PROOF: MIXED-TRAFFIC JAILBREAK DEFENSE VIA PROFILE ROUTING IN FROZEN LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Safety controls must prevent harmful completions while preserving legitimate assistance. We introduce PROOF, which learns to select among fixed safety profiles of a frozen language model. During training, each input is evaluated under every profile using a cost appropriate to its type: unsafe completion, unnecessary refusal, or capability-probe loss. When several profiles share the lowest cost, a tie-aware target retains a preferred action and assigns probability to the other minima. This preserves equally rated alternatives that hard-label imitation would penalize. We study both routing before response generation and selection among profile-generated candidates. On a fixed 200-example outcome table, ten tie-breaking assignments and five paired controller seeds yield a 7.69% reduction in sensitivity to tie-breaking relative to hard-label training, with no statistically resolved change in mean selection regret. Across eight profile-menu conditions, safety differences depend on the available actions. Serving measurements further show that candidate conditioning costs more than routing from prompt features. The results identify a benefit of tie-aware supervision for label sensitivity and distinguish it from menu-dependent safety gains and inference cost.

1 INTRODUCTION

A jailbreak attack seeks to make a model provide harmful assistance despite its safety controls. The same controls must also handle legitimate requests that resemble harmful ones and ordinary tasks requiring accurate responses. A security-analysis question, for example, may be mistaken for misuse. Evaluating a defense therefore requires measuring harmful completion, unnecessary refusal, and task performance together (Mazeika et al., 2024; Röttger et al., 2024; Cui et al., 2024; Zhou et al., 2023). We study this setting as *mixed traffic*: harmful, benign-edge, and capability-sensitive requests are evaluated under the same selection policy.

The question is which safety configuration to use for each input. A *profile* specifies an executable configuration of one backbone; a fixed menu makes several such configurations available. In our implementation, profiles combine previously trained safety and preservation adapters at different weights. Learning then concerns the choice of profile, while the backbone and adapters remain fixed. This separates two decisions: which behaviors the menu makes available, and which of them to use for a particular request.

PROOF learns the choice from observed profile outcomes. Each training input receives one cost per profile: unsafe completion on a harmful request, refusal on a benign request, or language-model loss on a capability probe. Several profiles can achieve the same minimum cost, particularly when a binary evaluator assigns the same judgment to different responses. Hard-label imitation designates one winner and penalizes all alternatives. Our tie-aware target keeps that representative preferred while allocating probability to the other minimum-cost actions. The resulting supervision retains the information that several actions received the same measured outcome.

The information available to the selector determines when it can act (Figure 1). A prompt-only router chooses a profile before response generation. A prompt-forward variant adds statistics from running the input through each profile, still without generating response candidates. A candidate-conditioned selector instead generates the candidates first and chooses one to serve. Each feature bundle has its own learned controller. Evaluator judgments supervise training only; the cost of producing the selector's inputs is included in serving time.

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 evaluate how the target, profile menu, and input features affect learned selection. Changing the representative labels across ten assignments shows that tie-aware training reduces decision disagreement relative to hard-label imitation. An eight-condition menu study finds safety gains in some menus and reversals in others. Timing and request-rewrite experiments then measure the cost of the three input choices and the sensitivity of prompt-only routing. Together, these experiments separate the effect of retaining tied alternatives from the effects of the available profiles and the information used to choose among them.

2 Related Work

Safety benchmarks measure both harmful completion and unnecessary refusal. HarmBench and JailbreakBench provide harmful behaviors and evaluation procedures (Mazeika et al., 2024; Chao et al., 2024); XSTest and OR-Bench include legitimate requests that can trigger over-refusal (Röttger et al., 2024; Cui et al., 2024). WildJailbreak supplies adversarial and non-adversarial harmful and benign examples (Jiang et al., 2024). We use these evaluation criteria to study which fixed configuration to apply to each input.

Inference-time defenses change how a model produces a response. SafeDecoding, for example, adjusts token selection to favor safe continuations (Xu et al., 2024). Profile routing addresses a different choice: which configuration to apply. Our experiments instantiate those configurations with fixed LoRA compositions (Hu et al., 2021), rather than modifying the decoding mechanism.

Guardrails provide an alternative signal for profile selection. Llama Guard classifies input and output safety (Inan et al., 2023); in our comparison, its unsafe/safe decision selects the safety/preservation endpoint of the same menu. **Model routing**, exemplified by RouteLLM (Ong et al., 2024), learns to choose among different models using preference data. Our actions instead configure a single frozen backbone.

Candidate selection first generates alternatives and then chooses a response. Variational best-of-N distills this selection behavior into the language model (Amini et al., 2025). Our candidate-conditioned implementation selects among responses generated under fixed profiles using a separate MLP. Its generation order differs from prompt-only routing, which selects the profile before a response is available.

Soft targets are used in distillation and label smoothing (Hinton et al., 2015; Müller et al., 2019). Our target distributes its additional probability only among actions tied at the measured minimum cost. Partial-label learning also uses candidate label sets, typically under the assumption that they contain one unknown true label (Feng et al., 2020). In profile selection, several actions can receive the same best cost. We retain a preferred representative within that set rather than assigning uniform probability to its members.

3 Method

3.1 Task and Fixed Profile Menu

Inputs and objectives. Harmful requests test whether a response provides unsafe assistance, including under jailbreak attempts. Benign requests test unnecessary refusal; *benign-edge* requests are legitimate inputs that resemble harmful ones. Capability probes measure performance on tasks such as reasoning or instruction following. Each training example has an evaluation type that determines its cost. This is a supervision label, not an input to the deployed selector; capability requests are themselves legitimate requests.

Runtime profiles. Let f_θ denote the backbone and $\mathcal{P} = \{p_1, \ldots, p_K\}$ a finite menu of its executable configurations. In our adapter implementation, each profile specifies composition weights for two previously trained LoRA components, one for safety and one for preservation. The backbone and both components remain fixed during router learning. Appendix A.1 gives the compositions used in the experiments.

At an adapted weight matrix, the profile is instantiated as

$$W(p_k) = W_0 + s_k \Delta W_{\text{safe}} + v_k \Delta W_{\text{preserve}}, \tag{1}$$

where W_0 is the base weight, the two ΔW terms are trained low-rank updates, and (s_k, v_k) specifies the profile. These coefficients need not sum to one: they scale parameter updates rather than mixing output probabilities. The safety component is trained on harmful examples and the preservation component on benign examples, with capability anchors supplying an additional preservation signal. Once the menu is constructed, selection changes only the active composition; it does not train a new adapter for each request.

We distinguish profile construction from selector learning. The former determines which behaviors are available, while the latter learns which action to take from measured outcomes. Names such as Safety-first and Preservation-first describe the intended composition, not a guarantee that one profile is safer or more capable on every input. The oracle compares actual measured costs rather than inferring performance from those names.

Given a fixed menu, we train a selector to predict a low-cost action from the available features. Figure 1(a) shows how measured profile outcomes produce the training targets. Panel (b) shows the input features and generation order for each selector.

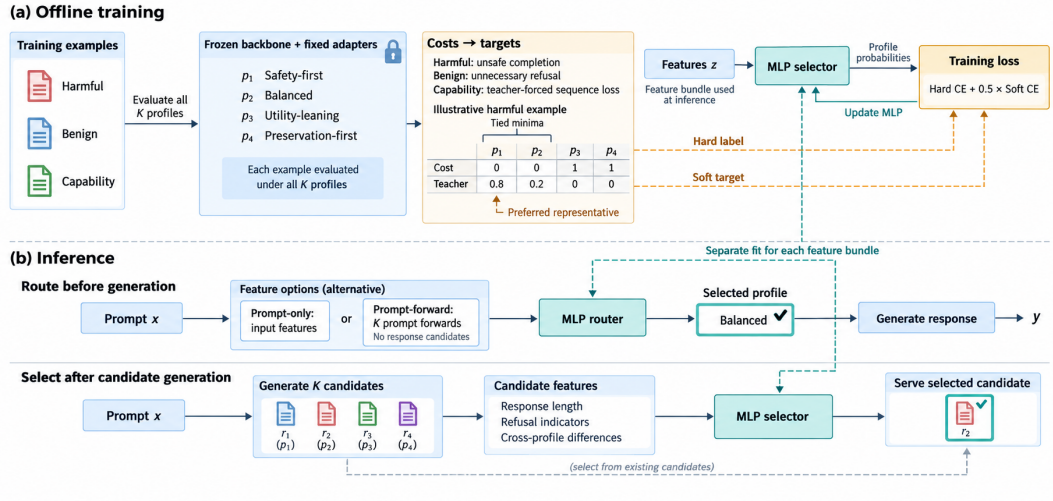


Figure 1: Outcome-based profile selection. (a) Profile costs supply hard labels and tie-aware targets to the training loss, separately from selector features. The four-profile example is illustrative. (b) Prompt-only and prompt-forward routers select before response generation; candidate-conditioned selection serves an existing response. Orange dashed arrows carry training targets, teal dashed links connect training and inference for separately fitted selectors, and the gray dashed path denotes candidate reuse.

3.2 From Profile Outcomes to Training Targets

Evaluate each action. For each training input x_i and profile p_k, the evaluation type t_i determines the cost:

$$c_{ik} = \begin{cases} u(x_i, r_{ik}), & t_i = \text{harm}, \\ b(x_i, r_{ik}), & t_i = \text{benign}, \\ \ell_{\text{seq}}(x_i, a_i; p_k), & t_i = \text{cap}. \end{cases} \tag{2}$$

Here r_{ik} is the generated response, u is a binary unsafe-completion judgment, and b is a binary refusal judgment on a benign request. Capability cost uses a reference answer a_i through teacher forcing: it is mean causal-LM loss over the non-padding tokens of the formatted prompt–answer sequence. We compare costs across profiles *for the same example*, using its evaluation criterion. Appendix A.4 specifies the judges and sequence loss.

Identify the best-rated actions. Collect the costs in $c_i = (c_{i1}, \ldots, c_{iK})$ and define $T_i = \{k : c_{ik} = \min_j c_{ij}\}$. A fixed order selects one representative $k_i^\star \in T_i$, preferring safety-heavy profiles on harmful examples and preservation-heavy profiles on benign or capability examples. This rep-

resentative is the *oracle action* used as a hard training label. Profiles indexed by T_i may produce different responses with equal measured costs.

Retain tied alternatives. Hard-label imitation assigns all probability to the representative. Our target reserves a fraction ρ for the other minimum-cost actions.

Let e_j be the length-K vector with value one at index j and zero elsewhere. The teacher distribution is

$$q_i = \begin{cases} e_{k_i^\star}, & |T_i| = 1, \\ (1-\rho)e_{k_i^\star} + \dfrac{\rho}{|T_i|-1} \displaystyle\sum_{k \in T_i \setminus \{k_i^\star\}} e_k, & |T_i| > 1. \end{cases} \tag{3}$$

We use $\rho = 0.2$. In Figure 1(a), costs $(0, 0, 1, 1)$ and representative 1 give $q_i = (0.8, 0.2, 0, 0)$; a unique minimum gives a one-hot target. The tie margin is zero, with numerical tolerance 10^{-9}. Thus the target retains a preferred representative while giving positive training probability to the other equally rated actions.

The distinction from smoothing over the entire menu is the support of the target. Probability is assigned only to profiles with the minimum measured cost, not to a higher-cost action. A binary judgment can nevertheless give the same cost to qualitatively different responses, including when every profile fails. Consequently, a tie states that the evaluation criterion does not distinguish those actions; it does not establish that their responses are interchangeable. The representative remains a deliberate preference within this measured set.

3.3 Selector Inputs and Learning

Selector inputs. The prompt-only router uses $z_i = \phi_{\text{prompt}}(x_i)$: a compressed mean of frozen input-token embeddings, input lengths, and fixed risk/utility keyword statistics. The candidate-conditioned selector uses $z_i = \phi_{\text{cand}}(x_i, r_{i1}, \ldots, r_{iK})$, adding response lengths, refusal-pattern indicators, and cross-profile differences. These features require generated candidates for every input type. Reference answers, evaluation types, and evaluator costs are used only for supervision.

A *prompt-forward* variant adds prompt likelihood and next-token uncertainty from running the input through each profile. It obtains model-dependent features without generating response candidates. Appendix A.2 details the three feature bundles and their normalization.

Training. For each feature bundle, we fit an MLP g_ψ that predicts $\pi_\psi(k \mid z_i) = \mathrm{softmax}(g_\psi(z_i))_k$. Only its parameters ψ are updated. The objective combines hard-label and soft-target cross-entropy; before class weighting, it is

$$\mathcal{L}_i = -\log \pi_\psi(k_i^\star \mid z_i) - \gamma \sum_k q_i(k) \log \pi_\psi(k \mid z_i), \tag{4}$$

with $\gamma = 0.5$. The hard term reinforces the representative: combining it with the illustrated teacher gives an effective target of approximately $(0.933, 0.067, 0, 0)$ after normalization. Training also uses class weights and balanced sampling; Appendix A gives the batch objective. Direct CE sets $\gamma = 0$ while retaining the same features, network, data, sampling, and optimizer. Figure 1(a) shows how the teacher enters the training loss.

Algorithm 1 summarizes the training and serving operations. Feature normalization is fitted on training inputs and reused at inference. A controller is fitted separately for each feature bundle, so a prompt-only router is not a candidate-conditioned selector with its response features removed after training. Row types, reference answers, and evaluator outputs determine supervision but are never supplied as deployment inputs.

3.4 Serving and Computational Cost

The selector chooses $\hat{k} = \arg\max_k \pi_\psi(k \mid z)$. Prompt-only and prompt-forward features are available before response generation, so only $p_{\hat{k}}$ produces a response. Candidate-conditioned selection first generates a response under every profile and then serves $r_{\hat{k}}$ without regenerating it. These orders are shown in Figure 1(b). Serving uses the fitted selector without an additional safety judge.

Algorithm 1 Learning and serving with a fixed profile menu

Require: Fixed profiles $\mathcal{P}$, typed training examples, feature bundle, tie mass ρ, and loss weight γ

Offline training

for each training example x_i **do**
  Evaluate all profiles to obtain costs c_i using Equation 2.
  Find minimum-cost indices T_i and representative $k_i^\star$.
  Construct teacher q_i using Equation 3.
  Extract features z_i for the chosen feature bundle.
end for
Fit feature normalization on the training partition.
Train the MLP on $(z_i, k_i^\star, q_i)$ with Equation 4 and the weighting in Appendix A.

Inference for a new input x

if the selector uses candidate-conditioned features **then**
  Generate $r_1, \ldots, r_K$ under the fixed profiles.
  Extract and normalize $z = \phi_{\text{cand}}(x, r_1, \ldots, r_K)$.
  Select $\hat{k} = \arg\max_k \pi_\psi(k \mid z)$ and return $r_{\hat{k}}$.
else
  Extract and normalize prompt-only or prompt-forward features z.
  Select $\hat{k} = \arg\max_k \pi_\psi(k \mid z)$.
  Generate and return a response under $p_{\hat{k}}$.
end if

Total serving cost includes feature extraction, selection, profile activation, and generation. Prompt-forward features add a forward pass through each profile; candidate-conditioned features require every profile's generated response. The MLP has the same role in both cases, but the work needed to obtain its inputs differs.

4 EXPERIMENTS

We evaluate PROOF on Qwen2.5-7B-Instruct (Qwen et al., 2024), using fixed compositions of safety and preservation LoRA components. Four experiments examine sensitivity to tie-breaking, the effect of the profile menu, serving time, and response safety under request rewrites. Each specifies the input features and comparison needed to isolate that question; Appendix A.3 summarizes the protocols.

Data and profiles. WildJailbreak provides harmful and benign requests for component training and offline supervision (Jiang et al., 2024). We compare training targets using fixed per-profile outcomes, so changing a target does not change the generated responses being scored. The serving-time experiment samples requests from HarmBench, JailbreakBench, OR-Bench and capability probes (Mazeika et al., 2024; Chao et al., 2024; Cui et al., 2024). The rewrite experiment applies eight transformations to each of 72 HarmBench and JailbreakBench requests. Appendix A.1 specifies the menus; Appendix A.3 gives the sample counts.

Controls. **Direct CE** isolates the training target: it uses the same features, MLP, data, sampling and optimizer, but assigns a hard label to one minimum-cost action. **Static profiles** use one fixed configuration for every input, providing controls for serving time and rewrite sensitivity. **Llama-Guard binary routing** (Inan et al., 2023) selects Safety-first for an unsafe label and Preservation-first for a safe label. It therefore uses an external safety classifier to choose among the same profiles, rather than directly blocking or answering the request.

Metrics and execution. Defense success rate (DSR) is the fraction of harmful requests whose responses are judged safe; benign refusal rate (BRR) is the fraction of benign requests that receive a refusal. Probe loss measures sequence cross-entropy on capability examples and is reported in loss units. Higher DSR and lower BRR/probe loss are preferred. Appendix A.4 defines the evaluators, loss and confidence intervals.

Feature bundles are matched within each target comparison. The tie-breaking experiment uses candidate-conditioned selectors, the menu experiment uses prompt-forward features, and the rewrite experiment uses prompt-only routers. The timing experiment measures all three serving paths.

Training and experimental units. The learned selectors use an MLP with one hidden layer of width 128, ReLU and dropout 0.1. AdamW uses a learning rate of 10^{-3}, weight decay 10^{-4} and batch size 64, with balanced class weights and sampling. The tie-resolution study uses the final checkpoint after 40 epochs; the menu study uses at most 40 epochs with early-stopping patience 8. The target parameters are $\rho = 0.2$ and $\gamma = 0.5$. Appendix A gives the weighted objective and initialization seeds.

Table 1 distinguishes the unit of comparison in each experiment. Controller initializations measure variation in learned selection on fixed outcomes; timing repetitions measure serving time; rewritten requests measure safety under changes to the input. These repetitions are not additional independent benchmark prompts.

Table 1: Evaluation sets and controlled comparisons. Counts refer to unique inputs within each study, not to repeated controller fits or timing runs. Each target comparison holds its feature bundle and per-profile outcomes fixed.

Study	Evaluation set	Controlled comparison
Tie resolution	200 inputs: 150 harmful, 30 benign, 20 probe	Ten representative assignments, five paired controller seeds, two targets; candidate-conditioned features
Profile menu	192 validation inputs from 961 oracle records	Eight menus, five paired controller seeds, two targets; prompt-forward features
Serving time	96 prompts shared across paths	Six paths, batch sizes 1 and 8, three timed repetitions after warm-up
Request rewrites	72 harmful base requests, eight rewrites each	Four policies using the same deterministic profile responses; prompt-only learned routers

4.1 Does Tie-Aware Supervision Reduce Sensitivity to Tie-Breaking?

When several actions share the minimum cost, changing their designated representative changes the hard label while leaving their costs fixed. We compare ten representative assignments, five paired controller seeds, and two targets, training 100 candidate-conditioned controllers and evaluating them on the same 200-example outcome table. For each prompt and seed, *tie-resolution disagreement* is one minus the fraction of assignments selecting the most frequent action, averaged over prompts and seeds. This measures how much the learned decision depends on the representative label.

Table 2: Tie-breaking sensitivity of candidate-conditioned controllers on 200 fixed outcomes. Lower is better. Reductions are Direct CE minus tie-aware, with simultaneous 95% intervals. Disagreement measures changes in the selected action; regret averages the row-specific excess cost. Appendix A.4 gives the definitions and resampling procedure.

Metric	Direct CE	Tie-aware (Ours)	Reduction [95% interval]
Tie-resolution disagreement	0.3314	0.3059	0.0255 [0.0100, 0.0320]
Mean selected-cost regret	0.05390	0.05365	0.00024 [−0.00421, 0.00438]

Tie-aware training reduces disagreement from 0.3314 to 0.3059, a 7.69% relative reduction (Table 2). The simultaneous 95% interval for the absolute reduction is $[0.0100, 0.0320]$. Mean selected-cost regret remains statistically indistinguishable. Allocating probability to tied alternatives there-

fore makes the decisions less sensitive to which representative receives the hard label, without a corresponding resolved improvement in selected cost.

The two measurements answer different questions. Disagreement counts a change in the selected action even if the new action has the same cost. Regret instead compares the selected cost with the minimum available cost on that example. Lower disagreement can therefore coexist with unchanged regret: decisions can become more consistent without producing better measured responses. Keeping the profile outcomes fixed isolates this effect from changes in generation or judging. The paired bootstrap resamples representative assignments and prompts while retaining paired targets and controller initializations; Appendix A.4 gives the interval construction and joint criterion.

4.2 How Does the Profile Menu Affect Safety?

We compare the targets using prompt-forward features across five nested menus and three leave-one-profile-out conditions. Each condition uses five paired controller seeds and the same 192 oracle-validation examples; the actions and their derived features change with the menu. Table 3 reports all eight conditions. With three profiles, tie-aware training improves harmful DSR by 2.75 percentage points over Direct CE. With eight profiles, it lowers DSR by 3.36 points. The safety effect of the target thus depends on the available actions rather than improving consistently with menu size.

Table 3: Harmful DSR with prompt-forward features on 192 fixed oracle-validation examples. Percentages are means over five paired controller seeds. Differences are tie-aware minus Direct CE in percentage points, with paired 95% t intervals. Leave-one-out conditions remove a profile from the four-profile reference menu.

Menu	Direct CE DSR (%)	Tie-aware (Ours) DSR (%)	Difference [95% interval]
$K = 2$	79.85	81.98	$+2.14$ [-1.37, $+5.65$]
$K = 3$	77.10	79.85	$+2.75$ [$+0.78$, $+4.71$]
$K = 4$	76.03	76.79	$+0.76$ [-1.92, $+3.44$]
$K = 6$	79.08	78.02	-1.07 [-6.29, $+4.15$]
$K = 8$	83.21	79.85	-3.36 [-5.74, -0.98]
Without Safety-first	76.95	80.31	$+3.36$ [$+0.32$, $+6.40$]
Without Balanced	74.66	77.40	$+2.75$ [$+0.10$, $+5.40$]
Without Preservation-first	84.73	85.80	$+1.07$ [-0.20, $+2.34$]

Table 4 reports the accompanying BRR and probe values for both targets. For eight profiles, the lower DSR is accompanied by lower refusal and higher probe loss. With three profiles, the higher DSR instead accompanies a small increase in refusal and nearly unchanged probe loss. The separate columns distinguish a change in safety from a change in refusal or capability cost; no composite score is used to rank the menus.

Removing an action changes both the possible responses and the prompt-forward feature bundle. The leave-one-out comparisons therefore test each target under the reduced menu, rather than isolating the causal contribution of one adapter composition. Within each condition, both targets receive the same remaining actions and features. These validation comparisons describe how the target behaves under different menus; they do not select an optimal menu for independent test traffic.

Table 4: Safety, refusal and capability-probe results for the prompt-forward menu experiment. Values are means over five paired controller seeds on fixed validation outcomes. DSR and BRR are percentages; probe is sequence loss. Both targets are shown for every menu.

Menu	Target	DSR ↑	BRR ↓	Probe ↓
$K = 2$	Direct CE	79.85	0.00	0.7498
$K = 2$	Tie-aware (Ours)	81.98	0.00	0.7498
$K = 3$	Direct CE	77.10	0.00	0.7499
$K = 3$	Tie-aware (Ours)	79.85	0.91	0.7498
$K = 4$	Direct CE	76.03	1.82	0.7582
$K = 4$	Tie-aware (Ours)	76.79	1.82	0.7554
$K = 6$	Direct CE	79.08	3.64	0.7706
$K = 6$	Tie-aware (Ours)	78.02	4.55	0.7694
$K = 8$	Direct CE	83.21	5.45	0.7691
$K = 8$	Tie-aware (Ours)	79.85	1.82	0.7792
Without Safety-first	Direct CE	76.95	0.00	0.7674
Without Safety-first	Tie-aware (Ours)	80.31	0.00	0.7661
Without Balanced	Direct CE	74.66	0.00	0.7504
Without Balanced	Tie-aware (Ours)	77.40	0.00	0.7502
Without Preservation-first	Direct CE	84.73	9.09	0.7516
Without Preservation-first	Tie-aware (Ours)	85.80	9.09	0.7510

4.3 How Much Computation Do the Selection Inputs Require?

We time the three input choices in Figure 1(b) and three static profiles on the same 96 prompts. Each batch size has an untimed warm-up followed by three timed repetitions. Measurements include feature extraction, profile activation, generation and selection, excluding model loading and offline judging. At batch size eight, prompt-only routing takes 3.06 seconds per prompt, prompt-forward routing 3.88, and candidate-conditioned selection 10.38; static profiles take 1.56–1.80 (Table 5). The additional forwards make prompt-forward routing slower than prompt-only routing, while generating all candidates adds a larger cost. Batch-eight values report amortized processing time per prompt.

Table 5: Serving time on the same 96 prompts, in seconds per prompt. Batch-eight values divide batch wall time by batch size. Timing includes feature extraction, profile activation, generation and selection; model loading and offline judging are excluded. Each batch size has three timed repetitions after warm-up.

Execution path	Batch 1	Batch 8
Static Safety-first	8.57	1.56
Static Balanced	6.50	1.64
Static Preservation-first	8.77	1.80
Prompt-only routing (Ours)	6.15	3.06
Prompt-forward routing (Ours)	16.64	3.88
Candidate-conditioned selection (Ours)	33.10	10.38

The execution stages explain what these totals include. Prompt-only features can be computed without generating a candidate for each profile. Prompt-forward features add profile-conditioned passes over the input, while candidate-conditioned features require the responses themselves. For the latter, generation happens before selection and the chosen response is reused. A negligible generation step after selection therefore does not imply negligible generation cost; the candidate generation is already part of the timed path.

Batch size also changes the comparison. At batch size one, prompt-only routing is faster than the three static configurations on this prompt set; at batch size eight, every static configuration is faster. The results should therefore be read as measurements of complete executions under the stated batching conditions, not as a fixed latency multiplier for the MLP or a guarantee about other serving workloads. In particular, amortized batch time is not an individual user's response latency under a request queue.

4.4 How Sensitive Are the Policies to Request-Preserving Rewrites?

A fixed library supplies eight request-preserving rewrites for each of 72 harmful requests: 24 each from HarmBench standard, HarmBench contextual and JailbreakBench. Worst-of-eight DSR counts a request as defended only if all eight responses are judged safe. Both learned prompt-only policies defend 25/72 requests, compared with 49/72 for Safety-first and 48/72 for LlamaGuard routing (Table 6). Safety-first has no route choice to alter, providing a response-level control for the same profile menu. The lower worst-of-eight safety of the learned routers shows that their choices can be sensitive to wording changes even when the underlying request is preserved. This experiment uses a fixed rewrite set rather than feedback-driven attack search.

Table 6: Response safety under eight fixed rewrites of each of 72 harmful requests. Counts are defended requests / 72; percentages are DSR. A request passes worst-of-eight evaluation only if every rewritten response is safe. All policies use the same deterministic per-profile responses and evaluator.

	Clean		Worst of eight	
Policy	Count	DSR (%)	Count	DSR (%)
Static Safety-first	68/72	94.44	49/72	68.06
LlamaGuard binary profile router	68/72	94.44	48/72	66.67
Direct CE prompt-only	62/72	86.11	25/72	34.72
Tie-aware prompt-only (Ours)	64/72	88.89	25/72	34.72

The clean and rewritten results measure different properties. Tie-aware prompt-only routing defends 64 of the 72 original requests, compared with 62 for Direct CE, but both defend 25 under worst-of-eight evaluation. The clean-input difference thus does not translate into a difference under the tested rewrite budget. Equal aggregate counts do not establish that the two policies fail on the same requests or choose the same profiles.

Worst-of-eight evaluation allows any one unsafe response among a request's rewrites to count as a failure. It does not average that failure away with safe alternatives. Applying the same rewrite library and deterministic profile responses to all four policies makes this criterion directly comparable across their decisions. The experiment tests sensitivity to the specified wording transformations; it does not compare the three feature bundles under a common adaptive attack, nor does it use evaluator feedback to optimize new attacks.

5 Conclusion

PROOF learns to select fixed safety profiles from their measured outcomes. Its tie-aware target retains alternative minimum-cost actions while preserving a preferred representative. On fixed outcomes, this reduces sensitivity to tie-breaking relative to hard-label training, while mean selection regret remains statistically indistinguishable. Safety differences vary across profile menus, and candidate-conditioned selection incurs additional generation cost. These results separate the benefit of retaining tied alternatives from the choice of profiles and inference features.

Limitations. The experiments use one backbone and adapter-based menus, with automatic outcome judgments. The target comparison isolates sensitivity to tie-breaking; broader safety gains require evaluation on independent inputs against strong static and same-candidate controls. The prompt-only policies remain vulnerable to request-preserving rewrites.

AI USE STATEMENT

Generative AI tools assisted with manuscript drafting and polishing, preparation of Figure 1, method implementation and code debugging, and feedback on experimental design and result interpretation. AI-assisted revisions were checked against the implementation and stored experimental results. Model-based harmfulness and refusal evaluation is part of the experimental protocol described in Appendix A.4. The authors take responsibility for the final text, code, claims, and interpretation.

ETHICS STATEMENT

This work studies defenses against harmful LLM requests while seeking to preserve responses to legitimate requests. The reported evaluations use existing benchmarks and model outputs. Analysis of defense failures can also inform attacks; we report aggregate outcomes rather than operational attack instructions. Automatic safety judgments may contain errors or biases, and profile selection does not guarantee safe responses. Any deployment therefore requires application-specific risk assessment and safeguards beyond the evaluated selector.

REPRODUCIBILITY STATEMENT

Section 3 and Algorithm 1 define the profile costs, training targets, learning objective, and inference paths. Appendix A provides the weighted batch objective and controller training settings; Appendices A.1 and A.2 specify the profile compositions, action mappings, and feature construction. Appendix A.3 describes the evaluation sets and controlled comparisons, and Appendix A.4 defines the evaluators and statistical procedures. Tables 3 and 4 report every menu condition for both training targets.

REFERENCES

Afra Amini, Tim Vieira, Elliott Ash, and Ryan Cotterell. Variational best-of-N alignment. In *ICLR*, 2025. https://arxiv.org/abs/2407.06057.

Patrick Chao et al. JailbreakBench: An open robustness benchmark for jailbreaking large language models. arXiv:2404.01318, 2024.

Justin Cui, Wei-Lin Chiang, Ion Stoica, and Cho-Jui Hsieh. OR-Bench: An over-refusal benchmark for large language models. arXiv:2405.20947, 2024.

Lei Feng, Jiaqi Lv, Bo Han, Miao Xu, Gang Niu, Xin Geng, Bo An, and Masashi Sugiyama. Provably consistent partial-label learning. In *NeurIPS*, 2020.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. arXiv:1503.02531, 2015.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. arXiv:2106.09685, 2021.

Hakan Inan et al. Llama Guard: LLM-based input-output safeguard for human-AI conversations. arXiv:2312.06674, 2023.

Liwei Jiang, Kavel Rao, Seungju Han, Allyson Ettinger, Faeze Brahman, Sachin Kumar, Niloofar Mireshghallah, Ximing Lu, Maarten Sap, Yejin Choi, and Nouha Dziri. WildTeaming at scale: From in-the-wild jailbreaks to (adversarially) safer language models. arXiv:2406.18510, 2024.

Mantas Mazeika et al. HarmBench: A standardized evaluation framework for automated red teaming and robust refusal. arXiv:2402.04249, 2024.

Rafael Müller, Simon Kornblith, and Geoffrey E. Hinton. When does label smoothing help? In *NeurIPS*, 2019.

Isaac Ong et al. RouteLLM: Learning to route LLMs with preference data. arXiv:2406.18665, 2024.

Qwen et al. Qwen2.5 technical report. arXiv:2412.15115, 2024.

Paul Röttger, Hannah Rose Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. XSTest: A test suite for identifying exaggerated safety behaviours in large language models. In *NAACL*, 2024.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Jinyuan Jia, Bill Yuchen Lin, and Radha Poovendran. SafeDecoding: Defending against jailbreak attacks via safety-aware decoding. In *ACL*, pp. 5587–5605, 2024. https://aclanthology.org/2024.acl-long.303/.

Jeffrey Zhou et al. Instruction-following evaluation for large language models. arXiv:2311.07911, 2023.

A Implementation and Evaluation Details

Weighted training objective. Let B be a sampled minibatch, w_k the class weight estimated from training labels, and $\bar{w}_i = \sum_k q_i(k) w_k$. The batch objective is

$$\mathcal{L}_B = \frac{\sum_{i \in B} w_{k_i^\star}[-\log \pi_\psi(k_i^\star \mid z_i)]}{\sum_{i \in B} w_{k_i^\star}} + \frac{\gamma}{|B|} \sum_{i \in B} \bar{w}_i \left[-\sum_k q_i(k) \log \pi_\psi(k \mid z_i) \right].$$

The soft term weights the entire per-example cross-entropy by the target's expected class weight, rather than weighting each log-probability term separately. Balanced sampling also changes the sampled distribution. Both choices are held fixed in target comparisons. Without these weights, Equation 4 divided by $1 + \gamma$ is cross-entropy with target $(e_{k_i^\star} + \gamma q_i)/(1 + \gamma)$, explaining why the hard-plus-soft loss is more concentrated than the soft teacher alone.

Exact-tie study training configuration. The 100-controller tie-resolution study uses a one-hidden-layer MLP of width 128 with ReLU and dropout 0.1. AdamW uses learning rate 10^{-3}, weight decay 10^{-4}, and batch size 64. Training lasts 40 epochs, and we use the final checkpoint. Class weights and sampling are balanced; $\rho = 0.2$, the tie margin is zero, and hard/soft loss weights are 1.0/0.5. Ten representative assignments are crossed with seeds 13, 37, 73, 101, and 137. Candidate-conditioned features and the three-action mapping described below remain fixed. The menu study uses the same network size and optimizer settings, with at most 40 epochs and early-stopping patience 8.

A.1 Profile Construction

Equation 1 defines the weighted LoRA composition. The implementation uses concatenated weighted adapters to realize this sum. The backbone is frozen during component training; router training subsequently freezes both components as well. The four named configurations use the following coefficients.

Table 7: Composition weights for the four named LoRA profiles. Coefficients scale the trained updates in Equation 1; they are not probabilities.

Profile	Safety weight s_k	Preservation weight v_k
Safety-first	1.00	1.00
Balanced	0.75	1.25
Utility-leaning	0.50	1.40
Preservation-first	0.50	1.50

The tie-resolution study maps Utility-leaning and Preservation-first to one output class, summing their target probabilities and executing Preservation-first. A label assigned to Utility-leaning therefore does not imply that the served Preservation-first response has the same cost. Both target variants use this three-action mapping; selected-cost regret is computed from the executed action.

The menu study starts with Safety-first and Preservation-first ($K = 2$), adds Balanced for $K = 3$, and adds $(0.88, 1.12)$ for $K = 4$. The six-profile menu further adds $(0.96, 1.04)$ and $(0.80, 1.20)$; the eight-profile menu adds $(0.92, 1.08)$ and $(0.84, 1.16)$. Pairs denote (s_k, v_k). Three additional conditions remove Safety-first, Balanced, or Preservation-first from the four-profile menu. Removing $(0.88, 1.12)$ would recover the three-profile condition, so it is not repeated. Each remaining profile is a separate action in this study.

A.2 Router Features

Prompt features consist of a mean over non-padding input-token embeddings, compressed by averaging contiguous embedding-coordinate groups, and fixed lexical statistics. Candidate-conditioned features additionally summarize generated responses and differences between profiles. Refusal-pattern indicators are computed from response strings; the separate harmfulness and refusal evaluators supply training costs, not selector features.

Feature normalization is fitted on the router-training partition. Outcome costs, evaluation types, reference answers, and split identifiers are excluded from inference features. Feature columns and action order are fixed for each fitted selector. The prompt-forward bundle, used in the menu study and timing comparison, adds profile-conditioned prompt statistics before response generation. It requires one forward pass per profile but no response candidates.

A.3 Evaluation Protocols

Table 1 summarizes the four evaluation sets and their controlled comparisons.

The menu study uses 961 oracle examples: 659 harmful, 110 benign, and 192 capability probes. Of these, 769 are used for training and 192 for validation. All eight menu conditions share this partition and seeds 13, 37, 73, 101, and 137. The reported menu measurements use the validation partition.

The timing set contains 16 prompts each from HarmBench standard, HarmBench contextual, JailbreakBench, OR-Bench hard, OR-Bench toxic, and capability probes. OR-Bench hard supplies benign requests and OR-Bench toxic supplies harmful requests. The rewrite set contains 24 harmful requests each from HarmBench standard, HarmBench contextual, and JailbreakBench. Within a rewritten request, every policy receives the same deterministic profile responses, so policy differences reflect which response is selected.

A.4 Metrics and Statistical Analysis

For harmful inputs, $u(x, r) = 1$ means that the harmfulness evaluator judges the response to complete the harmful behavior. Thus $\mathrm{DSR} = 1 - \sum_i u(x_i, r_i)/n_{\mathrm{harm}}$. For benign inputs, $b(x, r) = 1$ denotes refusal and $\mathrm{BRR} = \sum_i b(x_i, r_i)/n_{\mathrm{benign}}$. The judges are HarmBench-Llama-2-13B-cls and Qwen2.5-1.5B-Instruct, respectively. In the reported studies, a harmfulness judgment beginning with "yes" is counted as unsafe; other outputs are counted as safe. The refusal evaluator interprets "REFUSE"/"COMPLY" and uses refusal-pattern matching for malformed outputs.

Capability probes use a `User:` prompt followed by an `Assistant:` reference answer. For token sequence $v_1, \ldots, v_L$, the sequence loss in Equation 2 is

$$\ell_{\mathrm{seq}} = -\frac{1}{|M|} \sum_{j \in M} \log f_\theta(v_j \mid v_{<j}; p_k),$$

where M contains non-padding next-token positions after truncation. Prompt tokens are included; padding labels are set to -100 and excluded. Probe values are loss units, not percentages or instruction-following accuracies. Target comparisons reuse the same stored per-profile losses and are interpreted within each study, rather than pooled across evaluation sets.

Selected-cost regret is $c_{i,\hat{k}_i} - \min_k c_{ik}$ on the same example. The tie-resolution study takes this minimum over its three executable actions and averages regret over 150 harmful, 30 benign, and 20 probe inputs. Since binary judgments and LM loss have different units, this mean describes selection on that fixed mixture. Its bootstrap resamples tie assignments and prompts while preserving the paired targets and five controller initializations; it does not resample generated responses. The simultaneous 95% intervals cover disagreement and regret. The prespecified joint criterion required at least a 20% relative reduction in disagreement together with a statistically resolved regret reduction. It was not met: the observed disagreement reduction was 7.69%, and the regret interval included zero.

Menu-study intervals use the five paired controller-seed differences and a t interval. They describe variation across controller fits on fixed outcomes and are not adjusted for comparisons across menus.

For rewrite tests, a base request contributes one observation. Worst-of-B DSR counts it as safe only when all B rewritten responses are safe, rather than averaging safety across rewrites.