2.316	2.693	0.0093	0.0000
wikitext	4.885	5.299	0.0271	0.0000
owt	5.185	5.535	0.0261	0.0000
ptb	3.670	4.018	0.0225	0.0000

The historical prefix arm has negative worst composition regret on all four domains ($-0.012, -0.014, -0.015, -0.011$ nats), and full-width specialist losses increase by $0.377, 0.414, 0.350, 0.348$ nats. These are descriptive differences between confounded recipes, not the price of exact deployment-mask training. Relative support counts also depend on each model's full-width reference and cannot establish superiority across models with different absolute losses. A deployment-aligned recipe must coordinate the delivery across layers and use a stated scaling convention. Neither correction alone isolates the effect of coordination.

Scope of the revised sampler comparison. The 8M-token development pilot's `direct_full50` arm draws one mask for all tokens and layers in a domain microbatch and uses foreign-unit scaling conditional on block exposure. Its regional retained fraction is a 50% point mass at one plus a 50% uniform draw from $\{1, 0.75, 0.5, 0.25, 0.1\}$. Thus each visible region is full-width with probability 0.6 and has mean retained fraction approximately 0.76 (up to integer rounding), compared with 0.70 for private regions and 0.60 for the pool under the independent recipe. Foreign blocks retain the same exposure probability 0.35. This is a comparison of *sampler packages*: coordination, prefix dependence and retention marginals change together. It cannot attribute a difference to coordination alone. Doing so would require an additional comparison with matched retention marginals. Appendix P reports the separately evaluated full-budget comparison of these sampler packages.

P FULL-BUDGET SAMPLER-PACKAGE COMPARISON

The 8M-token, seed-0 pilot selected `direct_full50` for a full-budget comparison with independent dropout. Both use the 30M-parameter, six-layer model and the same nominal 80M-token budget: 79,994,880 actual tokens per run, 19,998,720 per domain, 3,255 AdamW updates with four domain microbatches each, and no auxiliary objective. The coordinated arm makes 13,020 whole-network mask draws. Foreign-block exposure remains 0.35 and foreign parameters receive updates in both arms. As Appendix O explains, the variant changes retention marginals as well as coordination and prefix dependence; this experiment compares the complete sampler recipes.

The frozen primary products retain $(25\%, 25\%)$, $(25\%, 100\%)$ and $(100\%, 100\%)$ of the pool and each specialist's private block, respectively: 168, 384 and 672 units per layer. These are exact prespecified index sets, not the best allocation selected at each size. The secondary grid contains all nine pool/block pairs from $\{25\%, 50\%, 100\%\}^2$.

Evaluation uses the same 128 context-256 sequences per domain for both arms (draw seed 20260916). Token-stream half B was used during pilot development, so this is a frozen-product, fresh-training-seed comparison, not an untouched or document-disjoint test. Paired 95% intervals use 2,000 bootstrap resamples of evaluation sequences. They condition on each checkpoint pair and the sampled, potentially overlapping sequences; they do not measure training-seed uncertainty and are not simultaneous intervals. Reporting two seed-specific effects describes limited between-run variation; it does not supply a robust population interval.

Both prespecified training seeds, 1 and 2, are complete. We report each seed separately; averages across the two seeds are descriptive.

Table 16: Every primary signed effect, in nats: coordinated `direct_full50` minus independent cross-entropy at the same product. Negative values mean lower loss. Intervals describe paired evaluation-sequence uncertainty within each seed.

domain	units	seed 1 Δ	95% interval	seed 2 Δ	95% interval
WikiText	168	-0.458	$[-0.476, -0.442]$	-0.479	$[-0.498, -0.462]$
WikiText	384	-0.299	$[-0.314, -0.284]$	-0.329	$[-0.344, -0.315]$
WikiText	672	$+0.062$	$[+0.053, +0.071]$	$+0.033$	$[+0.024, +0.042]$
OWT	168	-0.448	$[-0.465, -0.429]$	-0.479	$[-0.497, -0.460]$
OWT	384	-0.310	$[-0.324, -0.295]$	-0.334	$[-0.350, -0.319]$
OWT	672	$+0.059$	$[+0.044, +0.075]$	$+0.035$	$[+0.027, +0.044]$
PTB	168	-0.514	$[-0.535, -0.491]$	-0.588	$[-0.616, -0.562]$
PTB	384	-0.277	$[-0.290, -0.263]$	-0.341	$[-0.358, -0.324]$
PTB	672	$+0.044$	$[+0.034, +0.054]$	$+0.035$	$[+0.026, +0.043]$
code	168	-0.392	$[-0.414, -0.369]$	-0.492	$[-0.518, -0.465]$
code	384	-0.211	$[-0.228, -0.193]$	-0.294	$[-0.315, -0.274]$
code	672	$+0.083$	$[+0.070, +0.097]$	$+0.019$	$[+0.011, +0.029]$

Both seeds have lower loss on all eight primary smaller products (168 and 384 units), and higher loss on all four 672-unit specialists. Every corresponding paired-sequence interval excludes zero. This repeats a size–quality tradeoff under the specified protocol, not overall superiority.

Absolute-quality support and composition. Within each training-seed pair, both arms face the same domain-specific threshold: the independent full-width specialist's loss plus 0.10 nats. A singleton is judged only on its retained domain. An all-domain product must meet all four thresholds and its worst loss increase relative to its own matching-width singleton must be at most 0.10 nats. No capacity-reversal constraint is applied here. Counts cover the entire nine-cell grid and use unrounded point estimates, not confidence bounds. The thresholds are shared between methods within a seed but differ between seeds; these counts must not be conflated with the historical 25-cell, same-family-relative envelopes.

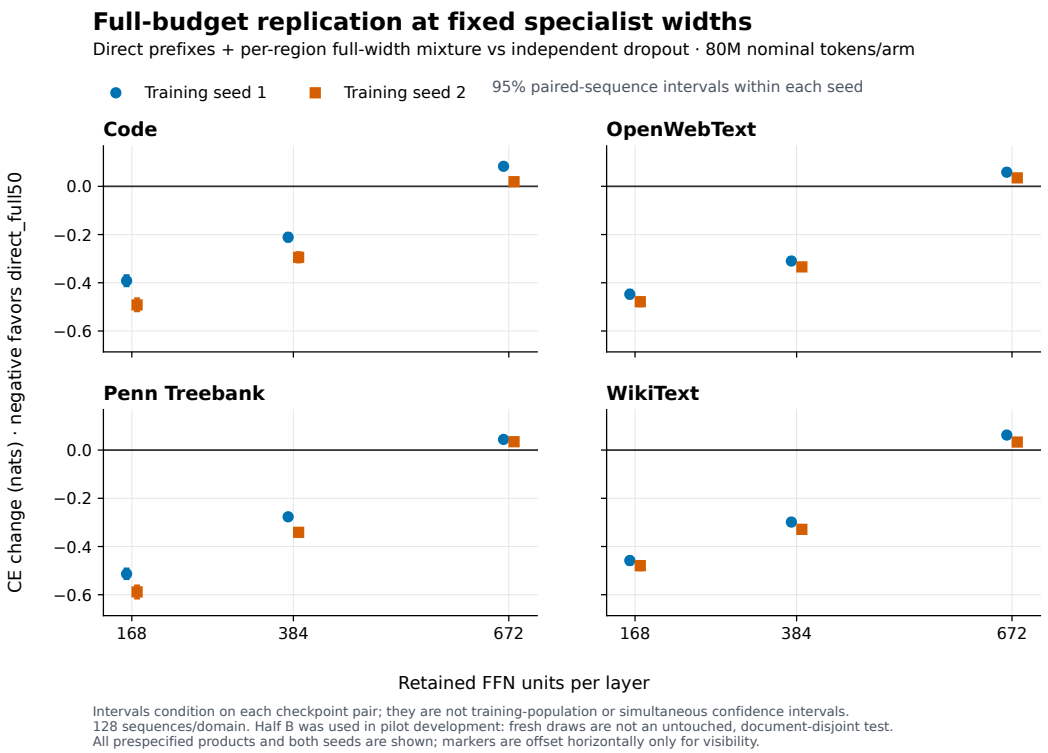


Figure 4: All primary products on both nominal 80M-token training seeds. The 672-unit point is a full-width specialist, not the 1,536-unit all-domain model. The 50% full-width mixture applies to each visible region; it is not a 50% rate of full-network passes. This comparison changes coordination and retention marginals together.

Table 17: Common absolute-quality support: count out of nine and minimum supported FFN units per layer. All-domain rows additionally enforce composition tolerance. Widths refer to the complete retained product, so singleton and all-domain widths have different meanings.

seed	retained set	CE threshold	independent: count; min	coordinated: count; min
1	WikiText	5.095	2/9; 528	3/9; 480
1	OWT	5.136	3/9; 456	4/9; 456
1	PTB	3.613	2/9; 528	4/9; 384
1	code	2.303	2/9; 528	2/9; 480
1	all four	all four bars	3/9; 960	1/9; 1536
2	WikiText	5.136	2/9; 528	6/9; 336
2	OWT	5.168	3/9; 456	6/9; 336
2	PTB	3.651	2/9; 528	6/9; 336
2	code	2.360	2/9; 528	8/9; 240
2	all four	all four bars	3/9; 960	5/9; 768

Table 18: Full-width composition regret ρ: all-domain loss minus the corresponding full-width specialist loss. The last column is the coordinated-minus-independent regret difference with a paired sequence interval. Smaller regret need not mean better absolute loss.

seed	domain	$\rho_{\mathrm{independent}}$	$\rho_{\mathrm{coordinated}}$	change [95% interval]
1	WikiText	$+0.0043$	-0.0025	-0.0067 $[-0.0114, -0.0018]$
1	OWT	-0.0029	-0.0008	$+0.0021$ $[-0.0029, +0.0071]$
1	PTB	$+0.0070$	$+0.0052$	-0.0018 $[-0.0058, +0.0021]$
1	code	$+0.0046$	$+0.0071$	$+0.0026$ $[-0.0020, +0.0071]$
2	WikiText	$+0.0038$	$+0.0002$	-0.0036 $[-0.0082, +0.0009]$
2	OWT	$+0.0019$	-0.0011	-0.0029 $[-0.0068, +0.0010]$
2	PTB	$+0.0083$	$+0.0005$	-0.0078 $[-0.0115, -0.0041]$
2	code	$+0.0153$	$+0.0050$	-0.0103 $[-0.0151, -0.0058]$

Across both seeds and all four domains, the coordinated recipe lowers loss by 0.392–0.588 nats at 168 units and 0.211–0.341 at 384, while 672-unit specialist loss rises by 0.019–0.083: a compact-against-full-width tradeoff, measured here between complete recipes whose retention also differs. Appendix K reports matched-retention token/batch contrasts and six corrected layer-sharing reruns; the original layer-shared arms were confounded by double exposure gating.

The all-domain support comparison changes direction between seeds: independent has 3/9 products in both, while coordinated has 1/9 at seed 1 and 5/9 at seed 2, so the support counts move from 3/9 to 1/9 at seed 1 and from 3/9 to 5/9 at seed 2. Thus improved common-bar family support is not stable across these two runs. The difference comes from absolute quality; composition alone passes all nine cells for both recipes in both seeds.

Deletion diagnostic. We also remove the code block, leaving 1,248 units per layer, and evaluate all four domains. Table 19 shows deleted-minus-full loss within each recipe and the difference of those changes. A smaller code loss increase means less sensitivity to this ablation; it does not establish stronger or weaker knowledge removal without a retrained-without-code reference, demonstrated target skills and adaptation tests. Retained-domain changes are shown alongside the target; they do not erase absolute-quality differences between recipes.

Table 19: Effect of code-block deletion in nats. Each within-recipe column is deleted-minus-full cross-entropy; the last is coordinated change minus independent change, with a paired sequence interval. Code is the omitted target and the other three domains are retained.

seed	domain	independent change	coordinated change	difference [95% interval]
1	WikiText	-0.0069	-0.0027	$+0.0042$ $[+0.0020, +0.0065]$
1	OWT	-0.0010	-0.0039	-0.0030 $[-0.0053, -0.0008]$
1	PTB	-0.0035	-0.0029	$+0.0006$ $[-0.0011, +0.0022]$
1	code	$+0.5951$	$+0.2987$	-0.2964 $[-0.3149, -0.2774]$
2	WikiText	-0.0057	-0.0022	$+0.0035$ $[+0.0017, +0.0054]$
2	OWT	-0.0030	-0.0014	$+0.0016$ $[-0.0004, +0.0035]$
2	PTB	-0.0027	-0.0000	$+0.0027$ $[+0.0009, +0.0045]$
2	code	$+0.6000$	$+0.3245$	-0.2756 $[-0.2960, -0.2550]$

At seed 1, deleted-product code losses are 2.803 nats (independent) and 2.592 (coordinated), a difference of -0.211 $[-0.231, -0.190]$ nats.

At seed 2, deleted-product code losses are 2.875 nats (independent) and 2.609 (coordinated), a difference of -0.267 $[-0.289, -0.244]$ nats.

The machine-readable confirmation report and its referenced evaluation artifacts record every evaluated grid cell, paired loss vector, exact index set, source/checkpoint hash and training/evaluation provenance. It strictly checks the frozen configuration, matching evaluation examples and completed run state before including a seed. These results concern the prespecified sampler packages and product grid, not an overall method ranking or out-of-distribution guarantee.

Q Frozen evaluation on external sources

Does the composition result extend beyond the development caches? This evaluation fixes an external document panel and compares existing ramp-exposure and no-exposure parents without further training. It is separate from the sampler-package comparison above: no coordinated-prefix parent is evaluated here. Unknown historical use of official test splits and incomplete historical source provenance prevent calling this an untouched in-distribution test. The sources and screening rules were fixed before model scoring; no result caused a document, checkpoint or product replacement.

Sources and grouping. The panel contains 128 documents from each of Wikipedia (20231101.en), C4 general web and news-like web, and CodeParrot Python. They are routed to the WikiText, OpenWebText, Penn Treebank and code-assigned blocks, respectively. News-like C4 is not Penn Treebank, and the primary code parents were trained on installed Python sources, not

CodeParrot. CodeParrot-trained scale checkpoints are excluded. These bounded source files do not represent their entire corpora.

We inspect at most 20,000 rows of one pinned file per source. Eligible documents have 512–8,192 GPT-2 text tokens, followed by one end-of-text token. Deterministic hash ranking selects at most 1,024 candidates per source, one per article, hostname or code repository. The general-web candidates exclude hostnames in the bounded news-like input; Wikipedia title keys are checked against preserved WikiText article titles. Code license labels and header filters restrict eligibility but are not comprehensive redistribution clearance.

An overlap screen compares all 4,096 candidates against 274,123,511 logical tokens in the current four-domain training/development caches and the WikiText/PTB test caches whose historical use is unknown. It rejects candidates when at least 20% of sampled 32-token windows occur in those references or a sampled 256-token span matches exactly. Candidate starts are spaced 16 tokens apart; reference starts are exhaustive, and hash matches are token-verified. Exclusions propagate through components linked by source group, normalised full text or sampled exact spans. One encyclopedia and 25 Python candidates are excluded. The final panel retains one document per component, 128 per source, with no shared component across sources. This checks lexical overlap with current caches, not every paraphrase or unknown historical exposure. A data-only protocol amendment improved title normalisation and license-header matching before the final panel and before any model outcome; both versions remain recorded.

Table 20: External panel: each selected document is a distinct final source component. Targets include end-of-text and exclude each document's first token.

Source	Assigned block	Documents/groups	Scored target tokens
Wikipedia	WikiText	128	255,661
C4 general web	OpenWebText	128	129,901
C4 news-like web	Penn Treebank	128	122,600
CodeParrot Python	code	128	275,347

Frozen models, products and scoring. Each recipe has three existing 30M parents, with six layers, hidden width 384, FFN width 1,536, a 384-unit shared pool and four 288-unit private blocks. Equal seed labels do not establish matched training trajectories; between-recipe comparisons use unpaired means of three parents. The fixed products are all 15 nonempty full-width domain subsets and a code singleton retaining 25% of both regions: 168 FFN units per layer against a 672-unit full-width code singleton. This is not the 51.3M-to-32.3M export in the throughput study.

All products use identical document windows. Scoring resets context and position every 256 targets, scores each target once except the first document token, includes end-of-text and ignores tail padding. Deterministic masks disable stochastic sampling and training scaling. The primary endpoints are each source's all-domain absolute loss and composition penalty relative to the same parent's singleton with identical shared and own-private indices. Compact-code shrinking compares with that parent's full-width code singleton. The other subset results, observed maxima and failures are retained as prespecified secondary summaries.

Parent variation and data uncertainty. Tables 21 and 24 show every parent. The accompanying machine-readable report additionally gives recipe means and sample standard deviations for every product and metric. Pointwise 95% intervals use 2,000 source-group bootstrap draws with the same resampled groups across products and all six parents within a source (PCG64 seed 260917). They quantify data uncertainty conditional on these fixed parents, not uncertainty over trained parents. Documents, products and bootstrap draws do not increase the three-parent replication count. The observed panel maximum and the chosen 0.10-nat failure criterion are descriptive; neither is a simultaneous statistical guarantee.

Table 21: Full-width all-domain and singleton losses for every evaluated parent. Negative differences mean that adding the other blocks improves the source's prediction loss.

Recipe	Seed	External source	All-domain	Singleton	Difference
Ramp	0	Wikipedia	5.7140	6.1280	-0.4140
Ramp	0	C4 web	5.6100	5.5946	+0.0154
Ramp	0	C4 news-like	5.4009	5.7896	-0.3887
Ramp	0	CodeParrot	2.9493	2.9292	+0.0200
Ramp	1	Wikipedia	5.6592	6.0698	-0.4106
Ramp	1	C4 web	5.5867	5.5774	+0.0093
Ramp	1	C4 news-like	5.3829	5.7310	-0.3481
Ramp	1	CodeParrot	2.9108	2.8934	+0.0174
Ramp	2	Wikipedia	5.7669	6.1701	-0.4032
Ramp	2	C4 web	5.6489	5.6288	+0.0201
Ramp	2	C4 news-like	5.4523	5.8188	-0.3665
Ramp	2	CodeParrot	2.9844	2.9613	+0.0230
No exposure	0	Wikipedia	7.1954	6.5563	+0.6391
No exposure	0	C4 web	6.8862	5.5243	+1.3619
No exposure	0	C4 news-like	6.7403	7.0886	-0.3483
No exposure	0	CodeParrot	10.2097	2.8126	+7.3971
No exposure	1	Wikipedia	7.2096	6.5564	+0.6532
No exposure	1	C4 web	6.8684	5.5241	+1.3444
No exposure	1	C4 news-like	6.7267	7.1068	-0.3800
No exposure	1	CodeParrot	10.6575	2.7924	+7.8652
No exposure	2	Wikipedia	7.3622	6.6241	+0.7381
No exposure	2	C4 web	6.9687	5.5773	+1.3914
No exposure	2	C4 news-like	6.8614	7.2380	-0.3766
No exposure	2	CodeParrot	10.2795	2.8474	+7.4322

Losses are token-weighted nats on external documents. Each row uses one parent. Difference is all-domain full-width loss minus that parent's singleton loss with the same shared and own-domain indices. Wikipedia, C4 web, C4 news-like and CodeParrot Python are routed to the WikiText, OWT, PTB and code blocks, respectively; these are not the original test splits.

Table 22: Primary external endpoints: absolute quality and composition, with conditional source-group uncertainty. Equal-weight recipe means retain all three parents.

External source	Endpoint	Ramp mean	No-exp. mean	Difference	Conditional 95% interval
Wikipedia	Absolute loss	5.7134	7.2557	-1.5424	[-1.6000, -1.4929]
Wikipedia	Composition	-0.4093	0.6768	-1.0860	[-1.1392, -1.0369]
C4 web	Absolute loss	5.6152	6.9078	-1.2926	[-1.3455, -1.2400]
C4 web	Composition	0.0149	1.3659	-1.3510	[-1.4131, -1.2917]
C4 news-like	Absolute loss	5.4120	6.7762	-1.3641	[-1.4260, -1.3202]
C4 news-like	Composition	-0.3678	-0.3683	+0.0006	[-0.0334, +0.0363]
CodeParrot	Absolute loss	2.9481	10.3823	-7.4341	[-7.8641, -6.9746]
CodeParrot	Composition	0.0202	7.5648	-7.5447	[-7.9664, -7.0950]

Full-width all-domain products; differences are ramp minus no exposure. Each recipe mean gives equal weight to three parents. Composition is loss minus the same parent's matching singleton. Intervals are pointwise 95% percentile intervals from 2,000 source-group resamples shared across products and parents. They measure data uncertainty conditional on these six fixed parents, not training-parent uncertainty. Seed labels do not establish matched parent histories; no simultaneous panel guarantee is claimed.

Composition and its limits. The full all-domain ramp model has lower mean absolute loss on all four sources. Its composition penalty is smaller than without exposure on Wikipedia, general web and Python, but the news-like contrast is unresolved: $+0.0006$ nats with conditional interval $[-0.0334, +0.0363]$. Both recipes improve news-like loss by combining all four blocks. The largest positive all-domain ramp penalty is 0.0230 nats, but that is not the maximum across intermediate assemblies.

Table 23 includes all 15 full-width subsets. Ramp parents have maxima 0.0906, 0.1082 and 0.1045 nats; two of 96 retained-domain/parent cases exceed 0.10. Both failures concern news-like text in the WikiText+PTB pair. Without exposure, maxima are 7.5003, 7.8771 and 7.4322 nats, with 59 of 96 cases above 0.10. These results support substantially fewer severe composition failures under ramp exposure on this panel, not a universal 0.04-, 0.090- or 0.10-nat bound.

Table 23: All full-width subsets, including the intermediate-assembly failures. Counts concern retained-domain cases, not independent parent replications.

Recipe	Seed	Worst source	Product ID	Max. comp.	> 0.10
Ramp	0	C4 news-like	`domains-0-2-s100-p100`	+0.0906	0/32
Ramp	1	C4 news-like	`domains-0-2-s100-p100`	+0.1082	1/32
Ramp	2	C4 news-like	`domains-0-2-s100-p100`	+0.1045	1/32
No exposure	0	CodeParrot	`domains-0-2-3-s100-p100`	+7.5003	19/32
No exposure	1	CodeParrot	`domains-0-2-3-s100-p100`	+7.8771	20/32
No exposure	2	CodeParrot	`domains-0-1-2-3-s100-p100`	+7.4322	20/32

Observed composition penalties in nats across all 15 full-width nonempty subsets: 32 retained-domain cases per parent. Each penalty is product loss minus the same parent's full-width singleton loss on that retained source. The count is the number of cases strictly above the chosen 0.10-nat criterion, not a count of products. Domain indices 0/1/2/3 denote the WikiText/OWT/PTB/code-assigned blocks; s100/p100 denotes full regional widths. The compact code product is excluded. These maxima and failure counts describe the evaluated panel, without a simultaneous or universal guarantee.

Extraction and shrinking trade-offs. The full-width code singleton averages 2.9280 nats with ramp exposure versus 2.8174 without it: a $+0.1106$-nat cost, with conditional source-group interval $[+0.0998, +0.1226]$. The 168-unit ramp specialists instead have lower loss than the no-exposure specialists at the same width, but still cost 0.5775–0.6403 nats relative to their own 672-unit references (Table 24). Safer composition does not imply better quality for every extracted product, and this severe width reduction does not inherit the historical larger model's 0.04–0.05-nat shrinking cost.

Table 24: Code specialist shrinking on the external Python panel, with the same-parent, same-domain full-width reference. All six evaluated parents are retained.

Recipe	Seed	168-unit loss	672-unit reference	Shrinking cost
Ramp	0	3.5067	2.9292	+0.5775
Ramp	1	3.5337	2.8934	+0.6403
Ramp	2	3.5705	2.9613	+0.6092
No exposure	0	3.8519	2.8126	+1.0393
No exposure	1	3.6819	2.7924	+0.8896
No exposure	2	3.9162	2.8474	+1.0688

CodeParrot Python loss in nats. The 168-unit code singleton retains 25% of both its shared and private regions; the reference is the same parent's 672-unit full-width code singleton. Shrinking cost is compact minus reference loss. Widths are FFN units per layer; each row uses one parent.

Export checks, actual work and reproducibility. Nine fixed representative products per parent undergo masked-versus-physical-export checks on fixed inputs in CUDA FP32 with TF32 disabled and prespecified tolerances `atol=1e-4`, `rtol=1e-5`. All 54 checks pass. They include compact, singleton, pair, triple and all-domain cases; they do not physically export every evaluated product or check every document. The evaluation scores 96 products, 198 retained-domain cases and 39,260,514 target tokens using 42,748,416 padded forward positions. Export verification adds 108 forward passes and 110,592 positions. The A100 launch takes 263 seconds, with no optimiser updates; this is an evaluation work record, not a training or serving-speed comparison.

The release records exact source revisions, file hashes, grouping/screening rules, checkpoint identities, retained indices, document losses, work counts and the frozen analysis. Data preparation was

independently replayed; a separate numerical audit recomputed every document aggregate and all 891 conditional intervals. The first reporting attempt rejected the equivalent device labels `cuda:0` and `cuda`. A recorded validation-only amendment accepts that alias while replaying the unchanged frozen numerical analysis; raw results and statistical rules were preserved. The report and table mapping are under `results/revision-20260917/external-analysis-02/`; acquisition instructions accompany the screening manifests. Raw external documents and reconstructive token caches are not redistributed.

R ABLATION OF REGIONAL BLOCKS AND ORDERING

Beyond the measured effect of foreign-block exposure, the question is what each ingredient supplies. We train them separately on CIFAR-10 MLP, with a width-sampling baseline from the slimmable family for contrast.

Table 25: Regional-block and ordering ablations. 400 units, 5 class groups, 10 seeds for blocks only and blocks plus ordering, and 3 for the other conditions. Entries are means ± sample standard deviations. Selection is target-group accuracy with prediction restricted to that group's classes. Subtraction is reference-gap closure: $100(A_{\text{extract}} - A_{\text{delete}})/(A_{\text{extract}} - A_{\text{no-target}})$, computed per seed, where the reference is trained without the target group.

training	select (↑)	subtract (↑)	note
ordinary	n/a	n/a	no assigned domain block
blocks only (flat keep)	$0.9076 \pm .005$	$18.0 \pm 7.2\%$	
ordering only (no blocks)	$0.9025 \pm .003$	$\mathbf{0.0 \pm 0.0}\%$	select = subtract arm
blocks + ordering	$0.9092 \pm .004$	$23.4 \pm 12.9\%$	
width sampling, no partition	$0.8447 \pm .012$	$15.3 \pm 12.4\%$	masking damage, not removal

Domain-labelled removal. Ordering alone gives $0.0 \pm 0.0\%$. Without domain-labelled blocks, its selection and subtraction arms are identical. These ablations characterize operations within the tested recipes; broader method families may support additional operations.

Incremental effect of ordering. Adding ordering to blocks changes subtraction by $+5.3$ points, 95% CI $[-3.0, +14.3]$, and selection by $+0.0016$, $[-0.0018, +0.0052]$, at n=10. These intervals leave the incremental selection and subtraction effects unresolved at this sample size. Figure 2 illustrates the two controls available in the combined construction. A baseline without a partition reports $15.3 \pm 12.4\%$ under the same subtraction score, reflecting arbitrary masking damage. This result is within one reported standard deviation of the combined construction, whose across-seed variability is roughly twice that of blocks alone.

Prefix selection across architectures. Comparing a prefix of k units against an *arbitrary* k units of the same model isolates what the ordering is worth from how much capacity was removed. Entries below are prefix-minus-random differences in accuracy averaged over all five class groups, using within-group predictions. We report means ± sample standard deviations across three seeds.

units removed	CIFAR-10 MLP (400)	CIFAR-10 CNN (256 ch.)
50%	$+0.0545 \pm .032$	$-0.0106 \pm .002$
75%	$\mathbf{+0.0749 \pm .041}$	$\mathbf{-0.0743 \pm .029}$
90%	$+0.0612 \pm .010$	$-0.0362 \pm .080$

On the CNN, mean prefix performance is lower than arbitrary-slice performance at all three tested widths. Capability selection worked across the tested substrates, whereas ordering improved size selection on the MLP and transformer but not this CNN. Possible explanations include channel ordering, batch normalisation and the use of 256 channels; these mechanisms remain untested. On the transformer, after repair of the pruned members, ordering improves loss by $+0.084$ nats at 90% removal, compared with -0.004 to $+0.002$ for the flat-droprate control; Section 5.2 reports the

comparison without repair. The benefit was negligible below ≈50% removal, concentrating the observed advantage at deeper cuts.

S COMBINED REMOVAL AND RECOVERY UNDER ADAPTATION

Table 26: Suppression and specified relearning attacks. Gap closure normalises the target loss increase by the gap to a model retrained without the target. Collateral averages retained-domain loss increases normalised by their unigram gaps, except the last row, which uses nats. The last four columns are target cross-entropy in nats after each stated attack-token budget. These are corpus-loss metrics, not measures of skill removal or worst-case attack resistance. Every row uses one checkpoint.

artifact	units	gap score	collateral ↓	25k	100k	400k	2000k
RMU, post-hoc	1536	92.8%	0.5%	5.094	2.894	2.215	2.179
gradient ascent, post-hoc	1536	198.3%	2.1%	2.503	2.351	2.237	2.194
ablation only	1324	11.3%	0.2%	2.451	2.365	2.288	2.213
partition + RMU	1248	110.9%	0.6%	4.957	3.304	2.492	2.333
adversarial objective	1248	133.2%	−1.4%	3.521	3.058	2.656	2.394
adversarial, 10% shipped	125	32.8%	+1.12 nats	3.579	3.157	2.776	2.487

The direct optimiser comparison. The matched RMU experiment is the central combined-method comparison. Partitioning, removing the code block and applying RMU yields higher target loss than plain RMU at delivery and after the nominal 100k, 400k and 2M-token relearning budgets. At 25k tokens the ordering reverses (4.957 versus 5.094 nats), so the advantage is not uniform throughout adaptation. The longer-budget result demonstrates useful composition of structural removal and optimisation under the tested schedule. Table 26 also reports the retained-loss collateral and other interventions; Appendix I gives the direct paired intervals and full-pipeline utility tradeoff.

A separate reference for learning from the target corpus. The six historical artifact reports end 0.309 nats apart after their nominal 2M-token attacks, despite much larger differences at delivery. A model pretrained without the target also learns from the attack corpus: the separate attack-floor control starts at 7.658 nats and ends at 2.750, a 4.907-nat improvement. This establishes that substantial recovery can come from the supplied data, but the floor is not a fully matched counterfactual for every artifact. In particular, its source defaults use batch 24 and draw code evaluation first, whereas the direct RMU comparison uses batch 16 and draws code after the retained domains. The table preserves these historical values as descriptive context.

delivered artifact	after 2M attack	vs. plain initial	gap to separate floor
RMU, post-hoc	2.179	+0.017	**+0.571**
gradient ascent, post-hoc	2.194	+0.033	+0.556
index-only	2.213	+0.052	+0.537
index + RMU	2.333	+0.172	+0.417
adversarial	2.394	+0.233	+0.356
adversarial, 10% shipped	2.487	+0.326	**+0.263**

The last column subtracts each reported artifact loss from the separate floor value. It is not a causal estimate of information supplied by the released weights. In contrast, the direct RMU pair uses the same attack and evaluation recipe, so its positive 0.154-nat difference after the longest measured attack does not depend on that floor. The aggressively thinned adversarial artifact retains still higher target loss, with the larger retained-utility cost shown in Table 26.

The attacks use the training corpus under specified adaptation schedules. Other budgets, objectives or data could change recovery and the method ranking. We therefore report target performance, retained utility and the adaptation protocol together.

T EVERY CONFIGURATION

The historical "guard flags" count measured loss-ordering reversals. They are diagnostics, not proofs of invalidity or failed training exposure. A dash marks configurations where that diagnostic was not applied to the adversarially trained mask. Gap closure is normalised by the difference between the original and retrained-without-target losses; "all ablated" reports the same quantity with every declared unit ablated. It is a reference, not a ceiling. Values outside $[0, 100]\%$ are possible; when the reference gap is nonpositive or nearly zero, the percentage has no useful removal interpretation. Such rows are retained here as historical records.

In the transformer table, s_f is the shared fraction and "sub" is the foreign-block exposure probability. "Units", "ext" and "del" give FFN units per layer in the full model, the target specialist and the product after target-block deletion, respectively. "Attn" and "rnn" denote attention and recurrent token mixers. Labels identify archived evaluation records.

T.1 TRANSFORMER CONFIGURATIONS

label	target	s_f	sub	mixer	units	ext	del	flags	gap/all ablated
a_flat	code	0.25	0.35	attn	1536	672	1248	0	11.4/31.2
a_noshared	code	0.0	0.35	attn	1536	384	1152	0	15.1/16.8
a_noshared_s1	code	0.0	0.35	attn	1536	384	1152	0	14.6/17.1
a_noshared_s2	code	0.0	0.35	attn	1536	384	1152	0	14.7/17.3
a_nosubset	code	0.25	0.0	attn	1536	672	1248	11	173.2/41.4
a_nosubset_s1	code	0.25	0.0	attn	1536	672	1248	11	182.4/42.0
a_nosubset_s2	code	0.25	0.0	attn	1536	672	1248	11	165.5/40.6
a_sub70	code	0.25	0.7	attn	1536	672	1248	0	5.8/32.8
adv_blockonly	code	0.25	0.35	attn	1536	672	1248	–	76.2/38.4
adv_l1	code	0.25	0.35	attn	1536	672	1248	–	80.4/33.7
adv_l3	code	0.25	0.35	attn	1536	672	1248	–	79.5/33.6
adv_nomu	code	0.25	0.35	attn	1536	672	1248	–	74.8/30.4
adv_poolonly	code	0.25	0.35	attn	1536	672	1248	–	121.8/20.9
adv_tau40	code	0.25	0.35	attn	1536	672	1248	–	95.6/33.5
adv_tau65	code	0.25	0.35	attn	1536	672	1248	–	156.2/33.0
adv_theory	code	0.25	0.35	attn	1536	672	1248	–	133.2/31.5
adv_theory_s1	code	0.25	0.35	attn	1536	672	1248	–	136.3/30.3
adv_theory_s2	code	0.25	0.35	attn	1536	672	1248	–	147.3/30.5
cvr_a_nosubset	code	0.25	0.0	attn	1536	672	1248	13	173.2/41.4
cvr_a_nosubset_s1	code	0.25	0.0	attn	1536	672	1248	13	182.4/42.0
cvr_a_nosubset_s2	code	0.25	0.0	attn	1536	672	1248	13	165.5/40.6
cvr_g_sf25_t35	code	0.25	0.35	attn	1536	672	1248	1	12.3/31.4
cvr_g_sf25_t35_s1	code	0.25	0.35	attn	1536	672	1248	1	11.7/30.7
cvr_geo2_joint	code	0.25	0.35	attn	1024	448	832	1	14.3/37.1
cvr_geo2_naive	code	0.25	0.0	attn	1024	448	832	11	184.9/59.2
dom_a_nosubset_code	code	0.25	0.0	attn	1536	672	1248	13	173.2/41.4
dom_a_nosubset_owt	owt	0.25	0.0	attn	1536	672	1248	14	-2488.7/-2870.3
dom_a_nosubset_ptb	ptb	0.25	0.0	attn	1536	672	1248	14	-7943.5/-6418.3
dom_a_nosubset_wikitext	wikitext	0.25	0.0	attn	1536	672	1248	15	-2423.3/-2062.5
dom_a_sub100_code	code	0.25	1.0	attn	1536	672	1248	0	-63.1/52.5
dom_a_sub100_owt	owt	0.25	1.0	attn	1536	672	1248	0	87.4/-125.5
dom_a_sub100_ptb	ptb	0.25	1.0	attn	1536	672	1248	0	91.1/-104.6
dom_a_sub100_wikitext	wikitext	0.25	1.0	attn	1536	672	1248	0	85.4/-123.2
dom_a_sub70_code	code	0.25	0.7	attn	1536	672	1248	0	5.8/32.8
dom_a_sub70_owt	owt	0.25	0.7	attn	1536	672	1248	0	-57.6/-774.3
dom_a_sub70_ptb	ptb	0.25	0.7	attn	1536	672	1248	0	-250.4/-1271.8
dom_a_sub70_wikitext	wikitext	0.25	0.7	attn	1536	672	1248	0	-99.6/-637.0
dom_g_deliv_code	code	0.25	0.35	attn	1536	672	1248	0	1.7/27.9
dom_g_deliv_owt	owt	0.25	0.35	attn	1536	672	1248	0	-2.2/-285.2
dom_g_deliv_ptb	ptb	0.25	0.35	attn	1536	672	1248	0	-12.1/-353.6
dom_g_deliv_wikitext	wikitext	0.25	0.35	attn	1536	672	1248	0	-12.7/-207.6
dom_g_sf25_t35_code	code	0.25	0.35	attn	1536	672	1248	1	12.3/31.4
dom_g_sf25_t35_owt	owt	0.25	0.35	attn	1536	672	1248	1	-163.8/-726.9
dom_g_sf25_t35_ptb	ptb	0.25	0.35	attn	1536	672	1248	1	-733.8/-1359.4
dom_g_sf25_t35_s1_code	code	0.25	0.35	attn	1536	672	1248	1	11.7/30.7
dom_g_sf25_t35_s1_owt	owt	0.25	0.35	attn	1536	672	1248	1	-174.8/-920.1
dom_g_sf25_t35_s1_ptb	ptb	0.25	0.35	attn	1536	672	1248	1	-814.7/-1631.7
dom_g_sf25_t35_s1_wikitext	wikitext	0.25	0.35	attn	1536	672	1248	1	-209.2/-702.0
dom_g_sf25_t35_wikitext	wikitext	0.25	0.35	attn	1536	672	1248	1	-196.2/-594.0
dom_geo2_joint_code	code	0.25	0.35	attn	1024	448	832	1	14.3/37.1
dom_geo2_joint_owt	owt	0.25	0.35	attn	1024	448	832	1	-92.0/-458.2
dom_geo2_joint_ptb	ptb	0.25	0.35	attn	1024	448	832	1	-293.8/-721.0
dom_geo2_joint_wikitext	wikitext	0.25	0.35	attn	1024	448	832	1	-110.8/-370.1
dom_geo2_naive_code	code	0.25	0.0	attn	1024	448	832	11	184.9/59.2
dom_geo2_naive_owt	owt	0.25	0.0	attn	1024	448	832	12	-677.9/-1252.2

label	target	s_f	sub	mixer	units	ext	del	flags	gap/all ablated
dom_geo2_naive_ptb	ptb	0.25	0.0	attn	1024	448	832	11	-1319.3/-1853.1
dom_geo2_naive_wikitext	wikitext	0.25	0.0	attn	1024	448	832	13	-654.2/-951.2
env_a_nosubset	code	0.25	0.0	attn	1536	672	1248	13	173.2/41.4
env_a_nosubset_s1	code	0.25	0.0	attn	1536	672	1248	13	182.4/42.0
env_a_nosubset_s2	code	0.25	0.0	attn	1536	672	1248	13	165.5/40.6
env_g_sf25_t35	code	0.25	0.35	attn	1536	672	1248	1	12.3/31.4
env_g_sf25_t35_s1	code	0.25	0.35	attn	1536	672	1248	1	11.7/30.7
env_geo2_joint	code	0.25	0.35	attn	1024	448	832	1	14.3/37.1
env_geo2_naive	code	0.25	0.0	attn	1024	448	832	11	184.9/59.2
g_pk040	code	0.25	0.35	attn	1536	672	1248	0	7.3/13.8
g_pk090	code	0.25	0.35	attn	1536	672	1248	0	19.4/26.0
g_pk100	code	0.25	0.35	attn	1536	672	1248	1	26.0/26.7
g_pk100_s6	code	0.25	0.35	attn	1536	672	1248	0	26.9/36.6
g_sf10_t35	code	0.1	0.35	attn	1536	501	1189	0	13.7/20.1
g_sf10_t68	code	0.1	0.35	attn	1536	501	1189	0	13.2/16.9
g_sf25_t00	code	0.25	0.35	attn	1536	672	1248	0	6.8/44.6
g_sf25_t35	code	0.25	0.35	attn	1536	672	1248	0	12.3/31.4
g_sf25_t35_s1	code	0.25	0.35	attn	1536	672	1248	0	11.7/30.7
g_sf25_t35_s2	code	0.25	0.35	attn	1536	672	1248	0	11.6/31.3
g_sf25_t68	code	0.25	0.35	attn	1536	672	1248	0	12.8/18.5
g_sf25_t89	code	0.25	0.35	attn	1536	672	1248	0	14.3/15.6
g_sf45_sub0_s1	code	0.45	0.0	attn	1536	903	1324	9	162.5/57.5
g_sf45_sub0_s2	code	0.45	0.0	attn	1536	903	1324	10	164.8/53.2
g_sf45_t35_s0	code	–	–	attn	1536	903	1324	0	9.3/45.5
g_sf45_t35_s1	code	0.45	0.35	attn	1536	903	1324	0	11.3/43.7
g_sf45_t35_s2	code	0.45	0.35	attn	1536	903	1324	0	9.0/50.2
g_sf45_t68	code	0.45	0.35	attn	1536	903	1324	0	13.5/22.0
graded_light	code	–	–	attn	1536	903	1324	10	147.5/51.1
graded_sh25	code	–	–	attn	1536	672	1248	11	173.9/24.1
graded_sh50	code	–	–	attn	1536	960	1344	10	141.7/32.9
j320_plain	code	–	–	attn	1536	672	1248	0	-304.0/260.7
j320_sub0_vs80m	code	0.25	0.0	attn	1536	672	1248	11	171.3/27.4
kk_a_nosubset	code	0.25	0.0	attn	1536	672	1248	13	173.2/41.4
kk_g_sf25_t35	code	0.25	0.35	attn	1536	672	1248	1	12.3/31.4
kk_geo2_joint	code	0.25	0.35	attn	1024	448	832	1	14.3/37.1
n6_adv_arith	arith	0.25	0.35	attn	1536	576	1344	–	95.4/114.2
n6_arith	arith	0.25	0.0	attn	1536	576	1344	22	373.7/131.3
n6_cite	cite	0.25	0.0	attn	1536	576	1344	19	194.2/103.2
plain_joint_80m	code	–	–	attn	1536	903	1324	0	-11.3/83.0
plain_joint_80m_s1	code	–	–	attn	–	–	–	0	–/–
plain_joint_80m_s2	code	–	–	attn	–	–	–	0	–/–
res_sub0	code	0.25	0.0	attn	1536	672	1248	17	309.5/167.7
res_t35	code	0.25	0.35	attn	1536	672	1248	6	145.0/170.1
res_t68	code	0.25	0.35	attn	1536	672	1248	4	261.0/179.6
res_t79	code	0.25	0.35	attn	1536	672	1248	8	296.0/184.1
rnn_joint	code	–	–	rnn	1536	903	1324	0	-9.3/109.8
rnn_sub0	code	0.25	0.0	rnn	1536	672	1248	11	147.6/48.1
rnn_sub35	code	0.45	0.35	rnn	1536	903	1324	0	5.9/53.3
sub0_vs_compute_matched	code	0.25	0.0	attn	1536	672	1248	13	173.2/41.4
tgt_owt	owt	0.25	0.0	attn	1536	672	1248	14	153.3/176.8
tgt_ptb	ptb	0.25	0.0	attn	1536	672	1248	14	123.5/99.8
tgt_wikitext	wikitext	0.25	0.0	attn	1536	672	1248	15	146.7/124.9

T.2 SMALL-SUBSTRATE CONFIGURATIONS

label	data/arch	units	groups	seeds	s_f	routing	width samp.	flags (mean)
cifar10-cnn-g5-h256-t0-shrink-cnn	cifar10/cnn	256	5	3	0.45	–	–	0.0
cifar10-cnn-g5-h256-t0	cifar10/cnn	256	5	3	0.45	–	–	0.0
cifar10-cnn-g5-h64-t0	cifar10/cnn	64	5	3	0.45	–	–	0.0
cifar10-mlp-g5-h100-t0	cifar10/mlp	100	5	3	0.45	–	–	0.7
cifar10-mlp-g5-h400-t0-c10-blocksonly	cifar10/mlp	400	5	3	0.45	–	–	1.7
cifar10-mlp-g5-h400-t0-c10-both	cifar10/mlp	400	5	3	0.45	–	–	1.7
cifar10-mlp-g5-h400-t0-c10-orderingonly	cifar10/mlp	400	5	3	1.0	–	–	0.0
cifar10-mlp-g5-h400-t0-c10-usnet	cifar10/mlp	400	5	3	0.45	–	40,400	4.0
cifar10-mlp-g5-h400-t0-comp-joint	cifar10/mlp	400	5	3	0.45	–	–	1.7
cifar10-mlp-g5-h400-t0-comp-naive	cifar10/mlp	400	5	3	0.45	–	–	9.0
cifar10-mlp-g5-h400-t0-s10-blocksonly	cifar10/mlp	400	5	10	0.45	–	–	1.3
cifar10-mlp-g5-h400-t0-s10-both	cifar10/mlp	400	5	10	0.45	–	–	1.4
cifar10-mlp-g5-h400-t0-shrink-mlp	cifar10/mlp	400	5	3	0.45	–	–	1.7
cifar10-mlp-g5-h400-t0	cifar10/mlp	400	5	3	0.45	–	–	1.7
cifar100-cnn-g5-h128-t0	cifar100/cnn	128	5	3	0.45	–	–	0.0
cifar100-cnn-g5-h512-t0	cifar100/cnn	512	5	3	0.45	–	–	0.0
mnist-mlp-g5-h100-t0	mnist/mlp	100	5	3	0.45	–	–	0.0
mnist-mlp-g5-h200-t0-base-usnet	mnist/mlp	200	5	3	0.45	–	20,200	0.0
mnist-mlp-g5-h200-t0-dec-blocksonly	mnist/mlp	200	5	3	0.45	–	–	0.0

label	data/arch	units	groups	seeds	s_f	routing	width samp.	flags (mean)
mnist-mlp-g5-h200-t0-dec-both	mnist/mlp	200	5	3	0.45	–	–	0.0
mnist-mlp-g5-h200-t0-dec-orderingonly	mnist/mlp	200	5	3	1.0	–	–	0.0
mnist-mlp-g5-h200-t0-k15	mnist/mlp	200	5	3	0.45	–	–	0.7
mnist-mlp-g5-h200-t0-k30	mnist/mlp	200	5	3	0.45	–	–	0.7
mnist-mlp-g5-h200-t0-k45	mnist/mlp	200	5	3	0.45	–	–	0.0
mnist-mlp-g5-h200-t0-k60	mnist/mlp	200	5	3	0.45	–	–	0.0
mnist-mlp-g5-h200-t0-k75	mnist/mlp	200	5	3	0.45	–	–	0.0
mnist-mlp-g5-h200-t0-k90	mnist/mlp	200	5	3	0.45	–	–	0.0
mnist-mlp-g5-h200-t0-nores-k15	mnist/mlp	200	5	3	0.45	last-layer	–	0.0
mnist-mlp-g5-h200-t0-nores-k30	mnist/mlp	200	5	3	0.45	last-layer	–	0.0
mnist-mlp-g5-h200-t0-nores-k60	mnist/mlp	200	5	3	0.45	last-layer	–	0.0
mnist-mlp-g5-h200-t0-nores-k90	mnist/mlp	200	5	3	0.45	last-layer	–	0.0
mnist-mlp-g5-h200-t0-res-k15	mnist/mlp	200	5	3	0.45	skip	–	0.0
mnist-mlp-g5-h200-t0-res-k30	mnist/mlp	200	5	3	0.45	skip	–	0.0
mnist-mlp-g5-h200-t0-res-k60	mnist/mlp	200	5	3	0.45	skip	–	0.0
mnist-mlp-g5-h200-t0-res-k90	mnist/mlp	200	5	3	0.45	skip	–	0.0
mnist-mlp-g5-h200-t0	mnist/mlp	200	5	3	0.45	–	–	0.0
mnist-mlp-g5-h40-t0	mnist/mlp	40	5	3	0.45	–	–	0.0

U THE 1.3B MODELS

One model per recipe was trained with the ramp from 0 to 0.35 over the full budget or without exposure, for 99,183 updates, on web, science, question-answer and code data. The models have 24 layers, width 2,048, 16 attention heads, a context of 1,024 tokens and the GPT-2 vocabulary of 50,257 tokens. Each feed-forward layer has 8,192 units, a shared pool of 2,048 and one region of 1,536 per domain, with drop-rate profiles $(0.10, 0.50)$ in the regions and $(0.30, 0.50)$ in the pool, for 1,313,626,112 parameters. Each update processes 64 sequences of 1,024 tokens per domain, so each domain contributes 6,500,057,088 training tokens and each model 26,000,228,352. The data are source-stratified shards of Dolma v1.7 (Soldaini et al., 2024): Common Crawl for web, peS2o for science, RedPajama StackExchange for question-answer and a 1/16 hash sample of StarCoder for code. Training used AdamW with $\beta = (0.9, 0.95)$, $\epsilon = 10^{-8}$, weight decay 0.1 on matrices and none on vectors, a peak learning rate of 3×10^{-4} after 992 linear warm-up updates with cosine decay to 3×10^{-5}, gradient clipping at 1.0, and BF16 autocast with FP32 parameters and optimiser state, on eight A100-80GB GPUs per model for about 105 hours. Both models were evaluated with deterministic masks on a held-out test split of about 10M tokens per domain, assigned by source group and screened against training and development data by exact hashing and near-duplicate detection with word 5-shingles at Jaccard similarity 0.8; no global near-duplicate removal within the training data was performed. Whole documents are scored in windows of at most 1,024 tokens and losses are aggregated over tokens.

Table 29: Loss (nats) of the single-domain members and the all-region member of the two 1.3B models on the held-out test split after full training. One model per recipe. The table does not show members of reduced width.

	No exposure		Ramp	
Domain	single-domain	all regions	single-domain	all regions
Web	3.227	3.775	3.242	3.237
Science	2.500	2.914	2.511	2.512
Q&A	1.441	1.860	1.452	1.451
Code	0.907	1.325	0.914	0.912

Over the 28 cases of the 11 combinations of two or more regions, the largest composition penalty was 0.549 nats without exposure and 0.008 nats with the ramp, and 18 and 0 cases exceeded 0.10 nats. Removing one region from the ramp model raised the loss of its domain by 0.132 to 0.372 nats and changed the other domains by less than 0.011 nats (Table 30). Nine members were also exported and compared with the masked model: 2 of 26 member and domain checks exceeded the FP32 tolerance (absolute 10^{-4}, relative 10^{-5}), and all 26 agreed in FP64.

Table 30: Removing one region from the 1.3B ramp model after full training: increase in the loss of the removed region's domain on the held-out test split, in nats. The largest absolute change on any other domain was 0.0102 nats. One trained model; the table does not show the model trained without exposure.

removed region	loss increase
web	0.372
science	0.273
question-answer	0.253
code	0.132

U.1 Program synthesis

The 1.3B models were also scored on the 257 problems of the sanitized MBPP test split (Austin et al., 2021), a public benchmark whose absence from the training data was not verified. Each problem was shown with one solved example and its first test assertion; answers were decoded greedily in FP32 with at most 256 new tokens, unlike the original three-shot protocol. A problem counted as solved when the generated code ran and passed all original assertions. Members were exported physically from the checkpoints after 84,000 of the 99,183 updates. Table 31 lists members at full width before any further training; no member solved more than 7 of the 257 problems.

Table 31: MBPP problems solved (of 257) by exported 1.3B members at full width, after 84,000 of 99,183 updates and before any further training. One model per recipe. MBPP is public and was not checked for overlap with the training data. The table does not show members after relearning or unlearning.

recipe	all regions	all regions except code	pool and code region
without exposure	0	1	7
ramp	3	3	3

V Exact answers of combined members

Table 32: Exact answers on 192 held-out arithmetic equations and code composition penalty in the second mixture (WikiText-103, OpenWebText, CodeParrot, arithmetic). The arithmetic member contains the shared pool and the arithmetic region; the all-region member contains every region. Each row is one trained model. The table does not show other domains or reduced widths.

size	exposure	seed	arithmetic member	all-region member	code penalty (nats)
30M	ramp	0	192/192	192/192	0.007
30M	ramp	1	192/192	192/192	0.005
30M	none	0	192/192	5/192	9.872
124M	ramp	0	192/192	192/192	0.007
124M	ramp	1	192/192	192/192	0.009
124M	none	0	192/192	53/192	11.521

W Adding a domain with replay

This screen asks how a trained model changes when a new domain is added to a region that was reserved for it. A 30M-parameter parent (six layers, hidden width 384, context 256, one seed) with a shared pool and three domain regions was trained on web, science and question-answer data, 16.8M target tokens per domain; a fourth region, reserved for code, was inactive during this training and its output weights were zero when the extension began. The screen therefore uses reserved capacity and does not measure the cost of growing a model. All twelve continuations start from the same

parent and receive the same 4.2M code target tokens. They vary whether the old domain regions can update, whether they are exposed during the code training, and the fraction of old-domain replay: 0, 1 or 10% of the processed code tokens, with the replay computation counted separately. Attention, embeddings and the shared pool remain trainable.

Table 33 lists the results on sixteen complete development documents per domain. Raising replay from 0 to 10% reduced the mean loss increase on the old domains in 4 of 4 matched comparisons. Freezing the old regions reduced it in 1 of 6 comparisons. The ramp reduced the largest composition penalty relative to no exposure in 6 of 6 comparisons. All comparisons are within one parent and are not independent replications.

Table 33: Adding a code domain to a three-domain parent. Old-domain changes use the single-domain members of the parent before the extension as reference; composition penalties use the single-domain members after the extension with the same shared and own-region units. Largest penalty and exceedances cover the 28 cases in combinations of two or more regions; exceedances use 0.10 nats. Losses in nats. One parent; sixteen development documents per domain.

old regions	exposure	replay (%)	code loss	mean old-domain change	largest penalty	> 0.10
trainable	none	0	3.544	$+0.604$	$+0.185$	9/28
trainable	none	1	3.544	$+0.376$	$+0.357$	9/28
trainable	none	10	3.534	$+0.216$	$+0.433$	10/28
trainable	ramp	0	3.568	$+0.545$	$+0.093$	0/28
trainable	ramp	1	3.567	$+0.407$	$+0.155$	2/28
trainable	ramp	10	3.556	$+0.217$	$+0.187$	5/28
frozen	none	0	3.544	$+0.608$	$+0.187$	9/28
frozen	none	1	3.543	$+0.413$	$+0.238$	11/28
frozen	none	10	3.536	$+0.224$	$+0.226$	8/28
frozen	ramp	0	3.569	$+0.551$	$+0.122$	1/28
frozen	ramp	1	3.570	$+0.407$	$+0.162$	2/28
frozen	ramp	10	3.560	$+0.217$	$+0.215$	6/28

The development parent and the limited continuation budget do not establish behaviour after full pretraining or across parent seeds.

X RESCALING REGIONS OF MODELS TRAINED WITHOUT EXPOSURE

This analysis tests whether the failure of combined members trained without exposure disappears when region outputs are rescaled. It uses existing checkpoints and forward passes only; the protocol was committed at 09:12:49 UTC on 25 September 2026 (manifest SHA-256 `2e43a792...`), before the first forward pass. It fixed the arms, the set of factors, the selection of each factor on validation half A by the equally weighted mean loss over the four domains, with ties going to the smaller factor, and the evaluation on half B. Factors multiply region outputs after the activation and before the output projection; biases are unchanged and nothing is retrained. Each model trained without exposure is compared with the ramp model of the same mixture and seed. Arms: U, unscaled; M, all domain regions multiplied by $1/4$; S, all domain regions multiplied by one factor s, the same for every input; oracle-M and oracle-S, only the regions of domains other than the evaluated one multiplied by $1/4$ or by s, which requires the evaluated domain to be known. s was chosen separately for S and oracle-S from $\{0.1, 0.2, 0.3, 0.5, 0.7\}$ on half A and applied unchanged on half B. Where the choice fell on 0.1, the smallest value, smaller factors were not tested. Penalties are code loss of the all-region member minus that of the code member with the same indices, on half B; intervals are 95% bootstrap intervals over sequences and condition on the trained model. Arithmetic counts exact answers of the all-region member on the 192 held-out equations. One model per recipe and size.

Bootstrap intervals for S: $[0.906, 0.969]$, $[1.004, 1.063]$ and $[2.931, 3.107]$ nats; for oracle-S: $[0.029, 0.034]$, $[0.062, 0.072]$ and $[0.749, 0.843]$ nats. The penalties of U on half B differ from those in the main text (8.428, 9.872 and 11.521 nats) because they use half B only.

Table 34: Code composition penalty (nats, half B) and exact arithmetic answers of the all-region member for models trained without exposure under five rescaling arms, and for the ramp model of the same mixture without rescaling. The table does not show other domains or reduced widths.

model	U	M	S (s)	oracle-M	oracle-S (s)	ramp
main 30M, penalty	8.448	1.044	0.937 (0.2)	0.271	0.032 (0.1)	0.022
second 30M, penalty	9.879	1.636	1.033 (0.1)	0.648	0.067 (0.1)	0.006
second 30M, answers	5	0	0	192	192	192
second 124M, penalty	11.576	6.806	3.020 (0.1)	4.146	0.796 (0.1)	0.005
second 124M, answers	53	1	0	192	192	192

X.1 Weighting by a domain posterior

This comparison asks how unweighted combination after exposure compares with an input-dependent weighting of the regions, as used by mixtures of domain experts (Gururangan et al., 2022). It is an adaptation to our existing checkpoints, not a reproduction of DEMix architecture or training, and an additional analysis of the previously scored external panel, not a new test. The protocol and code were committed before the first forward pass (manifest SHA-256 `d7a132fb...`).

At each token, the outputs of the domain regions are weighted by the posterior of each domain given the tokens already observed in the same 256-token evaluation window, computed from the single-domain members of the same model with a uniform prior; the weights sum to one, the shared pool and the output bias are used once, and no domain labels are used. For the prediction of token x_t in a window whose first input x_1 is not scored, the weight of region d is

$$w_{d,t} = \frac{\exp \sum_{s=2}^{t-1} \log p_d(x_s \mid x_{<s})}{\sum_{e=1}^{K} \exp \sum_{s=2}^{t-1} \log p_e(x_s \mid x_{<s})}, \tag{10}$$

where p_d is the single-domain member of domain d and $K = 4$, so the first prediction uses $w_{d,2} = 1/K$. The same weights multiply the output of region d after the activation and before the output projection in every FFN layer; the sums restart in every 256-token window, with a uniform prior and without temperature or smoothing, and are accumulated in double precision. When the posterior is one-hot, the weighted member equals the single-domain member. Four arms were scored: A, trained without exposure and unweighted; B, the same models weighted; C, ramp models unweighted; D, ramp models weighted. B and D need four single-domain passes and one weighted pass per token.

Table 35: Loss (nats) of the all-region member on the frozen external panel, averaged over three models per recipe, and the difference C − B with a 95% bootstrap interval over source groups that conditions on the six trained models. The table does not show members of reduced width.

source (region)	A	B	C	D	C − B
Wikipedia (WikiText)	7.256	5.681	5.713	5.716	$+0.032$ $[0.021, 0.042]$
C4 web (OpenWebText)	6.908	5.555	5.615	5.603	$+0.060$ $[0.048, 0.077]$
C4 news-like (Penn Treebank)	6.776	5.355	5.412	5.408	$+0.057$ $[0.042, 0.080]$
CodeParrot (code)	10.382	2.830	2.948	2.937	$+0.118$ $[0.106, 0.131]$

On the 192 held-out equations, the weighted members answered 60 (30M) and 57 (124M) without exposure, against 5 and 53 unweighted, and 180, 179, 157 and 181 with the ramp, against 192 unweighted. With Python text before each equation, all arms answered at most 2 of 192. Scoring the weighted members took 5.64 to 5.68 times as long as the unweighted members in this implementation on one A40 GPU, without key-value caching; this is a measurement of our implementation, not a lower bound for optimised routing. Unweighted losses reproduce Appendix Q within 10^{-5} nats, and unweighted answer counts reproduce Appendix Y.

Constant weights. An additional controlled analysis of the already scored panel replaces the posterior weights of arm B by constant weights of 1/4 on the four regions of the same three models without exposure (arm E), with the pool and the output bias used once and the same windows, targets and aggregation; no factor was selected. Weights of one reproduced arm A and one-hot weights

the single-domain members, and each model had 783,509 scored targets. The comparison was specified with the export catalogue of Appendix G.1 (same protocol and commit). Intervals are paired bootstraps over source groups (2,000 draws, seed 260917), conditional on the three models.

Table 36: Constant weights (E) against posterior weights (B) for the three models without exposure, and against the unweighted ramp members (C), on the frozen external panel. Losses in nats, averaged over three models; differences with 95% bootstrap intervals.

source (region)	E	B	E − B	E − C
Wikipedia (WikiText)	5.969	5.681	$+0.287\ [0.268, 0.307]$	$+0.256\ [0.236, 0.274]$
C4 web (OpenWebText)	5.836	5.555	$+0.281\ [0.258, 0.303]$	$+0.221\ [0.191, 0.247]$
C4 news-like (Penn Treebank)	5.670	5.355	$+0.314\ [0.298, 0.331]$	$+0.258\ [0.232, 0.281]$
CodeParrot (code)	3.622	2.830	$+0.792\ [0.762, 0.824]$	$+0.674\ [0.640, 0.710]$

X.2 Joint activity and foreign updates

With exposure, a foreign region is active in the same forward pass as the region of the training domain and also receives gradients from that domain's data. Four arms on the main 30M mixture separate these: the ramp; joint activity without foreign updates; foreign updates without joint activity; and no exposure. All other settings are those of the ramp: architecture, drop-rate profiles $(0.10, 0.50)$ and $(0.30, 0.50)$, independent draws per token, unit and layer, rescaling by $1/q_r(i)$ conditional on inclusion, region gates per layer shared over examples and tokens, $p(t) = 0.35\, t/T$, data and order, 3,255 updates with one microbatch per domain, and optimiser.

In the arm with joint activity only, the forward pass equals that of the ramp, but the contributions of included foreign regions are computed with detached weights, so that gradients reach their inputs and not their weights. In the arm with foreign updates only, a main pass with $p = 0$ is followed by an auxiliary pass on the same microbatch in which the region of the training domain is off and each foreign region is included with probability $p(t)$; of the auxiliary gradients, only those on foreign-region weights are kept, with weight 1/4. Foreign regions thus receive gradients from each domain's data with the same probability as with the ramp but are never active together with that domain's region; they can be active together with each other and with the pool. The auxiliary objective and gradient magnitudes differ from those of the ramp, and the arm costs one additional forward and backward pass per domain microbatch, four per update.

All twelve models (four arms, seeds 0, 1 and 2) were trained from one code version, committed with the protocol, the evaluation and the sentences for each outcome at 16:36 UTC on 25 September 2026, after an earlier version of this comparison (below) was known and before any of the twelve models was trained. Within each seed, the four arms shared the initial state and the data order, and each ran 3,255 updates. Before training, the regression tests passed on every GPU of every run: the forward pass and random-number use matched the original implementation at $p = 0$, 0.1, 0.35 and 1; at $p = 0$, losses were bit-identical across arms, the ramp and the arm with joint activity only had identical forward passes, and foreign-region gradients were nonzero for the ramp and the arm with foreign updates only and zero for the other two; the auxiliary pass changed only foreign-region parameters; inclusion frequency and rescaling matched $p\,q$ and $1/q$; and distributed and serial gradients agreed within the tolerance of 10^{-5}, checked in full FP32 precision. All twelve runs were completed. Evaluation follows Appendix N: the largest composition penalty of the all-region member over the four domains on the development data, per trained model, with 0.10 nats as a descriptive threshold and no pooling over seeds. The arms with joint activity only and with foreign updates only also gave code members above the 2.310 nats of a model trained on code alone, and above the ramp models (Table 37).

The earlier version trained only the two new arms (protocol committed at 13:15 UTC; a first launch had stopped in the gradient check, before any training, because TF32 matrix products were enabled, and the check was then run in full FP32 precision with all training settings unchanged) and compared them with the ramp models of the schedule comparison (Appendix N) and the three models without exposure of Appendix T. Its two new arms gave the same evaluation losses as their counterparts here to all reported digits. Its reference ramp models gave largest penalties of 0.033 nats in each seed and its reference models without exposure 8.319 to 8.886 nats.

Table 37: Joint activity and foreign updates, 30M main mixture, development data, one row per trained model; all twelve models trained from one code version with shared initial state and data order within each seed. Largest composition penalty over the four domains, code penalty and loss of the code member, in nats.

arm	seed	largest penalty	code penalty	code member
ramp (joint activity and foreign updates)	0	0.035	0.029	2.248
	1	0.032	0.025	2.232
	2	0.034	0.032	2.273
joint activity only	0	0.285	0.207	2.512
	1	0.335	0.274	2.473
	2	0.333	0.260	2.530
foreign updates only	0	0.638	0.638	2.433
	1	0.546	0.546	2.426
	2	0.642	0.642	2.455
no exposure (neither)	0	7.940	7.940	2.159
	1	8.970	8.970	2.154
	2	8.405	8.405	2.197

Y ARITHMETIC AFTER PYTHON CONTEXT

This diagnostic was designed after the main results were known; its protocol and scripts were committed before any model was scored. Each of the 192 held-out equations was preceded by one of eight Python snippets fixed before scoring, 24 equations per snippet. The equations are the validation equations used in the main text, not a new test set. Table 38 lists exact answers of four members of each model without and with the Python context.

Table 38: Exact answers (of 192) without Python context and with it (native → prefixed), for four members of each model of the second mixture. One row per trained model. The prefix does not require the model to use code for the answer, so the table does not test a task that needs both skills.

size	exposure	seed	arithmetic	code	code and arithmetic	all regions
30M	ramp	0	192→0	0→0	192→0	192→0
30M	ramp	1	192→0	0→0	192→0	192→0
30M	none	0	192→136	0→0	172→66	5→2
124M	ramp	0	192→0	0→0	192→0	192→0
124M	ramp	1	192→1	0→0	192→0	192→0
124M	none	0	192→134	0→2	192→112	53→0

With the Python context, the member combining the code and arithmetic regions answered no more equations than the arithmetic member alone in any of the six models: 0 against 0 or 1 in the ramp models, and 66 against 136 at 30M and 112 against 134 at 124M without exposure (exact McNemar tests, $p = 1.6 \times 10^{-16}$ and $p = 9.4 \times 10^{-4}$, both in favour of the arithmetic member). In the ramp models, the prefixed outputs contained repeated punctuation or digits, empty answers and continued Python text; the failures are not evidence that arithmetic was erased. No prompt was changed after scoring.