

Damage184 LMR experiments and recommendation

A measured study of depth rank and neural policy guidance

4 October 2026

The recommended experimental setting is policy ordering with lmr mode 22. It retains classic LMR eligibility and adds reductions by policy rank and depth only away from the principal variation. It falls back to classic LMR in small endings and in nodes offering promotion.

Across three fresh depth 26 runs from the standard starting position, the selected scheme used 41.50 million nodes and 4.29 seconds on average. History with classic LMR used 208.75 million nodes and 16.22 seconds. The measured reductions were 80.1% in nodes and 73.6% in time.

The same build paired position suite used 61.7% fewer total nodes and 62.2% less total time. The best move matched the classic policy baseline in all 15 searched cases. Three forced move cases returned without a search and are excluded from speed aggregates.

These are search efficiency and consistency results. They do not establish an Elo gain or prove that every tactical line survives the extra reductions. The report separates measured facts, explanations supported by counters, and hypotheses that still require testing.

Contents

1 The objective and the decision.....	3
2 What remains unchanged	3
3 The measurement protocol	4
4 Why the first probability scheme failed	5
5 What a policy probability can tell the search	5
6 The numbered variants.....	6
7 The first screen at depth 22	7
8 The depth 26 shortlist.....	8
9 Position validation and its limits.....	9
10 Deeper and LMR disabled checks.....	10
11 Why guarded mode 22 was selected.....	11
12 The final rule in precise terms.....	12
13 Repeated final measurements	13
14 What the debug counters show.....	14
15 Build controls and harness corrections	15
16 Verification and what remains unproven	15
17 Proposed sequence of further tests.....	16
Appendix A Complete run catalogue.....	16
Appendix B Position FENs and provenance	23
Appendix C Files and reproducibility.....	26
Appendix D The chosen LMR source and its search integration	27
Appendix E The deployed policy network and fast inference.....	32

1 The objective and the decision

The objective was to find an LMR scheme that halves time or nodes, rather than merely improve a move ordering average. Two baselines matter: the established history ordering engine, and policy ordering with classic LMR. The initial probability experiment was much slower than either, so a useful solution had to recover that loss and produce a substantial improvement.

Mode 22 meets the numerical target on the starting position and in the weighted position suite. It is a conservative version of mode 12, the strongest simple depth and rank schedule found in the first screen. The additional guards deliberately surrender potential speed in small endings and near promotion. This narrows the change to positions where the quiet men policy is more useful as an ordering signal.

The recommended setting is explicit opt in. Classic LMR remains available and remains the default search mode. Modes 0 through 22 are retained so the author can reproduce the experiments, compare alternatives and run matches later. Mode 17 is a diagnostic reference, not a recommended playing setting.

Configuration	Mean nodes M	Mean seconds	Repeats
History classic	208.751	16.218	3
Policy classic	191.532	19.712	3
Policy mode 22	41.503	4.287	3

Depth 26 mean nodes over three fresh runs



Figure 1 Measured mean nodes from the standard starting position at depth 26. Each configuration has three fresh process runs.

2 What remains unchanged

The TT move is searched first. Policy inference stays delayed until that move fails to cut off. Neither the generated move indices nor capture encoding is changed. The network weights, value evaluator, opening position and other pruning routines are held fixed for the benchmark comparisons.

Classic LMR eligibility is checked before any policy adjustment. The current classic rule allows a reduction at depth two or greater after at least one searched move at a non PV node, or three searched moves at a PV node. Captures and the promoting move itself are excluded. A non PV node with at least four moves already searched receives two plies; other eligible moves receive one.

The root still uses classic LMR in playing variants. This is important: the large gain is from the internal tree, not from suppressing the main root alternatives. An internal reduced search that exceeds alpha is still searched again at full depth. That recovery rule is retained throughout the playing experiments.

The word classic describes the engine as it existed at the start of this experiment. It does not mean an exhaustive minimax proof. The engine already contains reductions, a separate reduced depth pruning probe, transposition bounds and capture quiescence. The no LMR diagnostic disables reductions at root and internal nodes, but those other mechanisms remain.

```
Classic internal eligibility
depth >= 2
searched moves >= (PV ? 3 : 1)
no capture and this move does not promote
```

```
Classic reduction
non PV and searched moves >= 4 : 2 plies
otherwise : 1 ply
```

3 The measurement protocol

Every timed benchmark starts a new engine process with a fresh transposition table. The configuration uses one worker, a 256 MiB TT, no opening book and disabled tablebases. Both value and policy weights come from retained local assets. Policy arithmetic is float32 except the explicitly labelled int16 run. No engine or weights were deployed to S during this study.

The starting position is W:W31-50:B1-20. The screen used depth 22 to compare a large family cheaply. Promising variants advanced to depth 26. The final recommendation received three repeats, interleaved with three history classic and three policy classic runs to reduce the influence of a changing CPU state.

The position suite mixes legally replayed random games, a measured opening continuation, capture choices and constructed rule guard positions. Positions are not claimed to be a representative statistical sample of tournament play. Both sides to move are included. King examples use depth 20; the other paired suite cases use depth 22. Each ratio compares a candidate with the same position and depth.

The engine reports time_ms around the search. Process initialization and weight loading are outside that metric. Wall time is retained separately in raw records. This matters for the intended match engine, which loads once and searches many moves. All timed searches enable debug so the cost of counters is present on both sides.

Item	Setting
Workers	1
TT	24 entry address bits, 16 bytes per entry, 256 MiB
Book and tablebases	Disabled
Value weights	Identical retained value-best.weights
Policy weights	Identical retained policy-best.weights
Arithmetic	Float32 unless marked int16
Timing	Engine search time_ms; fresh process per run
Deployment	None

4 Why the first probability scheme failed

The first rule removed one ply from top three moves or any move above ten percent probability, kept reductions in the middle band and added one below one percent. This seemed modest, but the classic reduction is often only one ply. Removing that ply converts a reduced search into a full depth search. Many favoured alternatives therefore become much more expensive.

The earlier depth 26 policy experiment used 845.70 million nodes and 99.24 seconds, compared with 191.53 million nodes and 18.94 seconds for policy plus classic LMR in that session. It decreased reductions about 79.77 million times and increased them only 4.12 million times. That imbalance was visible before trying more variants.

The depth 22 rerun made the mechanism more explicit. Mode 1 decreased reductions 7,834,492 times; 6,497,717 of those adjustments made the reduction zero. Mode 2 protected only the top move, but retained the above ten percent rule. It still zeroed reductions 5,293,366 times. Protecting fewer ranks alone was therefore insufficient.

A favourable cutoff rank cannot compensate for the cost of searching many alternatives at full depth. The successful family keeps existing reductions and only adds depth selectivity where the move rank, depth or probability justifies testing it. This was the central change in the experimental direction.

Depth 22 rule	Nodes M	Seconds	Reductions zeroed
Mode 0	26.324	2.861	0
Mode 1	72.349	9.454	6,497,717
Mode 2	59.692	7.887	5,293,366
Mode 3	21.336	2.354	0
Mode 4	11.785	1.215	0
Mode 12	8.092	0.834	0

5 What a policy probability can tell the search

A policy probability is a preference learned from its training target. It is not the probability that a move is safe to reduce, or the probability that it cannot refute the current alpha bound. Those events depend on the search window, depth and tactical continuation. Treating a small policy value as a proof would discard the recovery mechanism that makes LMR useful.

Absolute thresholds also depend on how many legal moves exist. Ten percent can be relatively large among twenty alternatives and relatively small among five. Uniform relative variants compare p times the number of legal moves with a threshold. Favourite relative variants compare a move probability with the largest probability at the node. These alternatives were tested rather than assumed superior.

Probabilities are computed only in variants that use policy information for LMR. Existing legal move logits are normalized with $\exp(\text{logit} - \text{maximum})$, divided by their sum. The already searched TT move is included in that denominator. Otherwise a remaining move would appear more likely simply because the TT was removed.

The output layer still evaluates legal moves only. Computing softmax adds exponentials and a sum, not another hidden layer pass and not all 81 geometric outputs. Mode 0 measures this normalization path while leaving the reduction unchanged. Mode 19 is a separate history or search order control and needs no probability inference.

6 The numbered variants

The following catalogue states the implemented rules. Except modes 1 and 2, playing variants leave shallow classic eligibility unchanged below depth four and never remove an existing reduction. Their final reduction is bounded by four plies and depth minus two. Mode 17 disables LMR globally for diagnostics. Modes 20 through 22 apply the mode 12 schedule only after their node guards pass.

Rank means neural policy rank over all legal moves including TT. Number means actual search order including TT. A PV node is detected by a search window wider than one score unit. It is not a guarantee that every move in the node is the final principal variation.

Mode	Purpose	Implemented rule
0	Normalization control	Compute legal softmax but use classic reductions.
1	Original rule	Top configured ranks or $p > \text{high}$: minus one; other $p < \text{low}$: plus one. Defaults top 3, high .10, low .01.
2	Top one ablation	Mode 1 with rank protection reduced to one. The high probability protection remains.
3	Low probability only	Add one if $p < \text{configurable low}$. Do not remove existing reductions.
4	Rank tail	Add one for policy rank greater than three.
5	Uniform relative	Add one beyond rank one when p times legal move count is below .5.
6	Two rank tiers	Add one beyond rank two; add a second beyond rank six at depth eight or greater. Includes PV nodes.
7	Absolute probability tiers	Add one beyond rank one below .05; a second beyond rank three below .005 at depth eight or greater.
8	Relative probability non PV	Outside PV, add one beyond rank one below .75 of uniform; a second beyond rank three below .10 of uniform at depth ten or greater.
9	Relative probability including PV	Mode 8 thresholds with PV adjustments allowed.
10	Clear favourite only	Add one if the best p is at least .35 and the move p is below .15 of the best, beyond rank one.
11	Order and depth control	Outside PV, add one at depth six and search number at least three; another at depth twelve and number at least six.
12	Rank and depth	Outside PV, add one at depth six beyond rank one; another at depth twelve beyond rank four.
13	Conservative relative gate	Outside PV, at depth six beyond rank two, add one below .5 of uniform.
14	Conservative rank gate	Outside PV, at depth six beyond rank three, add one.

Mode	Purpose	Implemented rule
15	Non PV rank tiers	Outside PV, add one at depth six beyond rank two; another at depth ten beyond rank five.
16	Favourite relative tiers	Outside PV, add one at depth six beyond rank one below .5 of best; another at depth ten beyond rank three below .05 of best.
17	LMR disabled reference	Disable root and internal LMR; search a sole root move for diagnostic scoring. Other pruning remains.
18	Probability gated mode 12	Mode 12 first extra ply requires $p < .75$ of best; second requires $p < .20$ of best.
19	History order control	Outside PV, add one at depth six beyond search number one; another at depth twelve beyond number four. Works with history and requires no policy.
20	Small ending guard	Mode 12, but use classic LMR when twelve or fewer pieces remain.
21	Promotion option guard	Mode 12, but use classic LMR for the entire node if any legal move promotes.
22	Combined guards	Mode 12 with both the twelve piece and promotion option guards. Recommended experimental setting.

7 The first screen at depth 22

The initial screen tested history classic, policy classic and modes 0 through 16. Every configuration returned a score of zero from the starting position. History chose 32-28; policy configurations chose 31-26. This position alone cannot determine which reduction scheme is strongest, but it is useful for rejecting schemes that greatly increase work.

Modes 1 and 2 failed the efficiency objective. Low probability only mode 3 improved modestly but did not halve either baseline. Rank tail mode 4 passed the node target. Modes 6, 11, 12, 15 and 16 were more aggressive while using depth, rank or relative probabilities differently.

Mode 12 gave the smallest node count in this screen. The fact that a simple rank and depth schedule performed well matters: the network already supplies useful order, while its probability calibration may not be the best predictor of reduction safety. More elaborate probability bands are available for further tests, but complexity was not a prerequisite for the measured gain.

Starting position at depth 22

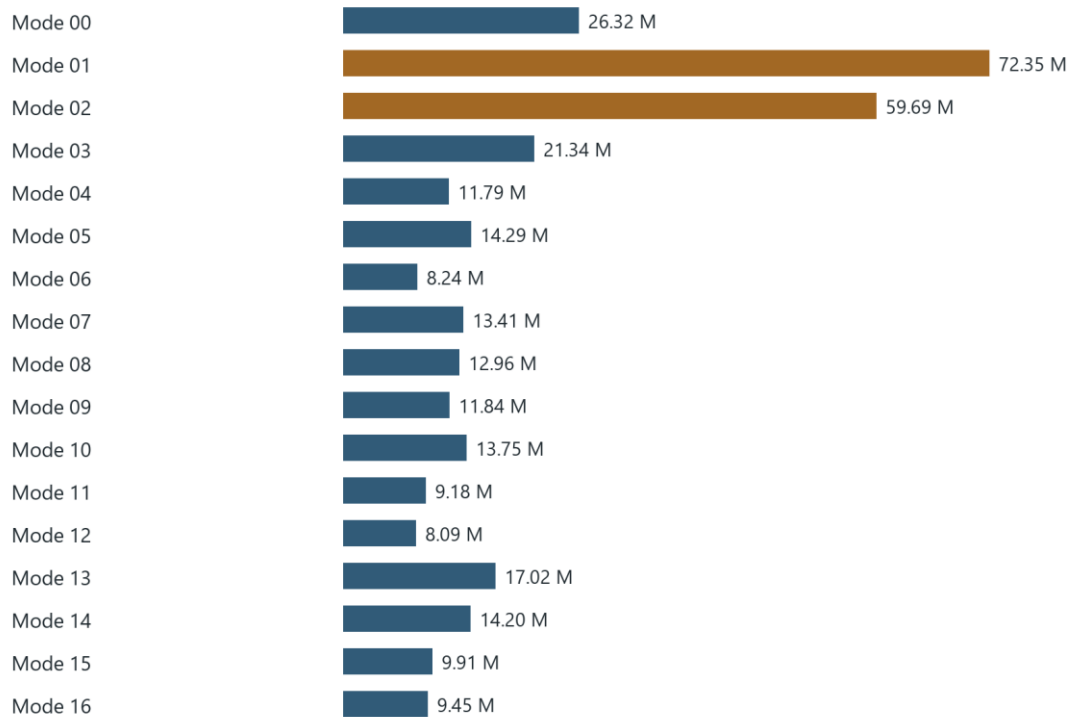


Figure 2 All initial policy mode screens at depth 22. Gold highlights the two rules that removed reductions. These are single runs, used for screening rather than precise timing estimates.

8 The depth 26 shortlist

Modes 4, 6, 8, 11, 12, 15 and 16 advanced to depth 26. Mode 12 used 41.50 million nodes and 3.98 seconds in the first shortlist run. Mode 6 was close in nodes but less conservative because it also adjusts eligible PV moves. Mode 4 was slower but still met the halving objective against the earlier history baseline.

The order and depth control, mode 11, also performed well. This shows that much of the speed comes from stronger depth selectivity, rather than a special interpretation of exact neural percentages. It does not make policy irrelevant: the network supplies the move order and rank used by the preferred scheme. It does mean that a history based stronger LMR schedule deserves its own controlled match comparison.

The final claim uses repeated current build baselines below. Single runs from earlier builds remain in the catalogue and help explain the decisions, but they are not pooled blindly with the final repetitions.

Mode	Nodes M	Seconds	Cutoff rank	Best move
4	74.504	7.585	1.5616	31-26
6	45.545	4.505	1.5300	31-26
8	81.249	8.052	1.5936	31-26
11	51.551	4.986	1.5796	31-26
12	41.503	3.981	1.5697	31-26

Mode	Nodes M	Seconds	Cutoff rank	Best move
15	53.519	5.392	1.5761	31-26
16	53.684	5.162	1.5876	31-26

9 Position validation and its limits

The final paired suite contains 18 positions: 15 require a search and 3 return a sole legal move immediately. The summed node reduction is 61.7% and the summed time reduction is 62.2%. The geometric mean node ratio is 0.4998, which gives equal multiplicative weight to each searched position.

Mode 22 matches the current build classic policy best move in all 15 searched positions. The largest score difference is 0.12 in the engine displayed units. That difference occurs without a changed best move. Agreement is encouraging but is not a proof of equal playing strength: both configurations can share an error.

The suite includes two root capture choices. The root captures themselves are not reduced; changes occur after those captures reach quiet men positions. A king on the current board disables policy LMR at that node, but later capture of the last king can reenter the policy domain. Therefore a mixed king root can legitimately have a different total tree even though no king position directly uses the policy rule.

The pure king guard position is intentionally small and tactically losing because the kings face one another on the long diagonal with the side to move trapped on its corner. Its role is to verify the unchanged fallback path, not to provide a representative king endgame timing. Small timings are dominated by fixed overhead and should not be interpreted as reliable speed ratios.

Position	D	Classic M	Mode 22 M	Node ratio	Same move
pv-ply10	22	58.080	15.195	0.262	Yes
pv-ply11	22	50.179	14.158	0.282	Yes
random1-ply20	22	34.966	11.451	0.327	Yes
random1-ply40	22	66.894	37.203	0.556	Yes
random2-ply20	22	55.584	14.213	0.256	Yes
random2-ply60	22	36.431	18.490	0.508	Yes
random3-ply40	22	28.394	12.816	0.451	Yes
random3-ply80	22	7.074	7.074	1.000	Yes
random2-ply80	22	0.331	0.331	1.000	Yes
capture-options-game1	22	23.214	8.425	0.363	Yes
capture-options-game5	22	70.209	18.419	0.262	Yes
king-options-game1	20	10.216	10.216	1.000	Yes

Position	D	Classic M	Mode 22 M	Node ratio	Same move
king-options-game4	20	2.574	1.041	0.405	Yes
pure-kings	20	0.000	0.000	1.000	Yes
promotion-options	22	1.904	1.904	1.000	Yes

Mode 22 nodes as a fraction of the same build baseline

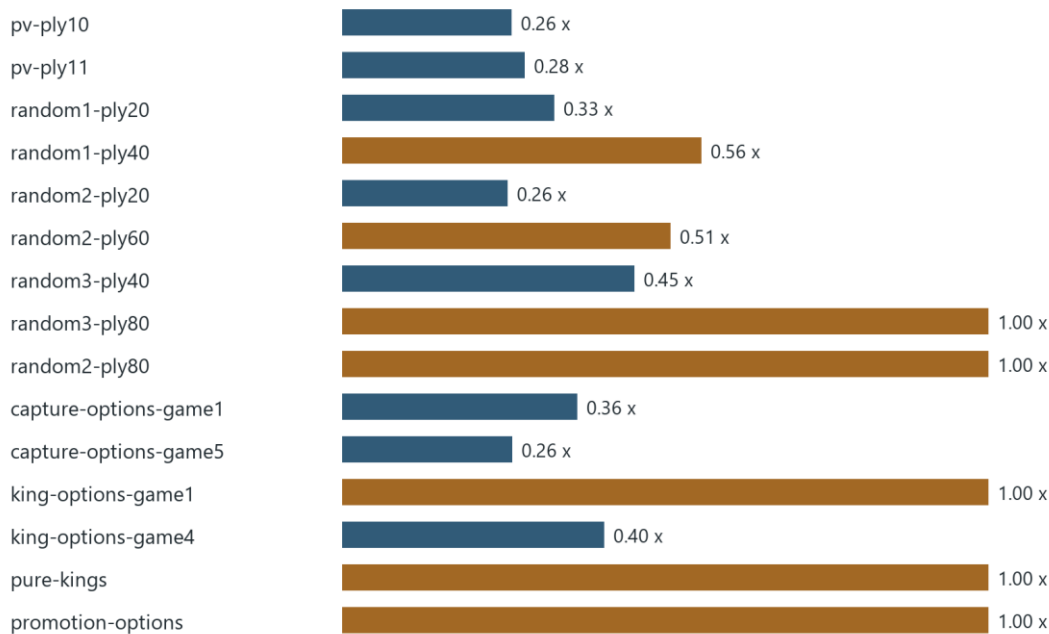


Figure 3 Paired node ratios from the final current build suite. A value of one means no node saving; one half means the target was reached for that position. Forced root returns are excluded.

10 Deeper and LMR disabled checks

Two late game positions received depth 26 checks and depth 18 searches with LMR disabled. The latter is a useful reference for the effect of reductions, but it still has the engine separate pruning and capture search. It must not be described as an exact full width minimax oracle.

In random2 ply60, the classic depth 26 score was -2.26 and modes 6 and 12 returned -2.24 with the same best move 37-31. The no LMR depth 18 score was -2.00. The changing score across horizons is a reminder that a deeper selective search and a shallower unreduced search answer related but different questions.

In random3 ply80, the classic depth 26 run preferred 39-33 with -1.60, while modes 6 and 12 preferred 46-41 with -1.46 and -1.32. The no LMR depth 18 root preferred 46-41 with -1.00. Neither configuration can be declared correct merely because it agrees with one reference at a different horizon. This case motivates the small ending guard and further independent tactical testing.

A common depth child comparison assessed 46-41 and 43-38 in that late man position. With LMR disabled at child depth 18, their parent perspective scores were -1.00 and -1.04. This slightly favours 46-41 in that particular diagnostic, but does not certify the whole position. Similarly, both 7-1 and 26-21 in the constructed promotion case returned zero in the child references. The promotion move change in an earlier candidate was therefore not demonstrated to be a blunder.

For the mixed king case, the common child reference slightly favoured 40-34 over 29-24 at one horizon, although the root no LMR run preferred 29-24 at its own horizon. The report retains these disagreements rather than forcing them into a simple accuracy percentage. A future validated tactical suite should supply independently established winning or drawing classes.

Position and mode	D	Nodes M	Seconds	Score	Move
random2-ply60-m00-d26	26	398.510	45.713	-2.26	37-31
random2-ply60-m06-d26	26	91.292	10.952	-2.24	37-31
random2-ply60-m12-d26	26	124.879	15.026	-2.24	37-31
random2-ply60-m17-d18	18	157.325	21.273	-2.0	37-31
random2-ply60-m00-d18	18	3.009	0.342	-1.84	37-31
random2-ply60-m06-d18	18	1.451	0.183	-1.82	37-31
random2-ply60-m12-d18	18	1.849	0.228	-1.82	37-31
random3-ply80-m00-d26	26	74.083	9.789	-1.6	39-33
random3-ply80-m06-d26	26	23.703	3.510	-1.46	46-41
random3-ply80-m12-d26	26	28.573	4.213	-1.32	46-41
random3-ply80-m17-d18	18	22.372	2.849	-1.0	46-41
random3-ply80-m00-d18	18	1.069	0.115	-0.88	39-33
random3-ply80-m06-d18	18	0.827	0.093	-1.0	46-41
random3-ply80-m12-d18	18	0.725	0.108	-0.82	39-33

11 Why guarded mode 22 was selected

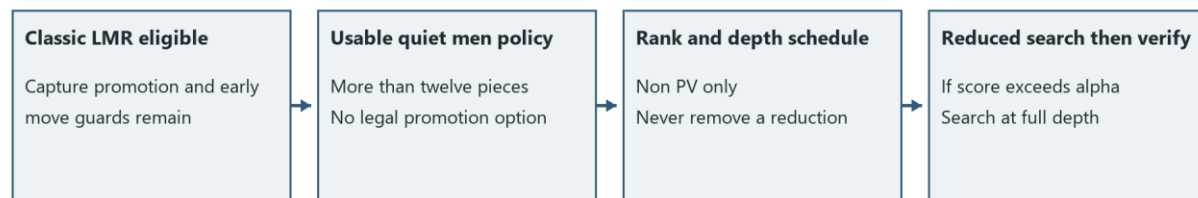
Mode 12 was already sufficient for the starting position target. Modes 20, 21 and 22 test whether inexpensive domain guards can retain that gain while limiting the changed search in sensitive cases. Their starting position node counts remain near 41.50 million, so the guards do not remove the main efficiency benefit there.

Mode 20 falls back at twelve or fewer pieces. Mode 21 falls back when any legal move in the node promotes, protecting the whole choice context rather than only the promoting move. Mode 22 combines them. Twelve is an initial conservative boundary chosen for this experiment, not an optimized universal endgame constant.

The fallback retains policy ordering. It changes the LMR rule back to classic and avoids the extra probability normalization path. It does not disable the neural network globally. This preserves the distinction between a model that ranks moves and a model allowed to control depth.

Mode 22 is selected because it meets the measured speed target, gives the same best moves as the current build paired baseline, retains PV reductions unchanged and restores the baseline in the small ending and promotion guard cases. Mode 6 remains a useful faster or more aggressive comparison. It should not replace the guarded recommendation solely on its single position screen time.

Mode 22 keeps classical eligibility before applying a depth schedule



TT remains first Root LMR stays classic Missing policy falls back to classic

Figure 4 The implemented mode 22 decision path. The guards precede the depth and rank adjustment; ordinary classic eligibility still controls which moves can enter it.

Recommended explicit configuration

```
movesort = policy
policy-arithmetic = float32
lmr = policy
lmr-mode = 22
```

Revert selectivity while keeping policy ordering

```
movesort = policy
lmr = classic
```

12 The final rule in precise terms

A move must first be eligible for classic LMR and must not be the TT move. The node must have usable policy results, quiet men only, more than twelve total pieces and no legal promotion option. If any condition fails, keep the classic reduction.

At an eligible non PV node, add one ply of reduction when remaining depth is at least six and policy rank is greater than one. Add another when depth is at least twelve and rank is greater than four. At a PV node, keep the classic reduction. Clamp the final reduction to at most four and at most depth minus two.

There is no subtraction from the classic reduction. A protected favourite still receives whatever classic LMR already prescribed; it is not automatically searched at full depth. This is the practical difference from the failed first scheme. Absolute probabilities and the original high and low parameters are not used by mode 22, although they remain available to the probability variants.

Remaining depth	Rank 1	Ranks 2 to 4	Rank 5 and later
Below 6	Classic	Classic	Classic
6 to 11	Classic	Classic plus 1	Classic plus 1
12 and above	Classic	Classic plus 1	Classic plus 2

The table describes eligible non PV moves after node guards. It is not a replacement for those guards and does not apply at the root. Depth is the engine remaining search depth at the current node, rather than the requested root depth.

```

if classic_reduction == 0 or move == TT:
    keep classic
else if policy_unavailable or kings or capture:
    keep classic
else if pieces <= 12 or any legal promotion:
    keep classic
else if PV:
    keep classic
else:
    R = classic_reduction
    if depth >= 6 and policy_rank > 1: R += 1
    if depth >= 12 and policy_rank > 4: R += 1
    R = min(R, 4, depth - 2)

search reduced depth
if reduced result > alpha:
    search again at full depth

```

13 Repeated final measurements

Three runs per final configuration were interleaved. Each run starts a new process and empty TT, rather than reusing a table warmed by an earlier configuration. The table reports means, ranges and observed node values. Timing variation is retained instead of presenting a single stopwatch reading as an exact property of the algorithm.

Mode 22 reduces mean nodes by 80.1% and time by 73.6% relative to history classic. Relative to policy classic, its reductions are 78.3% and 78.3%. Both comparisons comfortably meet the halving target on this position.

The isolated int16 run and history order control are recorded separately. They are useful engineering observations, not three repeat estimates. The principal recommendation uses float32 because that is the arithmetic used for screening and suite validation. Switching arithmetic can slightly change logits and consequently the searched tree.

Configuration	Mean s	Minimum s	Maximum s	Std dev s	Mean nodes M
History classic	16.218	15.822	16.823	0.532	208.751
Policy classic	19.712	18.307	21.017	1.357	191.532
Mode 22	4.287	4.265	4.324	0.033	41.503

Depth 26 mean elapsed time over three fresh runs

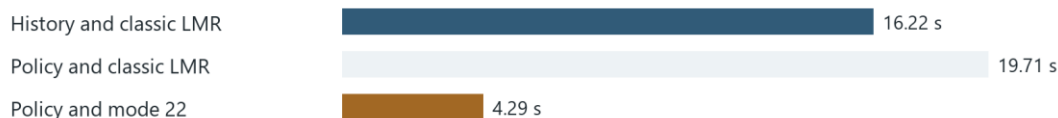


Figure 5 Repeated depth 26 elapsed times. The full run catalogue retains the individual measurements and executable hashes.

Control	Nodes M	Seconds	Score	Move
control-history-mode19-d26	38.208	3.818	0.0	32-28
control-policy-mode19-d26	37.381	3.783	0.0	31-26

Control	Nodes M	Seconds	Score	Move
arithmetic-int16-mode22-d26	41.872	4.369	0.0	31-26

The history order control separates the benefit of a stronger reduction schedule from the benefit of neural ordering. It has not received the same broad validation as mode 22. Its result is a reason to run a fuller history control series next, rather than attribute every saved node to the policy network.

14 What the debug counters show

The reduction counters cover internal search loops, including recursive verification searches, and exclude root reductions. The ordering rank statistics cover a narrower quiet men scope. Their denominators therefore differ. A reduction count should not be divided by eligible_nodes as if it were one event per ordering node.

lmr_less, lmr_same and lmr_more count the comparison with the classic reduction where usable variant information exists. lmr_fallback counts classic eligible internal moves without usable policy guidance or excluded by the new guards. lmr_zeroed identifies a reduction removed entirely. That last counter explained modes 1 and 2 directly.

lmr_reduced counts actual reduced internal searches. lmr_verify counts a reduced result above alpha that required a full depth search. lmr_verify_cut counts those full depth searches that then reached beta. A lower verification count is not automatically better: removing reductions also removes opportunities to count such verifications.

The existing cutoff rank is conditional on a non TT cutoff occurring. The no cutoff best rank describes completed nodes with no beta cutoff. Their populations change when LMR changes the tree. They help diagnose the search, but cannot be treated as fixed dataset classifier accuracy or a direct proof of improved ordering.

Debug field	History classic	Policy classic	Mode 22
nodes	208,750,728	191,531,809	41,502,882
eligible_nodes	20,936,633	18,408,714	3,949,571
policy_calls	0	14,005,285	2,947,512
non_tt_avg	1.5658	1.6366	1.5697
best_non_tt_avg	3.8792	3.5475	3.3456
lmr_less	0	0	0
lmr_same	0	0	7,012,283
lmr_more	0	0	852,371
lmr_fallback	0	0	34,630
lmr_zeroed	0	0	0
lmr_reduced	40,283,415	36,357,300	7,899,284
lmr_verify	1,186,919	815,865	161,014

Debug field	History classic	Policy classic	Mode 22
lmr_verify_cut	501,311	331,328	67,634

15 Build controls and harness corrections

One apparent small ending regression was caused by comparing results from different builds. The earlier classic run used 5,475,085 nodes and returned -0.94. The later build classic run used 10,215,800 and returned -1.34 in the same nominal case. Because guarded mode 22 made no LMR adjustments there, attributing that difference to its reduction rule would have been wrong.

Three pairs with the same current executable gave exactly 10,215,800 nodes, best move 29-24 and score -1.34 for both classic and mode 22. All final suite baselines were therefore rerun with that executable. The build sensitivity is recorded as an unresolved issue. Floating point ordering and selective search feedback are plausible mechanisms, but this experiment did not isolate the first diverging node and does not claim a proven cause.

Two harness corrections are retained in the record. A sole legal root move causes Damage184 to return without an info depth line. The harness originally assumed that every run would produce one. It now marks those cases forced-root and excludes them from speed aggregates while checking that the returned move remains consistent.

A diagnostic go analyze run did not retain the requested fixed depth. The HUB handler changes analysis to fixed time, so the attempted depth 26 reference was a 26 second setting and completed depth 25. It is retained as invalid-protocol and excluded from fixed depth comparisons. The corrected diagnostic uses go think, and mode 17 explicitly searches a sole root move for scoring.

16 Verification and what remains unproven

The Release x64 build succeeds. Focused tests cover variant reduction bounds, classic mode preservation, PV and rank protection, depth tiers, threshold boundaries, white and black legal probability mapping, TT inclusion in normalization and preservation of TT selection. Invalid LMR modes and thresholds are rejected during initialization.

The paired suite checks exhaustive capture, quiet men and quiet king node partitions against total nodes. Completed ordering nodes reconcile with cut and no cut nodes, and incomplete_nodes is zero. Verification cutoff counts never exceed full depth verification counts, which never exceed reduced searches. A tested int16 configuration completes with valid statistics.

These checks establish implementation consistency in the tested cases. They do not prove tactical completeness, correct bounds under every heuristic, or playing strength. In particular, the model was trained to prefer moves rather than predict whether an LMR reduction is safe at a specific alpha beta window. A move that fails low at reduced depth can still hide an improvement and never trigger a full depth recovery.

There is also selection bias: the variants were screened on the starting position and inspected on the available suite. The final suite includes validation cases that influenced the guards, so it is not a pristine holdout sample. A separate unseen suite and paired equal time games are the next required evidence before treating the setting as a stronger playing engine.

17 Proposed sequence of further tests

First preserve the current evidence and freeze mode 22, its weights and source hashes. Run an unseen position suite drawn from real games, keeping the original FEN and move notation. Stratify by material count, capture availability, imminent promotion, king presence and side to move. Report both summed cost and the distribution of per position ratios, rather than allowing a few expensive cases to hide regressions.

Second run paired equal time matches against history classic and policy classic, with both colours for each opening and repeated opening pairs. The search experiment establishes a resource saving, not an Elo result. A match should record wins, draws, losses, time forfeits, terminal adjudication rules and the exact engine configuration.

Third compare mode 22 with history order mode 19 over the same unseen suite and match openings. This measures whether the neural rank contributes beyond a stronger general depth schedule. Keep all other search parameters fixed. Only then decide whether a probability based extension adds enough value to justify extra inference and calibration work.

Fourth tune depth thresholds, rank thresholds, the twelve piece boundary and the four ply cap in small controlled batches. Avoid changing all knobs at once. Hold back a set of positions and openings during tuning so a faster setting is not simply fitted to the measured start position.

Fifth collect reduction safety labels. For sampled late moves, perform both reduced and full depth searches with the same window and record whether reduction hides a score above alpha or a beta cutoff. Features can include depth, PV status, rank, probability, score gaps, capture threats, promotion context and recent search history. Such a model directly addresses the scheduling decision more closely than a played move policy.

Sixth investigate the cross build sensitivity before using small evaluation or node differences as evidence. Record the first differing move selection or returned bound in paired instrumented runs. Compare full first layer rebuilds with incremental float32 accumulation and with int16 accumulation. Repeat with identical TT keys and history initialization. The goal is to distinguish acceptable numerical sensitivity from a state or initialization defect.

Finally consider a staged recovery scheme only after those tests. A reduced move that nearly reaches alpha could be searched at an intermediate depth before full depth. A risk sensitive version could protect capture threats and sacrifice preparations. These are proposals, not features implemented or performance gains claimed in this report.

Appendix A Complete run catalogue

The catalogue includes 202 retained engine run records from this research task. It includes completed searches, forced-root returns and the excluded analysis protocol attempt. The earlier three baseline runs from the preceding experiment are discussed in the text but are not counted as new runs here. Each catalogue ID maps to its exact label and raw result in `run_catalogue.json`. Complete debug lines, PVs and INIs are retained in `runs slash label`.

Nodes are expressed in millions; seconds are engine search time. The listed depth is the actual reported depth. A zero depth forced-root record is not a failed search: the engine deliberately returned the only legal move. The invalid analysis record is explicitly marked. None of those records contributes to the final speed target.

ID	Phase	Position	Mode	D	Nodes M	Seconds	Score
1	Screen	Start	Classic	22	24.630	1.993	0.0
2	Screen	Start	Classic	22	26.324	2.737	0.0
3	Screen	Start	0	22	26.324	2.861	0.0
4	Screen	Start	1	22	72.349	9.454	0.0
5	Screen	Start	2	22	59.692	7.887	0.0
6	Screen	Start	3	22	21.336	2.354	0.0
7	Screen	Start	4	22	11.785	1.215	0.0
8	Screen	Start	5	22	14.291	1.486	0.0
9	Screen	Start	6	22	8.238	0.889	0.0
10	Screen	Start	7	22	13.406	1.382	0.0
11	Screen	Start	8	22	12.957	1.337	0.0
12	Screen	Start	9	22	11.843	1.235	0.0
13	Screen	Start	10	22	13.746	1.397	0.0
14	Screen	Start	11	22	9.184	0.964	0.0
15	Screen	Start	12	22	8.092	0.834	0.0
16	Screen	Start	13	22	17.018	1.742	0.0
17	Screen	Start	14	22	14.197	1.462	0.0
18	Screen	Start	15	22	9.913	1.007	0.0
19	Screen	Start	16	22	9.451	0.982	0.0
20	Shortlist	Start	4	26	74.504	7.585	0.0
21	Shortlist	Start	6	26	45.545	4.505	0.0
22	Shortlist	Start	8	26	81.249	8.052	0.0
23	Shortlist	Start	11	26	51.551	4.986	0.0
24	Shortlist	Start	12	26	41.503	3.981	0.0
25	Shortlist	Start	15	26	53.519	5.392	0.0
26	Shortlist	Start	16	26	53.684	5.162	0.0
27	Suite	P01	Classic	22	58.076	5.875	0.0
28	Suite	P01	4	22	23.086	2.371	0.0
29	Suite	P01	6	22	12.948	1.337	0.0
30	Suite	P01	12	22	15.195	1.748	0.0

ID	Phase	Position	Mode	D	Nodes M	Seconds	Score
31	Suite	P01	16	22	20.112	2.111	0.0
32	Suite	P02	Classic	22	50.179	5.123	0.0
33	Suite	P02	4	22	20.139	2.073	0.0
34	Suite	P02	6	22	12.900	1.320	0.0
35	Suite	P02	12	22	14.158	1.492	0.0
36	Suite	P02	16	22	17.166	1.955	0.0
37	Suite	P03	Classic	22	34.966	3.621	-0.02
38	Suite	P03	4	22	13.354	1.417	-0.02
39	Suite	P03	6	22	7.836	0.851	0.0
40	Suite	P03	12	22	11.451	1.225	-0.02
41	Suite	P03	16	22	15.712	1.674	-0.02
42	Suite	P04	Classic	22	66.935	7.130	4.6
43	Suite	P04	4	22	22.546	2.588	4.6
44	Suite	P04	6	22	13.533	1.542	4.6
45	Suite	P04	12	22	37.198	4.094	4.6
46	Suite	P04	16	22	23.763	2.701	4.62
47	Suite	P05	Classic	22	55.579	5.945	-0.06
48	Suite	P05	4	22	22.781	2.452	-0.06
49	Suite	P05	6	22	13.709	1.489	-0.06
50	Suite	P05	12	22	14.213	1.540	-0.06
51	Suite	P05	16	22	18.865	2.113	-0.06
52	Suite	P06	Classic	22	36.344	3.949	-2.12
53	Suite	P06	4	22	26.041	3.001	-2.12
54	Suite	P06	6	22	14.032	1.540	-2.04
55	Suite	P06	12	22	18.394	1.990	-2.12
56	Suite	P06	16	22	29.511	3.129	-2.12
57	Suite	P07	Classic	22	28.370	3.034	-2.4
58	Suite	P07	4	22	14.267	1.569	-2.4
59	Suite	P07	6	22	12.399	1.380	-2.4
60	Suite	P07	12	22	13.661	1.526	-2.42

ID	Phase	Position	Mode	D	Nodes M	Seconds	Score
61	Suite	P07	16	22	16.778	1.839	-2.4
62	Suite	P08	Classic	22	7.074	0.908	-1.2
63	Suite	P08	4	22	4.724	0.625	-1.16
64	Suite	P08	6	22	2.860	0.371	-0.98
65	Suite	P08	12	22	3.532	0.464	-1.14
66	Suite	P08	16	22	5.200	0.624	-1.24
67	Suite	P09	Classic	0	0.000	0.000	Forced
68	Suite	P09	4	0	0.000	0.000	Forced
69	Suite	P09	6	0	0.000	0.000	Forced
70	Suite	P09	12	0	0.000	0.000	Forced
71	Suite	P09	16	0	0.000	0.000	Forced
72	Suite	P10	Classic	0	0.000	0.000	Forced
73	Suite	P10	4	0	0.000	0.000	Forced
74	Suite	P10	6	0	0.000	0.000	Forced
75	Suite	P10	12	0	0.000	0.000	Forced
76	Suite	P10	16	0	0.000	0.000	Forced
77	Suite	P11	Classic	22	0.331	0.028	-49.94
78	Suite	P11	4	22	0.320	0.032	-49.94
79	Suite	P11	6	22	0.304	0.028	-49.94
80	Suite	P11	12	22	0.318	0.032	-49.94
81	Suite	P11	16	22	0.319	0.027	-49.94
82	Suite	P12	Classic	0	0.000	0.000	Forced
83	Suite	P12	4	0	0.000	0.000	Forced
84	Suite	P12	6	0	0.000	0.000	Forced
85	Suite	P12	12	0	0.000	0.000	Forced
86	Suite	P12	16	0	0.000	0.000	Forced
87	Extra suite	P13	Classic	22	23.214	2.749	0.0
88	Extra suite	P13	4	22	7.436	0.816	0.0
89	Extra suite	P13	6	22	4.918	0.584	0.0
90	Extra suite	P13	12	22	8.425	1.028	0.02

ID	Phase	Position	Mode	D	Nodes M	Seconds	Score
91	Extra suite	P13	18	22	9.906	1.135	0.02
92	Extra suite	P14	Classic	22	70.185	8.602	4.58
93	Extra suite	P14	4	22	22.170	2.532	4.7
94	Extra suite	P14	6	22	17.777	2.107	4.6
95	Extra suite	P14	12	22	18.418	2.168	4.7
96	Extra suite	P14	18	22	24.368	2.815	4.7
97	Extra suite	P15	Classic	20	5.475	0.564	-0.94
98	Extra suite	P15	4	20	10.344	1.148	-1.3
99	Extra suite	P15	6	20	12.219	1.296	-1.34
100	Extra suite	P15	12	20	7.640	0.734	-1.1
101	Extra suite	P15	18	20	11.132	1.115	-1.34
102	Extra suite	P16	Classic	20	2.574	0.297	-3.56
103	Extra suite	P16	4	20	1.209	0.152	-3.58
104	Extra suite	P16	6	20	0.844	0.095	-3.58
105	Extra suite	P16	12	20	1.041	0.121	-3.58
106	Extra suite	P16	18	20	1.058	0.136	-3.58
107	Extra suite	P17	Classic	20	0.000	0.000	-49.98
108	Extra suite	P17	4	20	0.000	0.000	-49.98
109	Extra suite	P17	6	20	0.000	0.000	-49.98
110	Extra suite	P17	12	20	0.000	0.000	-49.98
111	Extra suite	P17	18	20	0.000	0.000	-49.98
112	Extra suite	P18	Classic	22	1.980	0.183	0.0
113	Extra suite	P18	4	22	1.949	0.219	0.0
114	Extra suite	P18	6	22	2.174	0.201	0.0
115	Extra suite	P18	12	22	2.822	0.283	0.01
116	Extra suite	P18	18	22	2.940	0.279	0.01
117	Reference	P06 invalid	Classic	25	270.606	30.513	-2.26
118	Quality	P06	Classic	26	398.510	45.713	-2.26
119	Quality	P06	6	26	91.292	10.952	-2.24
120	Quality	P06	12	26	124.879	15.026	-2.24

ID	Phase	Position	Mode	D	Nodes M	Seconds	Score
121	Quality	P06	17	18	157.325	21.273	-2.0
122	Quality	P06	Classic	18	3.009	0.342	-1.84
123	Quality	P06	6	18	1.451	0.183	-1.82
124	Quality	P06	12	18	1.849	0.228	-1.82
125	Quality	P08	Classic	26	74.083	9.789	-1.6
126	Quality	P08	6	26	23.703	3.510	-1.46
127	Quality	P08	12	26	28.573	4.213	-1.32
128	Quality	P08	17	18	22.372	2.849	-1.0
129	Quality	P08	Classic	18	1.069	0.115	-0.88
130	Quality	P08	6	18	0.827	0.093	-1.0
131	Quality	P08	12	18	0.725	0.108	-0.82
132	Child ref	P08 child	17	18	36.252	5.050	1.0
133	Child ref	P08 child	17	18	38.956	5.232	1.04
134	Reference	P15	17	18	24.739	2.449	-0.96
135	Child ref	P15 child	17	18	18.033	1.811	1.34
136	Child ref	P15 child	17	18	43.720	4.516	1.26
137	Reference	P18	17	18	10.478	1.006	0.0
138	Child ref	P18 child	17	18	32.873	2.688	0.0
139	Child ref	P18 child	17	18	8.306	0.678	0.0
140	Guard 22	Start	18	22	8.623	0.807	0.0
141	Guard 26	Start	18	26	45.929	4.583	0.0
142	Guard 22	Start	20	22	8.092	0.717	0.0
143	Guard 26	Start	20	26	41.503	4.106	0.0
144	Guard 22	Start	21	22	8.092	0.733	0.0
145	Guard 26	Start	21	26	41.503	3.983	0.0
146	Guard 22	Start	22	22	8.092	0.734	0.0
147	Guard 26	Start	22	26	41.503	4.025	0.0
148	Final suite	P01	22	22	15.195	1.544	0.0
149	Final suite	P02	22	22	14.158	1.420	0.0
150	Final suite	P03	22	22	11.451	1.238	-0.02

ID	Phase	Position	Mode	D	Nodes M	Seconds	Score
151	Final suite	P04	22	22	37.203	4.192	4.6
152	Final suite	P05	22	22	14.213	1.543	-0.06
153	Final suite	P06	22	22	18.490	1.988	-2.12
154	Final suite	P07	22	22	12.816	1.376	-2.42
155	Final suite	P08	22	22	7.074	0.797	-1.2
156	Final suite	P09	22	0	0.000	0.000	Forced
157	Final suite	P10	22	0	0.000	0.000	Forced
158	Final suite	P11	22	22	0.331	0.028	-49.94
159	Final suite	P12	22	0	0.000	0.000	Forced
160	Final suite	P13	22	22	8.425	0.878	0.02
161	Final suite	P14	22	22	18.419	1.972	4.7
162	Final suite	P15	22	20	10.216	0.945	-1.34
163	Final suite	P16	22	20	1.041	0.123	-3.58
164	Final suite	P17	22	20	0.000	0.000	-49.98
165	Final suite	P18	22	22	1.904	0.167	0.0
166	Same build	P15	Classic	20	10.216	1.086	-1.34
167	Same build	P15	22	20	10.216	0.980	-1.34
168	Same build	P15	Classic	20	10.216	0.976	-1.34
169	Same build	P15	22	20	10.216	1.045	-1.34
170	Same build	P15	Classic	20	10.216	1.051	-1.34
171	Same build	P15	22	20	10.216	0.974	-1.34
172	Final base	P01	Classic	22	58.080	6.170	0.0
173	Final base	P02	Classic	22	50.179	5.280	0.0
174	Final base	P03	Classic	22	34.966	3.631	-0.02
175	Final base	P04	Classic	22	66.894	7.414	4.6
176	Final base	P05	Classic	22	55.584	6.032	-0.06
177	Final base	P06	Classic	22	36.431	3.877	-2.12
178	Final base	P07	Classic	22	28.394	3.031	-2.4
179	Final base	P08	Classic	22	7.074	0.792	-1.2
180	Final base	P09	Classic	0	0.000	0.000	Forced

ID	Phase	Position	Mode	D	Nodes M	Seconds	Score
181	Final base	P10	Classic	0	0.000	0.000	Forced
182	Final base	P11	Classic	22	0.331	0.035	-49.94
183	Final base	P12	Classic	0	0.000	0.000	Forced
184	Final base	P13	Classic	22	23.214	2.599	0.0
185	Final base	P14	Classic	22	70.209	7.841	4.58
186	Final base	P15	Classic	20	10.216	0.963	-1.34
187	Final base	P16	Classic	20	2.574	0.295	-3.56
188	Final base	P17	Classic	20	0.000	0.000	-49.98
189	Final base	P18	Classic	22	1.904	0.168	0.0
190	Repeat	Start	Classic	26	208.751	16.010	0.0
191	Repeat	Start	Classic	26	191.532	21.017	0.0
192	Repeat	Start	22	26	41.503	4.265	0.0
193	Repeat	Start	22	26	41.503	4.324	0.0
194	Repeat	Start	Classic	26	208.751	16.823	0.0
195	Repeat	Start	Classic	26	191.532	19.811	0.0
196	Repeat	Start	Classic	26	191.532	18.307	0.0
197	Repeat	Start	22	26	41.503	4.270	0.0
198	Repeat	Start	Classic	26	208.751	15.822	0.0
199	Control	Start	19 H	26	38.208	3.818	0.0
200	Control	Start	19	26	37.381	3.783	0.0
201	Shortlist	Start	18	26	45.929	5.007	0.0
202	Int16	Start	22	26	41.872	4.369	0.0

Appendix B Position FENs and provenance

These FENs show the real board and actual side to move. Black positions are not rotated for display. The policy can use canonical internal coordinates without changing the notation shown to the reader. Random replay positions use seed 1842026 and the native engine legal generator, including compulsory maximum captures and promotion.

P01 pv ply10

W:W32,33,34,35,36,38,39,40,41,42,43,44,45,46,47,48,49,50:B1,2,3,4,5,6,8,9,10,11,12,13,15,16,18,20,21,23

classic opening line replay.

P02 pv ply11

B:W32,33,34,35,36,37,38,39,40,42,43,44,45,46,47,48,49,50:B1,2,3,4,5,6,8,9,10,11,12,13,15,16,18,20,21,23

classic opening line replay.

P03 random1 ply20

W:W27,31,32,35,36,37,39,42,43,44,45,46,47,48,49,50:B1,2,3,4,5,6,7,8,9,11,12,13,15,16,19,26

legal generator replay seed 1842026.

P04 random1 ply40

W:W31,33,34,35,36,37,40,42,45,46,47,48,49,50:B1,2,6,7,8,14,15,21,26

legal generator replay seed 1842026.

P05 random2 ply20

W:W27,35,37,38,39,40,41,42,44,45,46,47,48,49,50:B1,2,3,4,5,6,7,8,9,10,11,15,17,18,23

legal generator replay seed 1842026.

P06 random2 ply60

W:W26,29,30,35,37,43,44,45,46,47:B1,2,3,4,5,11,16,18,20,27,28,36

legal generator replay seed 1842026.

P07 random3 ply40

W:W31,32,34,38,40,41,42,44,45,47,49,50:B1,2,4,5,6,7,8,10,11,12,16,18,21,24,35

legal generator replay seed 1842026.

P08 random3 ply80

W:W23,39,43,46,49:B6,8,9,10,15,22

legal generator replay seed 1842026.

P09 random4 ply60

W:W25,26,33,34,36,39,40,41,42,44,46,48:B5,6,10,12,16,21,30

legal generator replay seed 1842026.

P10 random5 ply60

W:W25,26,27,28,33,36,42,43,45,46,48,49:B5,6,7,9,11,13,14,16,18,24,44

legal generator replay seed 1842026.

P11 random2 ply80

W:W15,26,31,46:B1,2,3,4,5,16,17,22,K27

legal generator replay seed 1842026.

P12 random4 ply80

W:W8,29,K32,33,34,36,38,40,41,43,44,46:B22,24,26,35

legal generator replay seed 1842026.

P13 capture options game1

B:W14,27,31,32,35,36,37,39,42,43,44,45,46,47,48,49,50:B1,2,3,4,5,6,7,8,9,10,11,12,13,15,16,26

legal generator replay seed 1842026.

P14 capture options game5

W:W21,27,28,34,35,36,38,39,41,42,43,44,45,46,47,48,49:B1,3,4,5,6,7,8,12,13,14,15,16,30

legal generator replay seed 1842026.

P15 king options game1

W:W29,40,42,45,46,47:B1,15,17,21,K50

legal generator replay seed 1842026.

P16 king options game4

B:WK3,26,28,31,35,36,37,38,39,41,42,43,46,48,49,50:B1,4,5,6,7,9,10,11,13,14,15,17,21

legal generator replay seed 1842026.

P17 pure kings

W:WK46:BK5

Constructed rule guard position.

P18 promotion options

W:W7,26:B20,45

Constructed rule guard position.

Child references use the native encoded move generator to apply the selected legal root move. Their resulting HUB positions are retained in child_positions.json and the individual run records. This avoids guessing which pieces a multiple capture removed from endpoint notation alone.

Appendix C Files and reproducibility

Damage184 is the canonical engine source for this change and has no Git repository metadata in its directory. The original source and executable were retained before editing. Source snapshots, patches, build information, hashes and all measurement records are retained locally. The Damage2026 application itself was not modified or deployed.

The engine identity, executable name and INI name remain Damage 18.4, damage184.exe and damage184.ini. User visible changes are recorded in the Damage184 change log. Classic LMR remains the default; mode 22 is selected by the explicit configuration in this report.

The final records distinguish compiler build changes. The per run executable SHA256 is retained. Value and policy hashes are listed below and in manifest.json. Reproduction requires the same weights, architecture, arithmetic, TT configuration, worker count, position, search depth, debug setting and relevant engine binary. Matching only the requested root depth is insufficient.

File or directory	Purpose
damage184/lmr.hpp	Numbered reduction functions
damage184/search.cpp	Eligibility, node guards, delayed policy, verification and counters
damage184/engine.cpp and engine.hpp	INI defaults, parsing, validation and engine options
damage184/search.hpp	Per worker debug counters
tests/policy_lmr_variants_test.cpp	Focused reduction properties and variant boundary tests
Damage184_LMR_research/runs	INI, transcript and JSON result for every run
analysis.json and run_catalogue.json	Paired suite analysis, repeat summary and complete catalogue
builds and source	Retained binaries and source snapshots
experiments scripts	Benchmark runner, suite construction and diagnostic checks

value-best.weights SHA256

89e874084a525edaaf2d000b64a1242ba643ae043c3b1d6f32ad469c0767ff56

policy-best.weights SHA256

30edb0c6aa79e13cb6d93c171e546244810f351983a474dff9610be42c3062e5

Primary sources for this report are the local Damage184 source, the retained INIs and raw HUB transcripts, and the structured result records. No external engine result or unmeasured playing strength estimate is used as evidence for the recommendation.

Appendix D The chosen LMR source and its search integration

This appendix adds the actual implementation behind the selected mode 22. The original measurements and recommendation remain unchanged. The source excerpts were read from the canonical Damage184 directory on 4 October 2026, after deployment. Ellipses are not inserted into the displayed excerpts; each quoted block is a contiguous section of its named file. Separate blocks show the caller, the shared reduction helper and the verification search.

Canonical source directory: C:\Users\bertt\source\repos\damage184\damage184. The deployed executable is S:\damage2026\engines\damage184.exe. Its SHA256 matches the final research executable. The deployment changed the INI selection, not the measured mode 22 algorithm.

D1 Start with the original reduction

reducedepth first decides whether this move is eligible at all. At a non-PV node, one move must already have been searched; at a PV node, three must already have been searched. The current move must not promote, the move list must not be a capture list, and remaining depth must be at least two. An eligible non-PV move after four searched moves receives two plies; the other eligible cases receive one. A move with zero classic reduction cannot acquire a reduction from mode 22.

Here bpv denotes the search window classification, not knowledge that the node will ultimately belong to the final principal variation. imove is the original generated move index. ixmovesearched counts completed move searches at this node, including TT when it was searched.

The classic eligibility function. Exact excerpt: search.cpp, lines 84 to 102.

```
inline int reducedepth(int imove, bool bpv, int idepth, movelist_t* movelist)
{
    // Diagnostic mode 17 disables LMR at root and internal nodes, but keeps
    // the engine's separate pruning rules and capture quiescence unchanged.
    if (enginedata.lmr == "policy" && enginedata.lmr_mode == 17) return 0;
    int ireducedepth = 0;

    if (idepth >= 2 && movelist->ixmovesearched >= (bpv ? 3 : 1)
        && movelist->ixcapture == 0 && movelist->qpromotionmove(imove) == false)
    {
        if (bpv == false && movelist->ixmovesearched >= 4)
            ireducedepth = 2;
        else
            ireducedepth = 1;
    }

    return ireducedepth;
}
```

D2 Decide whether policy may guide LMR at this node

The next block is in searchx, the internal recursive search. bcustom_order requires multiple legal moves, no capture and no king on either side. bpolicy_lmr additionally requires lmr=policy and a successfully enabled policy network. Mode 22 applies two further whole-node guards: at most twelve total pieces, or any legal move that promotes, turns off the policy LMR adjustment.

Turning off bpolicy_lmr does not turn off policy ordering. In an otherwise eligible quiet men position, the remaining legal moves can still be ordered by their logits, but reductions return to the classic schedule. This distinction matters when reading lmr_fallback: it is a reduction fallback, not necessarily a neural ordering fallback.

The ordering domain and material and promotion guards. Exact excerpt: search.cpp, lines 950 to 965.

```
// History remains unchanged for captures and positions with kings.
bool bcustom_order = enginedata.movesort != "history" && movelist.xcapture() == 0
    && node->position.m_bbking == 0 && ixmoves > 1;
bool border_ready = !bcustom_order;
bool bpolicy_lmr = enginedata.lmr == "policy" && enginedata.bpolicy && bcustom_order &&
enginedata.lmr_mode != 17 && enginedata.lmr_mode != 19;
// Guarded variants keep classical selectivity in small endings or
// when this node offers promotion. Policy ordering itself remains active.
if (bpolicy_lmr && (enginedata.lmr_mode == 20 || enginedata.lmr_mode == 22)
    && node->position.countallpiece() <= 12) bpolicy_lmr = false;
if (bpolicy_lmr && (enginedata.lmr_mode == 21 || enginedata.lmr_mode == 22))
    for (int i = 0; i < movelist.list(); i++)
        if (movelist.qpromotionmove(i)) { bpolicy_lmr = false; break; }
bool bpolicy_ready = false;
double policy_best = 0;
std::array<double, xmovelist> policy_probability;
std::array<int, xmovelist> policy_rank;
```

D3 TT first and delayed inference

The search tries a legal TT move before running the policy. movelist.setmove marks that generated move as searched, preserving its index. A TT cutoff ends the loop without the network being called. If TT fails to cut, the remaining-order path computes policy information once and sets border_ready. An excluded singular-verification TT move is not tried first.

The cached first layer belongs to the search worker and ply. The supplied parent is iply-2, a same-side-to-move ancestor. Two plies give the same orientation without eagerly maintaining both orientations after every move. The accumulator can rebuild when that ancestor is absent or invalid.

The TT first path and the eventual policy call. Exact excerpt: search.cpp, lines 976 to 1005.

```
while ((ialpha < ibeta) && (ixmoves-- > 0))
{
    int imove;
    if (bcustom_order && !border_ready && iordering_tt != tt_read_nomove)
    {
        // Try TT first. A cutoff avoids either policy inference or shuffling.
        imove = iordering_tt;
        movelist.setmove(imove);
        iordering_tt = tt_read_nomove;
    }
    else
    {
        if (!border_ready)
        {
            if (enginedata.movesort == "random")
                movesort::random_order(&movelist, sd->move_random, iskip_move);
            else
            {
                // Same-turn ancestor keeps its own sums; opposite-turn calls stay lazy.
                bool bevaluated = policy::policy_order(&node->position, &movelist, iskip_move,
                    &sd->policy_accumulator[iply], iply >= 2 ? &sd->policy_accumulator[iply - 2]
: nullptr,
                    bpolicy_lmr ? policy_probability.data() : nullptr, bpolicy_lmr ?
policy_rank.data() : nullptr);
                bpolicy_ready = bevaluated && bpolicy_lmr;
                if (bpolicy_ready)
                    for (int i = 0; i < movelist.list(); i++) policy_best =
std::max(policy_best, policy_probability[i]);
                if (bmeasure && bevaluated) ++sd->cutstats.policy_calls;
            }
            border_ready = true;
        }
        imove = select_move(&movelist);
    }
}
```

D4 The selected rank and depth rule

Mode 22 shares the two depth tiers with mode 12. Modes 20 and 21 share that helper too; their difference is the caller guards shown above. Therefore copying only the mode 22 case would lose the crucial material and promotion protection. The following helper fragments must be read together with D2 and D5.

The helper first preserves classic reduction for invalid rank or non-finite probability, and leaves later modes unchanged below depth four. Mode 22 then adds one ply at depth six or greater beyond policy rank one, and another at depth twelve or greater beyond rank four. Both additions require a non-PV node. The final clamp allows no more than four plies and never more than depth minus two.

The common helper entry and early exits. Exact excerpt: lmr.hpp, lines 19 to 29.

```
inline int variant_reduction(int mode, int classic, int depth, int rank, double probability,
    int number, int legal, bool pv, double best, int top, double high, double low)
{
    if (classic <= 0 || rank <= 0 || !std::isfinite(probability)) return classic;
    if (mode == 0) return classic; // Normalize probabilities but do not change LMR.
    if (mode == 1) return policy_reduction(classic, depth, rank, probability, top, high, low);
    if (mode == 2) return policy_reduction(classic, depth, rank, probability, 1, high, low);
    if (depth < 4) return classic; // Later variants do not alter shallow eligibility.
    int reduction = classic;
    switch (mode)
    {
```

The shared mode 22 branch. Exact excerpt: lmr.hpp, lines 62 to 68.

```
case 12: // Rank protected depth schedule; no reduction of the favourite beyond classic.
case 20: // Mode 12 with <=12-piece fallback applied by the caller.
case 21: // Mode 12 with promotion-option fallback applied by the caller.
case 22: // Mode 12 with both caller guards.
    if (!pv && depth >= 6 && rank > 1) ++reduction;
    if (!pv && depth >= 12 && rank > 4) ++reduction;
    break;
```

The final reduction cap. Exact excerpt: lmr.hpp, lines 92 to 92.

```
return std::max(0, std::min(reduction, std::min(4, depth - 2)));
```

The source still computes legal probabilities through the general LMR interface, and the helper checks that the supplied probability is finite. Mode 22 itself does not use a percentage threshold, probability gap or best probability to choose its extra reduction. Its decision uses policy rank and remaining depth. Removing unnecessary normalization specifically for rank-only modes is a possible future speed test; it was not part of the measured version.

Classic reduction	Remaining depth	Policy rank	Mode 22 reduction
1	8	1	1
1	8	2	2
2	14	3	3
2	14	5	4
2	14, PV node	5	2

These examples assume all ordinary move and node guards pass. The rank is among all legal policy outputs, including an already searched TT move. Thus the first move after a TT move can be rank one

if the policy preferred it, or rank two or later if TT ranked above it. Policy rank and actual search move number are deliberately separate quantities.

D5 Apply the helper only to classic eligible non TT moves

The internal reduction call. Exact excerpt: search.cpp, lines 1020 to 1029.

```
int iextenddepth = (movelist.list() == 1 ? 1 : 0);
int ireduceddepth = reduceddepth(imove, bpv, idepth, &movelist);
if (ireduceddepth && imove != itt_move && enginedata.lmr == "policy")
{
    int iclassic = ireduceddepth;
    bool blmr_ready = bpolicy_ready || (enginedata.lmr_mode == 19 && movelist.xcapture() == 0 &&
node->position.m_bbking == 0);
    if (blmr_ready)
        ireduceddepth = lmr::variant_reduction(enginedata.lmr_mode, iclassic, idepth,
            bpolicy_ready ? policy_rank[imove] : movelist.ixmovessearched + 1, bpolicy_ready ?
policy_probability[imove] : 0.0, movelist.ixmovessearched + 1,
            movelist.list(), bpv, policy_best, enginedata.lmr_top, enginedata.lmr_high,
enginedata.lmr_low);
```

The condition `ireduceddepth && imove != itt_move` protects classic-ineligible moves and the TT move. For mode 22, `blmr_ready` depends on `bpolicy_ready`. The mode 19 expression is a separate history-order experimental control; it does not make mode 22 work without policy. The probability and rank arrays are indexed by the original generated move index, not by an array that has been physically reordered.

After the call, debug counters classify the proposed reduction as less, same, more or fallback. A forced-move extension can subsequently zero a reduction. Consequently an adjustment counter is not a count of completed reduced searches. `lmr_reduced` is the counter for actual reduced probes.

D6 Reduced probe and full depth recovery

The child depth and verification search. Exact excerpt: search.cpp, lines 1056 to 1074.

```
int inextdepth = idepth + iextenddepth - 1;
sd->node[iply + 1].pvclear();

if ((bpv == true && movelist.ixmovessearched != 0) || ireduceddepth != 0)
{
    iscore = -searchx(sd, istid, iply + 1, inextdepth - ireduceddepth, -ialpha - 1, -ialpha,
move_none, bprune);

    if (sd->bcutstats && ireduceddepth) ++sd->cutstats.lmr_reduced;
    if (iscore > ialpha)
    {
        if (sd->bcutstats && ireduceddepth) ++sd->cutstats.lmr_verify;
        iscore = -searchx(sd, istid, iply + 1, inextdepth, -ibeta, -ialpha, move_none, bprune);
        if (sd->bcutstats && ireduceddepth && iscore >= ibeta) ++sd->cutstats.lmr_verify_cut;
    }
}
else
    iscore = -searchx(sd, istid, iply + 1, inextdepth, -ibeta, -ialpha, move_none, bprune);
```

`inextdepth` is remaining depth minus the played move, plus any extension. The first call subtracts `ireduceddepth` and uses the null window around alpha. Negating the child score converts it to the parent perspective. When that reduced result exceeds alpha, the second call restores `inextdepth` and uses the normal alpha-beta window. Full depth here means the intended depth for this child, including extensions, rather than the root requested depth.

A move can still be wrongly rejected by the shallow probe: if it fails low only because of the reduced horizon, it does not trigger verification. That is the central playing-strength risk of stronger LMR, even though the successful probes are verified. Root moves continue through the existing root search and classic reduction function; the mode 22 helper is applied only inside searchx.

D7 When mode 22 makes no extra reduction

There are two different outcomes: some conditions prevent policy ordering altogether, while others allow ordering but keep classic LMR. Captures, positions containing kings and nodes with one legal move lie outside the custom quiet men ordering domain. A TT cutoff skips inference entirely. A TT move itself does not receive the policy adjustment. Missing or failed policy initialization leaves the existing history/classic path available.

With policy ordering enabled, mode 22 keeps classic reductions at twelve or fewer pieces, whenever any legal move offers promotion, at PV nodes, and below remaining depth six. Rank one keeps classic reduction at every depth. Ranks two through four get at most the first extra ply; later ranks can receive the second only at depth twelve or greater. A classic-ineligible move, including the early moves, captures and the promoting move itself, is never made eligible by the helper.

lmr=classic bypasses the policy reduction adjustment. Selecting another lmr-mode chooses another experimental rule. The deployed choice is lmr=policy, lmr-mode=22 with movesort=policy and float32 arithmetic. Invalid mode values and malformed thresholds are rejected at initialization. Although mode 22 does not use the thresholds, the shared configuration validation remains active.

D8 Deployment configuration detail

```
movesort = policy
policy = true
policy-file = nn\policy-best.weights
policy-arithmetic = float32
lmr = policy
lmr-mode = 22
```

The deployed INI is S:\damage2026\engines\damage184.ini. Relative policy and value weight paths resolve from the engine process working directory. Start the engine with S:\damage2026\engines as that directory, or use absolute weight paths in a copied configuration. The deployed tablebase path remains S:\databases\.

The current ini_load parser accepts whitespace separated name = value entries and does not recognize comment lines. The deployment startup check caught this restriction and the deployed INI contains no comments. Adding a # comment can make loading fail before ini_update, causing defaults to be used. Keep explanatory notes in POLICY.md and this report.

Appendix E The deployed policy network and fast inference

This architecture is read from the actual deployed policy-best.weights, rather than inferred from the list of other neural files in the nn folder. Its header is D184POLICY 1 90 64 81. Version 1 fixes a 90-input layer, one 64-neuron ReLU hidden layer and an 81-output softmax head. The implementation computes the final linear logits and performs softmax only over the required legal move set.

File: S:\damage2026\engines\nn\policy-best.weights. SHA256:

30edb0c6aa79e13cb6d93c171e546244810f351983a474dff9610be42c3062e5. This is the same policy file used in the retained experiments. Other qnn and qtf files in that directory describe other networks and are not loaded by the deployed policy-file setting.

E1 Inputs and outputs

Inputs are binary occupancy features for men only. The first 45 features describe the side to move on canonical squares 6 through 50; the other 45 describe its opponent on canonical squares 1 through 45. Promotion-rank squares are omitted because a legal man promotes there. Empty squares contribute zero. There are no king features, capture features, separate side-to-move feature or value score input.

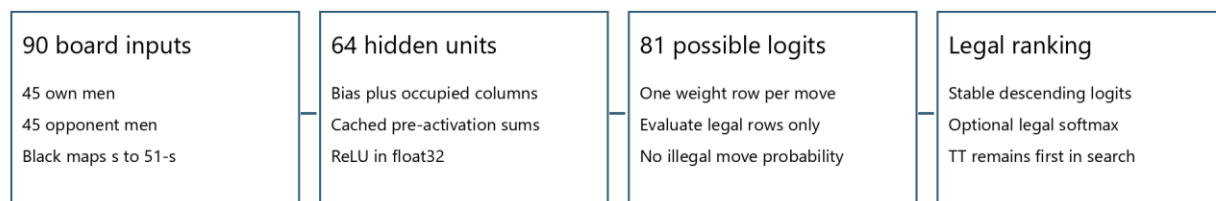
For White to move, canonical and real squares coincide. For Black, square *s* maps to 51-*s* and the colour roles swap. The engine scans the original set bits and uses `feature_index[turn][side][bit]` to map each occupied square directly to its canonical input. It does not rotate all 64 bit positions or construct a reversed bitboard first. A black man on real square 5 becomes the own-man canonical square 46 feature; a white man on real square 46 becomes the opponent canonical square 5 feature.

The 81 outputs enumerate geometric white single-step diagonal moves by ascending source square, then destination. Index 72 represents canonical 46-41. A real Black 5-10 move maps to that same output after 51-*s* is applied to both endpoints. Outputs include promotion steps geometrically, even though a promotion option makes mode 22 use classic reductions at that node. Legal move generation, including compulsory captures, remains the engine task.

Layer	Dimensions	Activation	Parameters
Board input	90 binary features	None	0
Hidden	90 to 64	ReLU	5,760 weights + 64 biases
Output	64 to 81	Linear logits, legal softmax when needed	5,184 weights + 81 biases
Total	90 to 64 to 81	One hidden layer	11,089

All 11,089 parameters need 44,356 bytes if stored as float32, about 43.3 KiB before padding, containers and retained alternate representations. The text weight file is larger. Hidden weights are fully connected mathematically, but their binary inputs let the engine replace a dense 90 by 64 multiply with sums of weight columns for occupied features. Biases are included once. The cached hidden sums are pre-activation values; ReLU is applied to a scratch copy before evaluating the output.

E2 Architecture diagram



Mode 22 uses rank and depth. Probabilities are still computed by its shared LMR interface.

Figure 6 The deployed 90 to 64 to 81 policy network and its legal move head.

E3 Initialization and data layout

`policy_init` reads and validates the header, dimensions and finite parameters. `policy_fast::init` builds float32 arrays once. First-layer weights are transposed so each input feature points to a contiguous vector of hidden weights; strides are padded to multiples of eight for AVX2. Later layers retain output-neuron rows for dot products. The version 1 hidden width 64 already fills eight vectors of eight floats.

The engine retains the loaded reference model and builds the optional quantized first layer as well. `policy_arithmetic` chooses which accumulator path search uses. The deployed selection is float32; choosing int16 does not change the network topology or require another policy file. If first-layer quantization is unsafe, initialization reports a float32 fallback. The output dot products remain float32 in the int16 path.

Generation identifiers invalidate cached sums when a new network is loaded. The architecture code also accepts version 2 self-describing files with more hidden layers and supported activations. That flexibility does not mean the deployed file has extra layers: the inspected version 1 file has exactly one hidden layer.

E4 Cached first layer and incremental updates

`policy_accumulator` is an array owned by each search worker, indexed by ply. It is separate from `position_t`, avoiding enlargement of every copied board structure. Each entry contains float and optional integer sums for both orientations, piece snapshots, generation identifiers and refresh counts. Storage is populated lazily when inference is actually required.

An exact position and generation match returns the existing pre-activation sums immediately. Otherwise a valid same-turn ancestor supplies the starting sums. XOR of current and ancestor occupancy identifies the changed squares. A newly occupied feature adds its 64 weights; a removed feature subtracts them. Captures and other intervening moves can change several squares, and the bitboard difference handles that set without assuming exactly one ordinary move.

For example, after White 46-41 and a Black reply, the next White-oriented node can compare against the White-oriented ancestor two plies above. It subtracts the own-square-46 column, adds the own-square-41 column and applies the changed Black columns as opponent features. If that ancestor was skipped because its TT move cut off, or its generation is stale, a bias-based rebuild is used.

Float accumulation is approximate. The parent refresh count must be below 16 for another incremental update; after that the sums rebuild from biases and occupied features. Large weights that require finite checks also disable the incremental path. The integer accumulator uses exact bounded integer sums and does not need that periodic refresh.

The actual float32 accumulator implementation. Exact excerpt: policy.cpp, lines 126 to 168.

```
static const float* policy_accumulate(position_t* position, accumulator_t& cache, const
accumulator_t* parent)
{
    int turn = position->bturn();
    const auto& first = fast_network.layers.front();
    auto valid = [&](const accumulator_t* p) {
        return p && p->generation[turn] == network_generation && p->values[turn].size() ==
first.stride;
    };
    if (valid(&cache) && cache.pieces[turn][0] == position->m_bbpiece[0]
        && cache.pieces[turn][1] == position->m_bbpiece[1]) return cache.values[turn].data();
    // Rebuild periodically: float sums are approximate, unlike int16 NNUE.
    bool incremental = valid(parent) && parent->updates[turn] < 16 && !first.check_finite;
    auto& sums = cache.values[turn];
    sums.resize(first.stride);
    if (incremental) std::copy_n(parent->values[turn].data(), first.stride, sums.data());
    else std::copy(first.bias.begin(), first.bias.end(), sums.begin());
    for (int side = 0; side < 2; side++)
    {
        bitboard_t current = position->m_bbpiece[side];
        bitboard_t changed = incremental ? current ^ parent->pieces[turn][side] : current;
        while (changed)
        {
            unsigned long bit;
            BITSCANFORWARD64(&bit, changed);
            changed &= changed - 1;
            int input = feature_index[turn][side][bit];
            // Captures/promotions may leave an invalid promotion-rank feature
            // in a stale ancestor snapshot; such a feature was never accumulated.
            if (input < 0) continue;
            const float* column = first.weight.data() + size_t(input) * first.stride;
            bool added = (current & (bitboard_t(1) << bit)) != 0;
            for (int i = 0; i < first.stride; i += 8)
            {
                __m256 value = _mm256_loadu_ps(sums.data() + i), weight = _mm256_loadu_ps(column
+ i);
                _mm256_storeu_ps(sums.data() + i, added ? _mm256_add_ps(value, weight) :
                _mm256_sub_ps(value, weight));
            }
            cache.pieces[turn][side] = current;
        }
        cache.generation[turn] = network_generation;
        cache.updates[turn] = incremental ? parent->updates[turn] + 1 : 0;
        return sums.data();
    }
}
```

BITSCANFORWARD64 finds one changed set bit, and changed &= changed - 1 removes it. feature_index maps that bit and colour to its orientation-dependent input. The AVX2 loop adds or subtracts eight hidden components at a time. Invalid omitted-rank features from a stale ancestor are skipped because they were not part of its accumulated feature set.

E5 Legal move outputs and stable ordering

policy_order enumerates generated legal moves, maps their endpoints to canonical coordinates and collects only the output indices needed. In classic policy ordering it excludes already searched moves and the singularly excluded move. If only one choice remains, inference is skipped. When the shared

policy LMR path requests probabilities, it includes every legal output, including the previously searched TT move, to keep ranks and normalization consistent.

The hidden layer is computed once for the position. The final head runs a 64-element dot product only for each requested move row. A position with nine legal quiet moves needs nine output dot products, not 81. With a TT move already searched and no probability request it can need eight. The generic double fallback for extreme non-finite float results can compute the full reference logits; this is an exceptional fallback, not the normal fast path.

The requested output rows in the fast inference head. Exact excerpt: `policy_fast.hpp`, lines 164 to 171.

```
const auto& output = network.layers.back();
for (int n = 0; n < count; n++)
{
    int i = outputs[n];
    scores[n] = dot(output.weight.data() + size_t(i) * output.stride, previous, output.stride,
output.bias[i]);
    if (output.check_finite && !std::isfinite(scores[n])) return false;
}
return true;
```

The AVX2 dot routine multiplies eight components together, accumulates the vectors, horizontally sums the result and adds the output bias. Version 1 has no intervening later hidden layer: cached first sums, ReLU, requested output dot products is its complete forward pass.

Descending logits determine the same order as descending softmax probabilities. `stable_sort` preserves generator order on equal logits. The engine assigns descending ordinal `iscore` values to the original generated move entries rather than physically rearranging the move array. That preserves TT move identifiers. `select_move` subsequently chooses the best remaining unsearched score; a cutoff can stop before the tail is visited.

Stable legal ranking and optional softmax. Exact excerpt: `policy.cpp`, lines 286 to 311.

```
// Softmax is monotonic: search needs only logits, preserving stable ties
// and original generated move indices used by the TT.
std::stable_sort(order.begin(), order.begin() + count,
    [&scores](int a, int b) { return scores[a] > scores[b]; });
// Normalize all legal logits, including the already searched TT move.
// Classic ordering retains remaining-move-only output evaluation.
if (probabilities && count)
{
    double maximum = scores[order[0]], sum = 0;
    for (int i = 0; i < count; i++)
    {
        probabilities[order[i]] = std::exp(scores[order[i]] - maximum);
        sum += probabilities[order[i]];
    }
    if (!(sum > 0) || !std::isfinite(sum)) return false;
    for (int i = 0; i < count; i++)
    {
        probabilities[order[i]] /= sum;
        ranks[order[i]] = i + 1;
    }
}
for (int i = 0; i < count; i++)
    if (!movelist->qsetmove(order[i])) movelist->iscore[order[i]] = count - i;
if (iskip_move != move_none) movelist->iscore[iskip_move] = -score_inf;
return true;
}
```

Subtracting the maximum logit before exponentiation prevents unnecessarily large exponentials. The denominator is the sum over legal outputs, not all 81 geometric outputs. The returned ranks begin at one, and both ranks and probabilities are stored at `order[i]`, the original generated move index.

Already searched moves can affect the probability denominator and rank without being selected again.

The policy is an ordering and selective-depth aid. It does not replace the value network, legal move generation, alpha-beta scores or terminal detection. Its percentages reflect the trained preference among legal moves; they are not win probabilities or a certified probability that a reduced move is safe.

E6 Practical optimization opportunities

The current implementation already gains from TT-first delayed inference, sparse occupied-feature sums, direct Black feature mapping, cached same-turn sums, AVX2 float32 and legal-only output rows. Mode 22 still pays for legal softmax because it shares the probability interface with percentage-based modes. A separate rank-only request could avoid the exponentials and sum while keeping the same descending logits and rank, provided initialization, singular exclusion and tie behavior remain consistent.

The next useful comparison is guarded rank-only mode 22 against the measured version, followed by a guarded history/search-order control under the same conditions. Increasing hidden width or depth would add inference cost and should be tested with ordering quality and equal-time playing strength, not assumed to improve this LMR scheme. Training directly for reduction safety is a different target from predicting a played move.

These appendices document the deployed implementation and potential experiments only. No source code, weights or deployed files were changed while adding them to the report.