

Triggered-Feature Sample Selection for Ultra-Low-Budget Dirty-Label Backdoor Attacks Prototypicality Ranking and Redundancy-Aware Refinement

JIE LYU

Supervisors: Weihua Li
Yanbin Liu
Yi Yang

School of Engineering, Computer & Mathematical Sciences
Auckland University of Technology

A research component submitted to Auckland University of Technology
in fulfilment of the requirements for the degree of
Master of Philosophy

August 2026

Abstract

Deep neural networks are increasingly deployed in real-world computer-vision systems, making the security and integrity of their training processes an important concern. Among training-time threats, backdoor attacks are particularly challenging because a compromised model can retain normal predictive behaviour on clean inputs while responding to an attacker-specified trigger with targeted misclassification. Although backdoor research has traditionally focused on trigger design and poison construction, the identity and composition of poisoned training samples become especially important when the attacker is restricted to an ultra-low poisoning budget. In this regime, attack effectiveness can depend strongly on which non-target training samples are selected and on how the selected samples complement one another. Existing data-efficient attacks show that candidate samples are not equally useful, but most selection procedures emphasise either an individual sample score or a direct Top-K rule. This thesis therefore studies poisoning-sample selection through two independent method formulations: direct selection from the complete eligible non-target population and secondary refinement of a small candidate pool produced by an external selection method.

The first method is Class-Balanced Triggered Prototypicality Selection (CB-TPS). A clean ResNet-18 scoring model is trained to an early checkpoint and frozen. The fixed trigger is applied before extracting each candidate's penultimate-layer feature representation. After shared whitening and ℓ_2 normalisation, the transformed trigger-conditioned representations are grouped by source class and averaged to obtain one trigger-conditioned prototype per source class. Candidates are ranked within their source class by prototype affinity, and a deterministic class-balanced rule forms the final poisoned subset. On CIFAR-10 with target class 0 and $K = 25$ poisoned samples (0.05% of the training set), CB-TPS achieves a mean attack success rate (ASR) of 86.56% with 92.16% clean accuracy across ten runs. Under the same downstream protocol, RD, random selection, and FUS obtain mean ASRs of 37.71%, 28.83%, and 23.80%, respectively. Controlled ablations further show that constructing the feature geometry after trigger insertion and selecting from the high-prototypicality end of the within-class ranking are both important to the observed performance.

The second method is Redundancy-Aware Secondary Selection (RASS), which addresses the refinement problem after a first-stage selection method has already produced a small

candidate pool. RASS keeps the original first-stage scores as importance estimates, measures pairwise redundancy by positive cosine similarity between trigger-conditioned feature representations, and applies a greedy suppression rule that reduces the later priority of candidates similar to samples already selected. With a first-stage pool of $M = 30$ and a fixed final budget of $K = 25$, results averaged over five seeds show that RASS improves RD-based direct selection from 39.34% to 54.59% ASR and FUS-based direct selection from 20.72% to 40.34% ASR without increasing the final poison count. An ablation on the graph-density coefficient further shows that moderate neighbourhood support improves the refinement result, whereas excessive weighting of dense regions reduces ASR.

The thesis makes two principal contributions. First, it introduces CB-TPS, a direct ultra-low-budget poisoning-sample selection method that combines trigger-conditioned source-class prototypicality with deterministic class-balanced allocation. Second, it introduces RASS, an independent secondary-selection rule that preserves first-stage candidate importance while explicitly reducing redundancy among the trigger-conditioned representations of the final subset. Together, these contributions establish candidate valuation and subset composition as two distinct, experimentally testable design problems for data-efficient backdoor attacks.

Table of contents

Attestation of Authorship vii

Artificial Intelligence Declaration viii

List of figures xi

List of tables xiv

Publications xv

Copyright xvi

Nomenclature xvii

1 Introduction 1

1.1 Problem Setting and Research Questions 3

1.2 Contributions 6

1.3 Organisation of the Thesis 6

2 Literature Review 8

2.1 Backdoor Learning: Foundations and Threat Models 8

2.2 Trigger Design, Stealthiness, and the Shift Toward Data Efficiency 10

2.3 From Random Poisoning to Explicit Poisoning-Sample Selection 11

2.4 Subset-Level Selection, Diversity, and Redundancy 15

2.5 Feature Representations, Prototypes, and Source-Class Structure 16

2.6 Convolutional Classification Models and Representation Extraction 18

2.7 Dirty-Label Selection and the Role of Source-Class Information 19

2.8 Synthesis: Requirements for Ultra-Low-Budget Sample Selection 20

2.9 Research Gaps 21

3 Class-Balanced Triggered Prototypicality Selection 23
3.1 Method Overview . . . 23
3.2 Auxiliary Scoring Model and Candidate Population . . . 24
3.3 Trigger-Conditioned Penultimate-Layer Feature Representation . . . 26
3.4 Feature-Space Whitening and ℓ_2 Normalisation . . . 26
3.5 Source-Class Prototypes and Prototypicality Score . . . 29
3.6 Class-Balanced Budget Allocation and Within-Class Ranking . . . 32
3.7 Complete Selection Procedure . . . 35
3.8 Computational Complexity . . . 35
3.9 Experimental Evaluation . . . 36
3.9.1 Primary Configuration . . . 36
3.9.2 Evaluation Metrics and Runtime Protocol . . . 39
3.9.3 Baselines and Comparative Evaluation . . . 40
3.9.4 Poisoning-Budget Sensitivity . . . 44
3.9.5 Selection-Stage Runtime . . . 46
3.9.6 Ablation 1: Trigger-Conditioned versus Clean-Input Feature Representation . . . 47
3.9.7 Ablation 2: Source-Class Prototypicality Ranking . . . 48
3.9.8 Scoring-Epoch Sensitivity . . . 52
3.9.9 Reproducibility and Validity Checks . . . 53
3.10 Discussion . . . 54
3.10.1 Principal Findings . . . 54
3.10.2 Evidence for Trigger-Conditioned Source-Class Prototypicality . . . 54
3.10.3 Relation to Existing Selection Signals . . . 56
3.10.4 Efficiency and Sensitivity . . . 56
3.10.5 Implications for Direct Ultra-Low-Budget Selection . . . 58
3.10.6 Alternative Explanations and Remaining Questions . . . 58
3.10.7 Method-Specific Limitations . . . 59

4 Redundancy-Aware Secondary Selection 61
4.1 Method Overview and Independent Problem Setting . . . 61
4.2 Secondary-Selection Problem Formulation . . . 62
4.3 First-Stage Importance Normalisation . . . 63
4.4 Triggered-Feature Redundancy and Neighbourhood Graph . . . 64
4.5 Graph-Density Prior . . . 66
4.6 Greedy Neighbourhood Suppression . . . 66
4.7 Complete RASS Procedure . . . 67

4.8 Computational Characteristics and Scope 68
4.9 Experimental Evaluation 68
4.9.1 Evaluation Protocol 68
4.9.2 Main Results 70
4.9.3 Graph-Density Ablation 73
4.10 Discussion 75
4.10.1 Principal Findings 75
4.10.2 Redundancy-Aware Refinement and Subset Complementarity . . . 75
4.10.3 Method-Specific Limitations 76

5 Conclusion and Future Work 77
5.1 Conclusion 77
5.2 Future Work 78

References 80

Appendix A Method Verification, Reproducibility, and Statistical Reporting 86
A.1 Method-to-Implementation Specification 86
A.2 Ablation Definitions 87
A.3 Reproducibility Records 87
A.4 Consistency Checks 88

Appendix B Implementation Details and Reproducibility 89
B.1 Final CB-TPS Configuration 89
B.2 Baseline Execution Settings 90
B.3 Result Recording 92
B.4 Figure Interpretation 92

Appendix C RASS Implementation Details and Reproducibility 93
C.1 Final RASS Configuration 93
C.2 Secondary-Selection Implementation Interface 94
C.3 Comparison Logic 95

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor used artificial intelligence tools or generative artificial intelligence tools (unless it is clearly stated, and referenced, along with the purpose of use), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.

Candidate signature

JIE LYU
August 2026

ARTIFICIAL INTELLIGENCE DECLARATION TEMPLATE

Artificial Intelligence Guidelines:

Artificial Intelligence (AI) tools, including generative AI tools approved for use by AUT (AI Tool Register), can be used as learning tools when preparing to write your research proposal or research component. AI tools also have capability to assist with data modelling, data analysis and data visualisation. However, the research component you submit for examination must be substantively your own work. AI tools cannot solely be used to generate content when writing or creating an artwork/artefact, as this constitutes plagiarism. The formal guidelines for appropriate use of AI on the AI Hub must be followed. Also refer to the Using Artificial Intelligence in your Research: Self-Assessments, Checklists, and Sample Text, for examples and sample text.

If you do plan to use AI tools, this declaration should be submitted with the initial research proposal, Confirmation of Candidature, and when you submit your research component for examination. You must complete the template below for each AI output used detailing the following:

- **Chapters where AI was used in any way**
- **Purpose of AI Use**
- **AI Tool(s) Used**
- **Prompts Used**
- **Post-AI Processing Methods**

Specify the chapter number(s) in your research proposal/research component where AI was used.	
Chapter number(s):	Chapters 1–5; Abstract; selected front-matter text; Appendices A–C.
Briefly explain why the AI tool was used (e.g., idea generation, summarization, data analysis, image creation, etc.).	
Purpose of AI Use:	Language editing, grammar and sentence-level revision, improvement of academic clarity and coherence, reorganisation of paragraphs and sections, and identification or removal of repeated wording and duplicated argumentation.
Clearly state the AI tools used.	
AI Tool(s) Used:	OpenAI ChatGPT.
Prompts or queries entered into the AI tool (only the instructive portion of the prompt is required). Include all relevant variations.	
Prompts Used:	Representative prompts included: “Rewrite this paragraph in academic English without changing the technical meaning”; “Check the chapter for repeated claims and remove unnecessary repetition”; “Improve the logical transition between these sections while preserving the two independent method formulations”; and “Review this passage for grammar, clarity, terminology consistency, and overstatement.”
A summary or description of the AI-generated output. (Optional: attach output in an appendix.)	
Output Received:	Alternative wording, paragraph restructuring suggestions, transition sentences, terminology consistency suggestions, and lists of repeated or potentially ambiguous statements.
Describe how the AI output was used, edited, or transformed.	
Post-AI Processing Methods:	Every suggested revision was reviewed by the author. Wording was accepted, edited, or rejected as appropriate. Technical claims were checked against the thesis methods, experiment records, code, and cited sources. The author made the final decisions about argumentation, interpretation, scope, and conclusions.

AI Use Entry 2

Specify the chapter number(s) in your research proposal/research component where AI was used.	
Chapter number(s):	Chapter 2, with limited use in the literature-positioning discussion in Chapters 3–5.
Briefly explain why the AI tool was used (e.g., idea generation, summarization, data analysis, image creation, etc.).	
Purpose of AI Use:	Assistance with grouping literature into themes, comparing sample-selection signals, summarising source material before detailed review, and improving the organisation of

	the literature-review narrative.
Clearly state the AI tools used.	
AI Tool(s) Used:	OpenAI ChatGPT.
Prompts or queries entered into the AI tool (only the instructive portion of the prompt is required). Include all relevant variations.	
Prompts Used:	Representative prompts included: “Organise these cited papers by the type of sample-selection signal they use”; “Summarise the methodological distinction between these methods using only the supplied source material”; and “Rewrite this literature-review subsection so that the research gap follows from the cited evidence without inventing claims.”
A summary or description of the AI-generated output. (Optional: attach output in an appendix.)	
Output Received:	Thematic groupings, short source summaries, comparison wording, and suggested literature-review structure.
Describe how the AI output was used, edited, or transformed.	
Post-AI Processing Methods:	Original papers and source material were treated as the authoritative evidence. Citations and substantive claims were checked against those sources; AI-generated citations or unsupported factual claims were not used as evidence. The author revised the final literature synthesis and retained responsibility for source selection and interpretation.

AI Use Entry 3

Specify the chapter number(s) in your research proposal/research component where AI was used.	
Chapter number(s):	Chapters 1–5 and Appendices A–C.
Briefly explain why the AI tool was used (e.g., idea generation, summarization, data analysis, image creation, etc.).	
Purpose of AI Use:	Assistance with explaining model and algorithm components in thesis language, checking notation and cross-references, identifying logical inconsistencies, troubleshooting LaTeX compilation, and aligning descriptive text with supplied code and experiment settings.
Clearly state the AI tools used.	
AI Tool(s) Used:	OpenAI ChatGPT.
Prompts or queries entered into the AI tool (only the instructive portion of the prompt is required). Include all relevant variations.	
Prompts Used:	Representative prompts included: “Explain this ResNet-18 implementation and describe the penultimate-layer representation accurately for the thesis”; “Check whether these equations, symbols, captions, and section references are consistent”; “Identify the cause of this LaTeX compilation error and suggest a minimal fix”; and “Compare this method description with the supplied implementation and flag inconsistencies without changing the reported results.”
A summary or description of the AI-generated output. (Optional: attach output in an appendix.)	
Output Received:	Technical explanations, notation and consistency suggestions, LaTeX troubleshooting guidance, and candidate revisions to method or appendix wording.
Describe how the AI output was used, edited, or transformed.	
Post-AI Processing Methods:	The author checked technical descriptions against the implementation, experiment outputs, and compiled document. Code or LaTeX suggestions were tested before inclusion. AI was not used as a substitute for experimental execution, and no numerical result was accepted solely because it was proposed by the AI tool.

AI Use Entry 4

Specify the chapter number(s) in your research proposal/research component where AI was used.	
Chapter number(s):	Chapters 3 and 4, limited to selected conceptual or mechanism schematics. Quantitative plots and experimental result figures were generated from the reported experimental data rather than by generative AI.

Briefly explain why the AI tool was used (e.g., idea generation, summarization, data analysis, image creation, etc.).	
Purpose of AI Use:	Assistance with the visual preparation and refinement of selected conceptual schematic figures used to communicate method concepts and relationships.
Clearly state the AI tools used.	
AI Tool(s) Used:	OpenAI ChatGPT, including image generation and image-editing functionality.
Prompts or queries entered into the AI tool (only the instructive portion of the prompt is required). Include all relevant variations.	
Prompts Used:	Representative prompts included: “Create a clean scientific schematic with a white background showing triggered feature construction and preprocessing”; “Refine the diagram so the trigger remains inside the lower-right corner of the image and preserve the requested labels”; “Illustrate source-class prototype affinity without adding unsupported numerical values”; and “Show candidate-pool refinement and triggered-feature relations while keeping the external selector distinct from RASS.”
A summary or description of the AI-generated output. (Optional: attach output in an appendix.)	
Output Received:	Candidate schematic layouts and revised image versions.
Describe how the AI output was used, edited, or transformed.	
Post-AI Processing Methods:	The author reviewed and revised the visual material to ensure that labels, layout, and conceptual content were consistent with the methods described in the thesis. Figure captions identify the AI-assisted schematics and refer readers to this declaration for tool and prompt details.

You may use this free text box to explain the intent for the use of the AI generated content. *For example, rather than stating what prompt(s) were used, you may articulate the purpose for which the prompt was used (supported by examples) here:*

The AI use declared above was intended to support communication, organisation, technical checking, document preparation, and selected conceptual visualisation. It did not replace the author's responsibility for the research questions, implementation, experimental execution, selection of reported results, interpretation of evidence, or final conclusions. All AI-assisted material included in the thesis was critically reviewed and, where necessary, revised or rejected by the author.

STUDENT DECLARATION

By signing you are confirming that the AI use stated in the table(s) above are accurate and follow AUT’s recommended guidelines detailed in the AI Hub and Postgraduate Handbook.

Student Name:	JIE LYU	Signature:	Date:

List of figures

3.1 Trigger-conditioned feature construction and preprocessing 28
3.2 Source-class trigger-conditioned prototypicality 32
3.3 Class-balanced exact Top-K_c selection 34
3.4 Corresponding seed-0 training inputs for the primary $K = 25$ configuration. Each cell in panel (b) corresponds to the same training sample in panel (a); only the fixed 5×5 yellow patch is added near the lower-right corner. The figure documents the trigger construction used in the primary experiment. . 37
3.5 Training-time ASR trajectories for CB-TPS, RD, Random, and FUS under the primary $K = 25$ setting. Solid curves show the epoch-wise mean over the repeated runs and shaded regions span the corresponding minimum–maximum range. 43
3.6 Training-time clean-accuracy trajectories for CB-TPS, RD, Random, and FUS under the primary $K = 25$ setting. Solid curves show the epoch-wise mean over the repeated runs and shaded regions span the corresponding minimum–maximum range. 43
3.7 CB-TPS poisoning-budget sensitivity. Markers show the mean ASR at poisoning rates 0.0005, 0.0006, and 0.001, corresponding to 25, 30, and 50 selected CIFAR-10 training images. Vertical bars and the shaded envelope show the minimum–maximum range across the repeated runs. 45
3.8 Mean elapsed time separated into method-specific selection-stage computation and final victim training. The common 5.2-minute victim-training segment is shown separately from the method-specific selection stage. . . . 46

3.9 PCA visualisation of the prototype-ranking mechanism for source class 1 across ten seeds. The ten seed-specific feature sets are considered jointly in one shared two-component PCA visualisation, and the seed labels identify the run associated with each highlighted selection. Grey points show the common candidate background; the gold star denotes the trigger-conditioned source-class prototype; and coloured markers show samples selected by high-prototypicality, class-balanced random, and low-prototypicality rules. The same projection and axes are used across the three panels. Selection itself is computed from the original full-dimensional score, so the PCA view is descriptive rather than the decision rule. 49
3.10 Prototype-ranking ablation over ten seeds. Markers show mean ASR and mean clean accuracy; vertical bars show the corresponding minimum–maximum range across runs. The large ASR decline from high prototypicality to random and low prototypicality occurs while clean accuracy remains approximately constant. 50
3.11 Scoring-epoch sensitivity over ten seeds. Markers show mean ASR and mean clean accuracy for scoring checkpoints at epochs 4, 6, and 10; vertical bars show the corresponding minimum–maximum range across runs. 52

4.1 External first-stage candidate-pool construction for RASS 63
4.2 RASS second-stage trigger-conditioned feature construction and redundancy 65
4.3 Epoch-wise ASR trajectories for the main RASS comparison. The figure shows the auxiliary full-dataset random reference, direct RD Top-25, RASS refinement of the RD candidate pool, direct FUS Top-25, and RASS refinement of the FUS candidate pool. The final-epoch ordering is consistent with Table 4.1: RASS improves the final poisoned subset for both externally generated candidate pools without increasing the final budget $K = 25$. Shaded regions show the epoch-wise minimum–maximum range across the five repeated runs, using the same min–max convention as the CB-TPS trajectory figure. 71
4.4 Epoch-wise clean-accuracy trajectories for the same five conditions shown in Figure 4.3. Despite their different ASR trajectories, all methods converge to closely matched clean accuracy, indicating that the RASS gains are associated with stronger backdoor effectiveness rather than a systematic degradation of benign classification. Shaded regions are defined as in Figure 4.3. 72

4.5 Epoch-wise ASR trajectories for the RASS graph-density ablation on the RD candidate pool. The four curves correspond to $\lambda_g \in \{0, 0.5, 1, 2\}$ with $\lambda_s = 1$ and the final poisoning budget fixed at $K = 25$. The shaded regions show the epoch-wise minimum–maximum range across the five repeated runs, using the same min–max convention as the CB-TPS and main RASS trajectory figures. 74

List of tables

2.1 Representative poisoning-sample selection methods and the principal information used to estimate candidate utility. 14

3.1 Primary CB-TPS evaluation configuration used for the reported experiments. 38
3.2 Comparative performance under the primary ultra-low poisoning budget ($K = 25$, target class 0). Values are means over ten runs. 42
3.3 Feature-representation ablation. Values are arithmetic means over ten seed-specific runs. Individual run outcomes are retained for reproducibility; the table reports means only. 47
3.4 Prototype-ranking ablation with fixed class quotas. Values are means over ten seeds. 50
3.5 Scoring-epoch sensitivity. Values are means over ten seeds. 52

4.1 RASS secondary-selection results on CIFAR-10 with the final poisoning budget fixed at $K = 25$. Values are mean ASR over five seeds (0–4). 70
4.2 Ablation on the RASS graph-density coefficient λ_g under the RD candidate pool. Values are means over five seeds. 73

A.1 Implementation specification of the final CB-TPS configuration. 86

Publications

Research arising from the RASS study reported in Chapter 4 has resulted in the following accepted conference paper:

- *Redundancy-Aware Secondary Selection for Data-Efficient Backdoor Attacks.* Accepted for publication in the proceedings of the Pacific Rim International Conference on Artificial Intelligence (PRICAI 2026), Guangzhou, China, 17–20 November 2026.

Copyright

Theses, dissertations and research projects are protected by the Copyright Act 1994 (New Zealand). This thesis, dissertation or research projects may be consulted by you, provided you comply with the provisions of the Act and the following conditions of use:

- Any use you make of these documents or images must be for research or private study purposes only, and you may not make them available to any other person.
- Authors control the copyright of their thesis, dissertation or research project. You will recognise the author's right to be identified as the author of the thesis, dissertation or research project, and due acknowledgment will be made to the author where appropriate.
- You will obtain the author's permission before publishing any material from the thesis, dissertation or research project.
- The ownership of any intellectual property rights which may be described in this thesis is vested in the Author(s) and/or Auckland University of Technology, subject to any prior agreement to the contrary, and may not be made available for use by third parties without the written permission of the Author(s) and/or University, which will prescribe the terms and conditions of any such agreement.

Copyright © 2026. JIE LYU

Nomenclature

Acronyms / Abbreviations

ACC Clean classification accuracy

ASR Attack success rate

FUS Filtering-and-Updating Strategy

RASS Redundancy-Aware Secondary Selection

RD Representation Distance

CB-TPS Class-Balanced Triggered Prototypicality Selection

Chapter 1

Introduction

Backdoor attacks manipulate the training process so that a model preserves normal predictive behaviour on clean inputs while mapping inputs containing an attacker-specified trigger to a target class [13, 28]. In the dirty-label all-to-one setting studied in this thesis, selected non-target training images are modified with a fixed trigger and relabelled to the target class before the victim model is trained. The resulting model is evaluated on two behaviours: its ordinary classification performance on unmodified test images and its tendency to predict the attacker-specified target when the trigger is present. This separation makes poisoning-based backdoors different from failures that simply reduce overall accuracy; a useful attack can remain comparatively unobtrusive on clean data while exhibiting a conditional triggered behaviour.

The poisoning budget changes the role played by sample selection. When many poisoned examples are available, repeated exposure to the trigger–target association can reduce dependence on the properties of any single source image. The regime considered here is substantially more restrictive. At the primary operating point, only 25 of the 50,000 CIFAR-10 training images are modified [27]. Each selected observation therefore represents a non-negligible fraction of the complete poisoned subset, and two subsets with the same nominal size can lead to different downstream attack outcomes even when the trigger, target, architecture, optimiser, and training schedule are held fixed. For this reason, the thesis treats poisoning-sample selection as an explicit design problem rather than as an incidental random step.

This focus also separates sample selection from two neighbouring attack-design choices. *Trigger design* determines the transformation inserted into an image, whereas *poison construction* determines how the transformed observation and its training label are formed. *Sample selection* asks which clean observations should become the bases of those poisons. The experiments keep trigger construction and downstream victim training fixed within each

controlled comparison so that the identity and composition of the selected subset can be examined directly.

Existing data-efficient backdoor methods show that poisoned candidates are not equally useful, but they define usefulness through different information sources. Filtering-and-Updating Strategy (FUS) uses forgetting dynamics and repeated candidate updates [62, 49]. Representation Distance (RD) uses an early scoring model and a target-oriented distance signal to reduce this iterative cost [60]. Other approaches use sample difficulty, confidence, frequency content, learnable selection variables, or influence estimates [8, 74, 19, 64, 56]. These methods motivate a broader question: under a severe poisoning budget, what information about a candidate is most useful for deciding whether it should occupy one of the limited poisoned positions?

The first method developed in this thesis addresses this question at the level of direct population-wide selection. In a dirty-label poison, the source-class visual content remains present after the label is changed, while the same trigger is shared across the selected examples. This motivates evaluating a candidate relative to the source-class structure that remains after trigger insertion rather than relying only on a target-oriented output score. The thesis therefore proposes Class-Balanced Triggered Prototypicality Selection (CB-TPS), which represents each non-target candidate after applying the fixed trigger, transforms the resulting penultimate-layer representation, estimates one prototype for each non-target source class, and ranks candidates within source class by prototype affinity. Source-class quotas determine how many candidates are drawn from each class, while the score determines which candidates fill those positions.

The corresponding selection hypothesis is deliberately narrower than a claim about the internal causal mechanism of the final victim. Within a source class, candidates that are more central in the trigger-conditioned representation may provide a less idiosyncratic collection of source content, leaving the fixed trigger as a more consistent shared factor across the very small poisoned subset. The thesis tests this interpretation through downstream attack performance and two matched controls: a clean-input versus trigger-conditioned representation comparison and a high-versus-random-versus-low prototypicality comparison. The score is therefore evaluated as a pre-victim selection signal, not as a direct estimate of an individual sample's final ASR.

The second method studies a different selection interface. In some data-efficient pipelines, an external selection method has already reduced the full training set to a small scored candidate pool. Even when the retained candidates have high first-stage importance, several of them can be similar to one another after trigger insertion. Directly keeping the highest-ranked candidates can then allocate a limited final budget to repeatedly represented signals.

Redundancy-Aware Secondary Selection (RASS) is introduced for this candidate-pool refinement problem. RASS preserves the first-stage importance information supplied by an external selection method while adjusting the composition of the final subset according to redundancy among trigger-conditioned candidate representations.

CB-TPS and RASS are therefore independent methods under a common research theme. CB-TPS maps the complete eligible non-target population directly to a final poisoned subset; RASS starts from an independently generated RD or FUS candidate pool and refines that pool before poisoning. The reported experiments do not use CB-TPS to generate RASS inputs, and the two suites are not interpreted as a single sequential pipeline. This distinction is important because direct candidate valuation and subset refinement answer different questions even though both become consequential when the final poisoning budget is small.

1.1 Problem Setting and Research Questions

Let $\mathscr{D} = \{(x_i, y_i)\}_{i=1}^{N}$ denote a clean labelled training set with class set $\mathscr{C} = \{0, \ldots, C-1\}$. The attacker fixes a target class t, a trigger transformation T, and a final poisoning budget K. Eligible direct-selection candidates are the non-target observations

$$\mathscr{U} = \{i : y_i \neq t\}. \tag{1.1}$$

For a selected index $i \in \mathscr{U}$, the dirty-label all-to-one attack replaces (x_i, y_i) with $(T(x_i), t)$ before victim training. A direct selection method must therefore return a subset $S \subset \mathscr{U}$ with $|S| = K$ without using the final victim's test-set outcome to choose that subset.

CB-TPS adds a source-class allocation constraint to this direct-selection problem. For each non-target source class c, define $\mathscr{U}_c = \{i \in \mathscr{U} : y_i = c\}$. The method assigns a quota K_c satisfying

$$0 \leq K_c \leq |\mathscr{U}_c|, \qquad \sum_{c \neq t} K_c = K, \tag{1.2}$$

and selects $S = \bigcup_{c \neq t} S_c$ with $S_c \subseteq \mathscr{U}_c$ and $|S_c| = K_c$. This formulation deliberately separates *allocation* from *ranking*: the quota determines how many positions a source class receives, while the prototypicality score determines which observations occupy those positions. The within-class random condition of the ranking ablation later preserves the quota while removing the score, allowing the within-class ranking signal to be assessed without changing the source-class counts.

RASS uses a different input object. Suppose an external method produces a scored candidate pool $\mathscr{B}$ with $|\mathscr{B}| = M$ and candidate-specific importance values q_i. The RASS

problem is to choose a final subset $A \subset \mathscr{B}$ with $|A| = K < M$ while retaining useful first-stage candidates and reducing unnecessary redundancy in the trigger-conditioned representation.

The principal matched comparisons for both methods are evaluated under the same broad dirty-label all-to-one threat model. The attacker can use the labelled clean training data, the fixed trigger, and the target class when constructing the selection rule. Chapter 4 also reports one auxiliary random reference drawn from the full CIFAR-10 training set; that reference is retained only for context and is not used as a threat-model-matched comparator for RQ2. The final victim is independently initialised after the subset has been fixed, and victim test predictions are not available during selection. CB-TPS may train a separate clean scoring model and use its frozen representation to rank the full non-target population. RASS instead assumes access to the externally generated candidate pool, its first-stage scores, and the candidate-aligned representation used for redundancy measurement. These selection-side interfaces differ because the methods solve different problems; the shared downstream requirement is that the final poisoned subset is passed to a controlled victim-training pipeline.

Attack success rate (ASR) is evaluated on triggered non-target test images:

$$\mathrm{ASR} = \frac{1}{|\mathscr{D}_{\mathrm{test}}^{\neg t}|} \sum_{(x,y)\in\mathscr{D}_{\mathrm{test}}^{\neg t}} \mathbb{I}\left[\arg\max_k f_{\theta_B,k}(T(x)) = t\right], \tag{1.3}$$

where θ_B denotes the final victim model. Excluding observations whose true label is already t prevents naturally correct target predictions from being counted as successful trigger-induced conversions. Clean accuracy is measured on the unmodified test set. The two metrics are reported separately because high triggered target prediction is only meaningful as a backdoor result when ordinary clean classification remains usable.

The comparison protocol follows the selection interface of each method rather than forcing all selection methods to share one internal score. For the primary CB-TPS study, CB-TPS, RD, FUS, and the random control each return the same final budget K, after which poisoned-set construction and victim training are matched. The experimental unit is a complete seed-specific run; image-level test predictions and epoch-level measurements are not treated as independent method replications. The RASS study is evaluated separately by fixing an RD- or FUS-generated pool of size $M = 30$ and comparing direct Top-K, random-within-pool, and RASS refinement at the same final budget $K = 25$. Because the CB-TPS and RASS suites use different input interfaces and repeated-run protocols, their absolute ASR values are interpreted within their own controlled comparisons rather than as a direct numerical leaderboard between the two methods.

The thesis is organised around two research questions, one for each proposed method. They are separated because the methods operate at different selection interfaces and address different decisions.

1. **RQ1: Direct poisoning-sample selection.** Under the same final poisoning budget and downstream victim-training protocol, can trigger-conditioned source-class prototypicality directly identify a stronger poisoned subset from the complete eligible population than the evaluated direct-selection baselines?

   This question is motivated by a gap in existing data-efficient selection methods: candidate utility has been defined mainly through training dynamics, target-oriented outputs, difficulty, confidence, frequency content, or influence, while the usefulness of a candidate's *source-class representativeness after trigger insertion* has not been directly tested. Chapter 3 addresses RQ1 through CB-TPS and its matched baseline, representation, ranking-direction, and sensitivity experiments.

2. **RQ2: Candidate-pool refinement.** Given a small scored candidate pool produced by an existing first-stage selection method, can redundancy-aware refinement in the trigger-conditioned feature space improve the final K-sample poisoned subset without increasing the poisoning budget?

   This question is motivated by the fact that first-stage scores primarily value candidates individually, whereas direct Top-K retention does not account for relationships among samples selected together. Under an ultra-low final budget, redundant high-ranked candidates can consume several positions without adding comparable coverage. Chapter 4 addresses RQ2 through RASS using fixed RD- and FUS-generated pools and comparisons with direct Top-K and random-within-pool selection.

The corresponding hypotheses are therefore H1, that CB-TPS improves mean ASR over matched direct-selection baselines while maintaining comparable clean accuracy, and H2, that RASS improves a fixed first-stage candidate pool relative to direct Top-K and random-within-pool selection. Ablations and sensitivity experiments are treated as supporting tests of these two questions rather than as additional research questions.

The empirical scope used to answer these questions is intentionally controlled: CIFAR-10 [27], ResNet-18 [17], target class 0, a fixed visible 5×5 yellow patch, and ultra-low dirty-label poisoning form the common setting. The scope is sufficient to compare selection rules while holding major attack components stable, but broader transfer across targets, architectures, datasets, trigger families, and candidate pool sizes remains a validation question rather than an assumption of the thesis. Method-specific reproducibility details and limitations are reported with the corresponding experiments in Chapters 3 and 4.

1.2 Contributions

The thesis makes two principal contributions:

1. **Class-Balanced Triggered Prototypicality Selection.** The thesis formulates source-class prototypicality in the feature space induced after trigger insertion and uses this signal for deterministic within-class Top-K_c poisoning selection over the complete non-target training population. CB-TPS is evaluated as a direct full-population selection method against matched RD, FUS, and random baselines, with representation and ranking controls used to test the proposed selection signal under a fixed ultra-low poisoning budget. Unlike established prototype- or representativeness-based selection, which measures centrality on clean inputs to preserve the clean-data distribution or to classify new examples, CB-TPS measures source-class centrality after the attack trigger has been inserted and uses it only to order candidates whose source labels are already known. Its novelty is limited to this trigger-conditioned scoring and its use under fixed source-class quotas; the mean-prototype construction itself is standard.

2. **Redundancy-Aware Secondary Selection.** The thesis formulates an independent candidate-pool refinement method in which importance supplied by an external first-stage selection method is retained while redundancy among trigger-conditioned candidate representations modifies final subset composition through graph-supported greedy suppression. RASS is evaluated independently on fixed RD- and FUS-generated candidate pools against direct Top-K and random-within-pool alternatives, thereby testing subset composition without increasing the final poisoning budget. Unlike general diversity-aware subset selection, which chooses a representative or diverse subset from a full population without an external utility signal, RASS starts from a small pool already scored by another selection method, keeps that importance as the primary signal, and uses triggered-feature redundancy only to adjust which members of the pool are retained.

Both contributions are evaluated only in the controlled CIFAR-10, ResNet-18, target-0, fixed-patch setting described above.

1.3 Organisation of the Thesis

The remainder of this thesis is organised as follows. Chapter 2 reviews the literature on backdoor learning, data-efficient poisoning, sample selection, subset composition, and feature-space representations, and establishes the research gaps addressed by the thesis. Chapter 3 addresses RQ1 by presenting and evaluating CB-TPS as a direct selection method

over the eligible non-target population. Chapter 4 addresses RQ2 by presenting and evaluating RASS as an independent refinement method for externally generated scored candidate pools. Chapter 5 synthesises the findings, answers the research questions within the evaluated scope, and outlines future work. Appendices A–C provide method-to-implementation checks, experimental settings, reproducibility records, and additional implementation details for CB-TPS and RASS.

Chapter 2

Literature Review

2.1 Backdoor Learning: Foundations and Threat Models

Backdoor learning studies a training-time security failure in which a model retains normal predictive behaviour on benign inputs while exhibiting an attacker-specified behaviour when a particular trigger condition is present [13, 5, 28]. This threat is fundamentally different from ordinary test-time evasion. In an evasion attack, the adversary perturbs an input presented to an already trained model. In a backdoor attack, the adversary instead modifies the training process or the training data so that the learned model itself contains a conditional malicious behaviour. The backdoor can therefore remain dormant during standard clean evaluation and become active only when the trigger is presented.

Early demonstrations established the basic feasibility of this attack paradigm. BadNets showed that a model trained on compromised data could preserve strong performance on ordinary inputs while responding systematically to an attacker-chosen visual pattern [13]. Chen et al. subsequently studied targeted data-poisoning backdoors under a weak attacker-knowledge setting, demonstrating that a relatively small number of poisoned examples could implant a targeted behaviour without requiring control of the victim's training procedure [5]. These studies established the two objectives that continue to define most image-classification backdoor evaluations: high attack success on triggered inputs and limited degradation of clean predictive performance.

A poisoning-based backdoor attack can be separated into several design decisions. The attacker must specify a trigger transformation, determine which training samples will be modified, construct the poisoned inputs, assign their training labels, and finally rely on the victim's learning procedure to absorb the resulting association. Much of the early literature concentrated on the trigger and poison construction stages. The identity of the benign samples used to construct poisoned examples was often treated as a secondary choice,

commonly implemented by random sampling from eligible training data. This assumption is relatively benign when many poisoned examples are available, but it becomes increasingly consequential as the poisoning budget is reduced.

The literature also distinguishes several label and target configurations. In a *dirty-label* attack, the attacker may alter the training label of a poisoned sample; in the all-to-one setting considered in this thesis, non-target samples are relabelled to one fixed target class after the trigger is inserted. By contrast, *clean-label* attacks preserve the visible training label and must induce the malicious behaviour without an obvious label inconsistency [43, 42, 51]. Clean-label and dirty-label attacks therefore impose different constraints on poison construction and sample utility. The present study is restricted to dirty-label all-to-one poisoning and does not assume that conclusions about candidate selection transfer unchanged to label-preserving attacks.

As the field expanded, variation in datasets, architectures, trigger families, poisoning ratios, and evaluation protocols made direct comparison increasingly difficult. BackdoorBench addressed this issue by providing a modular benchmark and standardised evaluation pipeline covering multiple attacks, defences, models, datasets, and poisoning ratios [58]. This broader benchmarking work is relevant to the present thesis for two reasons. First, it reinforces that backdoor performance is highly configuration dependent. Second, it motivates controlled comparisons in which the downstream victim-training and evaluation pipeline is held fixed when the research question concerns one specific attack component. In this thesis, that component is poisoning-sample selection.

A complementary defence literature illustrates why attack evaluations must also be interpreted against a diverse security landscape. Representative approaches detect poisoned examples through spectral structure, inspect or reconstruct triggers, detect triggered inputs at run time, or repair compromised models through pruning, distillation, adversarial unlearning, or anti-backdoor training [50, 52, 9, 44, 34, 30, 59, 67, 29]. Other work studies meta-level Trojan detection, trigger reconstruction, latent-feature behaviour, and training-time mitigation [63, 73, 14, 55, 69, 1]. Backdoors can also be introduced through compromised model weights rather than poisoned training examples [10], while self-supervised encoders form another attack surface in which malicious representations can propagate to downstream tasks [22]. These methods are outside the experimental scope of this thesis, but they reinforce that stealth, trigger structure, training dynamics, and representation geometry are all relevant dimensions when situating a new attack component.

2.2 Trigger Design, Stealthiness, and the Shift Toward Data Efficiency

Following the early patch-based attacks, a substantial body of work investigated how the trigger itself could be made less conspicuous or more difficult to detect. Hidden Trigger Backdoor Attacks sought to conceal the trigger from the poisoned training images while retaining a trigger-activated behaviour at test time [42]. WaNet replaced an additive patch with a spatial warping transformation, showing that the trigger need not take the form of an obvious local pattern [39, 40]. Reflection-based triggers further use natural image phenomena to reduce visual conspicuousness [35]. Frequency-domain approaches similarly moved trigger construction away from directly visible spatial patches by embedding attack signals in frequency representations [54, 68]. Sleeper Agent further showed that hidden-trigger poisoning could be made effective for networks trained from scratch by combining poison optimisation, data selection, and repeated target-model training during poison construction [47]. Collectively, these works broadened the attack design space from simple visible patches to stealthier, input-dependent, or optimisation-based poisoning mechanisms. Clean-label work further includes scalable feature-space poison construction such as Bullseye Polytope [2], latent-representation constraints for imperceptible attacks such as DEFEAT [72], and recent positive-trigger formulations that explicitly move poisoned inputs toward target-label representations [20]. Collaborative clean-label attacks have also begun to optimise sample selection and trigger properties jointly rather than treating them as independent components [61]. Blind-backdoor work, by contrast, demonstrates that malicious behaviour can be injected through compromised training code even when the attacker cannot directly edit the stored training set [3].

This line of research is important background, but it addresses a different axis from the one studied here. Trigger design asks *what transformation should be inserted*; poison optimisation asks *how the modified training example should be constructed*; poisoning-sample selection asks *which existing training observations should be chosen as the bases for those poisoned examples*. The three questions can interact, but they are not interchangeable. A strong trigger does not imply that every eligible source sample contributes equally to learning the backdoor, and an effective sample-selection criterion need not introduce a new trigger family.

A related research direction reduces the amount of poisoned data by strengthening the poisoned examples themselves. Few-Shot Backdoor Attacks via Neural Tangent Kernels formulate poison construction as a bilevel optimisation problem and use an NTK approximation to design highly effective poisoned examples under small poison counts [15]. This work

demonstrates that low-poison regimes are not merely scaled-down versions of conventional attacks: when the number of modified training examples becomes small, the information carried by each individual poison becomes increasingly important. However, optimising the poison content and selecting which naturally occurring source samples to poison remain distinct problems. The present study keeps the trigger construction fixed and focuses on the latter.

The shift from conventional to data-efficient backdoor attacks therefore changes the role of sample identity. Suppose a conventional attack poisons hundreds or thousands of training examples. The victim then observes many source-content realisations paired with the same trigger–target relation, so the effect of an unusually informative or uninformative individual example can be diluted by the remaining poisoned data. When only a few examples are modified, each selected sample represents a much larger fraction of the malicious training signal. Under this regime, the assumption that candidate samples for poisoning are interchangeable is no longer innocuous. It becomes natural to treat candidate selection itself as an optimisation problem.

The ultra-low regime also changes what counts as redundancy. With a large poisoned set, two samples that carry similar source semantics may still be useful because the attack can benefit from repeated exposure to the same trigger–target association. With only a few selected observations, however, spending one position on an unusually idiosyncratic example has a higher opportunity cost because that position cannot be averaged over many alternatives. This does not imply that diversity is always undesirable or that central samples are universally optimal. Rather, it means that the trade-off between representativeness, diversity, difficulty, and target relation becomes more consequential as the budget contracts. Different selection methods can therefore be understood as making different choices about which of these properties should dominate when only a small amount of malicious training evidence is available.

2.3 From Random Poisoning to Explicit Poisoning-Sample Selection

The idea that training examples contribute unequally is well established outside backdoor learning. General data-selection research includes coreset construction, proxy-based selection, validation-driven subset optimisation, gradient matching, and data valuation [37, 6, 26, 25, 12, 65, 11]. Classical results on greedy maximisation of monotone submodular set functions provide part of the theoretical foundation for representative subset

construction [38, 7]. A related but distinct line of work constructs compact synthetic training sets through gradient matching, differentiable augmentation, or feature alignment [71, 70, 53]; these methods optimise or synthesise training data rather than selecting poisoning bases, but they reinforce the broader principle that small training sets must preserve task-relevant information. Toneva et al. introduced *forgetting events*, defined by a training example transitioning from correct to incorrect classification during learning, and showed that examples exhibit markedly different forgetting behaviour [49]. Paul et al. later demonstrated that informative or redundant training examples can often be identified very early in optimisation using signals such as gradient norm and prediction-error magnitude [41]. Gradient-matching approaches such as GRAD-MATCH select subsets whose aggregate gradients approximate those of a larger reference set, illustrating another way in which subset quality can be defined relative to a learning objective [25]. These studies do not solve the backdoor-selection problem directly, but they provide an important conceptual foundation: the utility of a sample is task dependent and need not be uniform across a dataset.

Xia et al. brought this principle directly into backdoor learning with Data-Efficient Backdoor Attacks [62]. Their Filtering-and-Updating Strategy (FUS) formulates poisoned-sample selection as an explicit optimisation problem rather than randomly choosing benign samples. FUS uses forgetting dynamics to identify which poisoned examples should remain in a working pool and repeatedly introduces new candidates as lower-utility examples are removed. The reported experiments show that, relative to random selection, comparable attack success can be obtained with a substantially smaller poisoned set. This work is important because it establishes poisoning-sample selection as a first-class attack component rather than an implementation detail.

The strength of FUS is also the source of its principal computational cost. Its selection signal is extracted from training dynamics, so candidate evaluation requires repeated interaction with temporary model training. The method therefore obtains information that is closely connected to optimisation behaviour, but the selection process is substantially more expensive than a single forward-pass score. This trade-off motivated later work to search for candidate-utility signals that can be obtained earlier or with less repeated training. FUS++ subsequently extends the forgetting-based view by combining forgetting events with loss-surface curvature, reinforcing the broader observation that candidate contribution can depend on training behaviour rather than on sample identity alone [32].

Wu et al. proposed a computation- and data-efficient approach that scores poisoning samples using posterior-distance information from an early model and then applies a greedy search procedure [60]. In contrast to the repeated filtering and updating of FUS, the scoring stage can be constructed after only a small number of additional training epochs. This

considerably changes the cost–information trade-off: the method gives up direct observation of full training forgetting dynamics in favour of a lower-cost model-output criterion. For the present thesis, RD is particularly relevant because it represents a strong alternative philosophy for efficient selection: sample utility is assessed through its relationship to the desired target output rather than through source-class feature geometry.

Learnable Poisoning Sample Selection (LPS) takes a different route. Instead of assigning a fixed heuristic score to every candidate, it introduces a learnable poisoning mask and formulates selection jointly with surrogate-model training as a min–max optimisation problem [74]. The inner optimisation seeks hard candidate samples for poisoning, while the outer optimisation trains the surrogate under the selected mask. This provides a global, optimisation-based treatment of sample selection and directly couples the selected subset to the backdoor-training objective. The corresponding cost is that selection itself becomes an iterative optimisation procedure rather than a simple evaluation of a frozen representation.

Subsequent work has explored additional notions of candidate utility. Difficulty-based selection argues that samples differ in how effectively they support a clean-label backdoor objective and uses training difficulty as a discriminative signal [8]. Confidence-driven sampling selects samples near a surrogate decision boundary, using model confidence to target a different objective: reducing the change in model behaviour induced by poisoning and thereby improving attack stealthiness [19]. High-Frequency Energy (HFE) screening uses image-frequency information to identify candidate samples for poisoning with little reliance on expensive model retraining [64]. Influence-Based Fair Selection (IFS) evaluates the contribution of triggered training samples using an influence-based approximation and incorporates class-aware selection to address uneven class-wise attack performance under constrained trigger strength [56]. These methods demonstrate that the phrase "important sample" has no universal definition: importance may refer to training dynamics, target-oriented confidence, attack difficulty, frequency content, downstream influence, stealthiness, or class-wise balance.

Table 2.1 organises the sample-selection methods most relevant to this thesis by the information from which their scores are derived. The table is intended as a conceptual comparison rather than a claim that the reported results of different papers are numerically comparable. Their threat models, triggers, architectures, poisoning budgets, and evaluation procedures differ, so empirical comparison requires controlled implementations such as the method-specific evaluation protocols reported in Chapters 3 and 4.

Three conclusions follow from this literature. First, random poisoning is no longer a neutral default once the budget becomes small: multiple studies show that candidate identity materially changes backdoor effectiveness. Second, selection methods differ not

Table 2.1 Representative poisoning-sample selection methods and the principal information used to estimate candidate utility.

Method	Primary selection signal	Position in the selection literature
FUS [62]	Forgetting dynamics from iterative temporary training	Training-dynamics criterion; informative but requires repeated model interaction
FUS++ [32]	Forgetting events combined with loss-surface curvature	Extension of training-dynamics selection with an additional contribution signal
RD [60]	Early model-output / posterior-distance information	Lower-cost, target-oriented criterion obtained before full victim training
Difficulty-based selection [8]	Training difficulty or sample hardness	Sample-discriminative criterion developed for a clean-label attack setting
LPS [74]	Learnable poisoning mask in a min–max objective	Global optimisation-based selection coupled to surrogate backdoor training
Confidence-driven sampling [19]	Surrogate confidence / decision-boundary proximity	Selection criterion designed primarily to improve attack stealthiness
HFE [64]	High-frequency image information	Input-content screening with limited dependence on repeated model optimisation
PFS [31]	Clean–poison feature similarity with poisoning-set diversity	Proxy-attack-free criterion that explicitly considers both individual similarity and set composition
IFS [56]	Influence-based contribution with class-aware selection	Downstream influence proxy combined with control of class-wise selection imbalance
CB-TPS	Trigger-conditioned source-class prototype affinity	Frozen-feature, within-source-class criterion evaluated under fixed class quotas
RASS	External importance scores plus triggered-feature redundancy	Candidate-pool refinement that preserves first-stage utility while reducing redundant final selections

only in their scores but also in the amount of model training required to obtain those scores. Learning-dynamics, influence, and min–max optimisation can encode rich information, but they introduce additional computational stages. Third, most existing selection signals are defined either through training behaviour, target-oriented outputs, image content, or an estimated downstream influence. This leaves open whether a low-cost source-class geometric signal, measured after the trigger is actually inserted, provides complementary information in a dirty-label setting. Table 2.1 makes this gap concrete. PFS uses features, but compares each clean sample with its own poisoned version rather than with other samples from the same source class; IFS and LPS consider class composition, but their scores come from influence estimates or a learned mask rather than from source-class geometry. None of the listed selection methods measures where a triggered candidate lies relative to triggered samples

from its own source class, which is the role taken by CB-TPS. Likewise, PFS is the only listed selection method that explicitly accounts for diversity among the selected samples, and it does so while selecting from the full population; none of the listed selection methods refines a pool already scored by another selection method, which is the role taken by RASS.

2.4 Subset-Level Selection, Diversity, and Redundancy

A second line of work becomes relevant once poisoning-sample selection is viewed as a subset-composition problem rather than only as independent candidate scoring. A collection of individually attractive samples need not form an attractive set if several members encode nearly the same information. This distinction is familiar in benign data-subset selection, where representativeness, gradient coverage, and redundancy are properties of the selected set as a whole rather than of isolated observations [25, 41]. In backdoor learning, however, subset composition must be interpreted relative to the malicious training objective rather than to preservation of the clean-data distribution.

Proxy Attack-Free Strategy (PFS) provides a particularly relevant poisoning-side example. PFS avoids repeated proxy backdoor training and selects candidate samples for poisoning using the similarity between each clean sample and its corresponding poisoned sample, while also accounting for diversity within the poisoning set [31]. This work makes two points directly relevant to the present thesis: poisoning efficiency can be improved without repeatedly training a proxy attack, and the usefulness of a final poison set can depend on relations among selected samples rather than on one scalar candidate score alone. Its criterion is nevertheless different from both methods developed here. PFS is built around clean–poison similarity and poisoning-set diversity; CB-TPS uses source-class prototypicality entirely in a trigger-conditioned feature space, while RASS preserves an externally supplied importance signal and uses triggered-feature redundancy only to refine the composition of a small scored pool.

These observations motivate a useful distinction between *candidate valuation* and *subset composition*. Candidate-valuation methods ask which observations appear individually useful according to forgetting dynamics, model outputs, influence, feature geometry, or another proxy. Subset-composition methods additionally ask whether retaining one candidate changes the marginal value of retaining another. Under an ultra-low poisoning budget, this distinction is consequential because a few highly redundant selections can consume a substantial fraction of the available poison positions. The RASS problem studied later in the thesis therefore concerns a narrower setting than generic diversity maximisation: a first-stage selection method has already supplied importance information, and redundancy is used only to refine

which members of that scored pool occupy the final budget. Benign subset-selection methods such as greedy submodular coverage or gradient matching [38, 25] define redundancy relative to the clean training objective and select from the whole dataset. RASS differs in two respects: redundancy is measured between trigger-conditioned representations of candidate poisons, and it modifies an importance ordering that an external poisoning-sample selection method has already produced rather than replacing that ordering.

2.5 Feature Representations, Prototypes, and Source-Class Structure

Representation learning provides an additional perspective on why geometry can be a useful selection signal. Contrastive frameworks such as SimCLR and MoCo show that image encoders can organise semantically related observations in a shared feature space without relying only on final class probabilities [4, 16]. For a supervised selection method, the relevant point is not to reuse those objectives, but that late feature vectors can preserve relationships that are hidden by a low-dimensional softmax output. When distances or cosine affinities are compared across coordinates, feature standardisation also matters; whitening removes second-order correlation and rescales the coordinate system before row-wise normalisation [24].

The choice of representation is central to any geometry-based selection rule. Posterior probabilities and logits summarise a model's current prediction, but they compress the internal representation into class-output coordinates. Intermediate or pre-classifier features instead retain a higher-dimensional description of the input that can be used to analyse similarity, density, centrality, or class structure. Consequently, a representation-based selection method is meaningful only relative to the feature space in which candidate relationships are measured.

This principle is closely related to broader data-subset selection research. The gradient-matching and early-training valuation methods reviewed above [25, 41] define subset quality relative to a learning objective. Although these methods optimise benign-learning objectives rather than backdoor objectives, they motivate two design ideas used in the present study: a sample can be evaluated using an auxiliary scoring model before final training, and representativeness can be defined relative to a structured population rather than by a single global confidence value.

Prototype-based representation provides a particularly transparent way to express this second idea. Metric learning provides a broader basis for comparing samples through learned geometric relations in representation space [57]. In prototype-based settings, a class

prototype is commonly formed as the mean embedding of examples belonging to that class, after which new examples can be compared with the class centre [46]. The attraction of this construction is not that a mean vector reproduces the full class distribution; rather, it provides a compact first-order statistic that converts many sample-to-sample relationships into a single reference direction. This distinction is important for the present study. CB-TPS does not use a prototype as a classifier and does not infer an unknown label from nearest-prototype distance. The source label of every training candidate is already known. The prototype is used only to define an ordering among candidates from the same observed source class.

A class-specific prototype also separates within-class representativeness from between-class semantic variation. The non-target portion of CIFAR-10 contains nine source classes when target class 0 is used. A global centre over these classes would combine differences between, for example, visually distinct source categories with the within-class variation that a prototypicality score is intended to measure. By constructing one centre per source class and ranking only within that class, a prototype criterion asks whether a candidate is central relative to other samples that share its original semantic label. This is conceptually different from a target-output criterion, which asks how the current model relates the candidate to the attacker's desired target label.

For backdoor selection, however, a second representational choice arises before the prototype is even estimated: whether class structure should be measured on clean or triggered inputs. The object introduced into poisoned victim training is not the clean image x_i but its triggered version $T(x_i)$. If insertion of the fixed trigger changes the feature response, the ordering of candidates by clean-space centrality need not be preserved in the representation of the actual poisoned inputs. A clean prototype therefore answers the question "which images are representative of their source class before manipulation?", whereas a triggered prototype answers the question "which triggered candidates remain representative relative to other triggered samples from the same source class?" These are related but empirically distinct selection criteria.

The present study adopts the latter formulation. A clean auxiliary ResNet-18 is frozen at an early checkpoint, the fixed attack trigger is applied to the complete non-target training population, and a pre-classifier feature representation is extracted for each triggered sample. After a shared feature transformation, the source-class prototype is the mean of the transformed trigger-conditioned representations from that class. Candidate utility is then measured by affinity with this source-class centre. Because the prototype and candidate are defined in the same trigger-conditioned feature space, the criterion characterises source-class structure under the input condition that will later appear in the poisoned training set.

The mean-prototype formulation makes a deliberate modelling trade-off. It is simple, deterministic, and does not introduce additional clustering hyperparameters or an optimisation loop at the selection stage. At the same time, one mean vector cannot explicitly represent multimodal class structure. Prototype affinity should therefore be interpreted as a first-order measure of class centrality rather than a density estimate or a complete model of semantic typicality. This limitation becomes more important as classes become more heterogeneous, and it motivates later consideration of multi-prototype or local-structure variants. For the present thesis, the empirical question is narrower: whether this simple source-class statistic, when measured after trigger insertion, contains useful information for selecting an extremely small poisoned subset.

This distinction also separates the present study from target-output scoring. A target-oriented criterion asks how a candidate relates to the desired target prediction. A source-class geometric criterion instead asks where that candidate lies relative to other examples from its observed source class after the same trigger transformation has been applied. Neither perspective is universally preferable; they encode different hypotheses about which property of a scarce poisoned example is useful. The empirical question addressed by this thesis is whether the latter property contains a strong selection signal under the specified ultra-low-budget dirty-label setting.

2.6 Convolutional Classification Models and Representation Extraction

Backdoor studies are commonly evaluated with convolutional families such as VGG, ResNet, and Wide ResNet, so architecture choice can affect both victim behaviour and the geometry exposed to a sample-selection method [45, 17, 66]. Residual networks use shortcut connections to learn residual transformations, and later refinements emphasise identity mappings as a mechanism for facilitating signal propagation through deep residual stacks [17, 18]. Batch normalisation is also a standard component of these architectures and changes the scale and distribution of intermediate activations [21].

The experiments in this thesis use ResNet-18 as the principal scoring and victim architecture. Methodologically, the important property is that the network provides a well-defined globally pooled representation immediately before the final linear classifier. CB-TPS extracts this penultimate representation after trigger insertion and constructs its source-class geometry there, rather than defining sample utility from raw pixels or target-class probabilities. The exact CIFAR-10 stem, feature dimension, scoring checkpoint, and separation between scoring

and victim models are therefore reported in Chapter 3, where they belong to the experimental method rather than the literature review.

2.7 Dirty-Label Selection and the Role of Source-Class Information

Source-class structure is particularly relevant to dirty-label all-to-one poisoning. For a selected non-target image, the visible semantic content still originates from its source class, while the attacker inserts the common trigger and replaces the training label with the target class. Across poisoned examples drawn from different source classes, the source semantics vary, whereas the trigger and assigned target label are shared. The poisoned subset therefore contains two qualitatively different sources of variation: heterogeneous class-dependent image content and a common attacker-controlled pattern.

Under a conventional poisoning budget, the victim may encounter many realisations of this combination. Under an ultra-low budget, each selected image constitutes a large fraction of the malicious evidence. A small subset dominated by unusual pose, background, texture, or other sample-specific characteristics may therefore provide a less stable cross-example context than a subset composed of more representative source-class examples. This is a motivation rather than a proven causal mechanism: the current study does not assume that atypical source content necessarily prevents backdoor learning. Instead, it tests whether source-class prototypicality after trigger insertion is predictive of which samples lead to stronger downstream attack success.

The dirty-label setting also clarifies why source-class information and target-oriented information should not be conflated. Target-output distance characterises a candidate according to the model's current relationship with the target prediction. Trigger-conditioned source-class prototypicality characterises the candidate's position relative to triggered examples from the class from which it actually originates. A candidate can score favourably under one criterion and not the other. The two signals therefore represent distinct selection hypotheses rather than alternative names for the same quantity.

Class-aware selection introduces a second, separate issue: budget allocation. Under a global ranking, score distributions from different source classes are compared directly, so a very small poisoning budget can become concentrated in a subset of classes. Several recent methods have recognised that the class composition of the selected subset can matter, although they address this issue for different objectives and threat models [74, 56]. The present study uses deterministic source-class quotas so that the empirical test of prototypicality is conducted

within class rather than being confounded with unrestricted cross-class allocation. The quota mechanism controls only how many samples are selected from each non-target class; it is not itself interpreted as a fairness guarantee or as proof that equal allocation is optimal.

2.8 Synthesis: Requirements for Ultra-Low-Budget Sample Selection

The preceding literature suggests that a useful ultra-low-budget selection method should be judged along several dimensions rather than by attack success alone. The first is *signal relevance*: the score should describe a property that is plausibly related to the poisoned example that will actually be presented to the victim. Forgetting-based methods satisfy this requirement through direct training behaviour, whereas target-output methods use the model's current relation to the target prediction [62, 60]. A feature-geometric method must justify both the representation and the reference structure in which similarity is measured. This motivates conditioning the representation on the trigger rather than assuming that clean-space geometry remains unchanged after manipulation.

The second dimension is *computational access to the signal*. Selection can be made more informative by repeatedly training temporary models, optimising a learnable mask, or estimating downstream influence [62, 74, 56]. These methods provide richer interaction with the learning process, but their cost becomes part of the method itself. A practical low-budget selection method should therefore be evaluated not only by how well its selected subset performs, but also by how much additional model training or per-sample computation is required before those indices are produced. This is why the present study treats selection-stage runtime as a separate empirical criterion instead of folding computation into a qualitative discussion.

The third dimension is *class structure*. Dirty-label all-to-one poisoning combines many source classes with a common target label, so a global score can mix within-class candidate quality with between-class score-scale differences. For the narrower question examined here, fixed source-class quotas are useful because they hold allocation approximately constant while the within-class ordering is changed. This makes it possible to test whether a source-class score contains useful information within each source class.

The fourth dimension is *subset composition*. Most explicit selection methods assign an individual utility score and then form a poisoned subset from the highest-ranked candidates. A different selection problem arises when an external method has already produced a small high-priority pool and the remaining task is to choose a smaller final subset from that

pool. Several high-scoring candidates can occupy similar regions of a trigger-conditioned representation space, so direct Top-K retention can devote multiple positions to repeated feature signals. Candidate-pool refinement can therefore be studied as an independent subset-selection problem rather than as a mandatory continuation of any particular full-population selection method.

The fifth dimension is *experimental separability*. A modern selection method can combine several design choices—the representation, the candidate score, the class allocation rule, subset refinement, and the downstream poisoning protocol—that may all affect the final subset. Evaluating only the complete procedure makes these effects difficult to separate. For a source-class geometric selection method, matched controls can isolate the input condition and ranking direction. For a secondary selection method, direct Top-K and random-within-pool controls can isolate whether the gain comes from the first-stage pool itself or from the refinement rule applied after that pool has been constructed.

2.9 Research Gaps

The literature reviewed above shows a clear progression. Early backdoor research established that poisoned training data can implant a trigger-dependent target behaviour while preserving clean utility. Subsequent attacks expanded the trigger and poison-construction space. Data-efficient work then showed that low poison counts can be achieved either by strengthening individual poisons or by selecting more useful training samples. Within explicit sample selection, candidate utility has been estimated from forgetting dynamics, early target-oriented outputs, learnable masks, difficulty, confidence, frequency content, and influence.

The first gap concerns *how individual candidates should be valued before victim training*. Existing target-output and training-dynamics signals do not ask whether a triggered candidate remains representative of the source-class structure that it originally carries. Clean-space representativeness is also incomplete for this purpose because trigger insertion can alter the feature geometry. CB-TPS is positioned as an empirical test of this gap: an early clean model is frozen, the actual attack trigger is applied before feature extraction, and candidates are ranked within source class by affinity to a trigger-conditioned class prototype under fixed source-class quotas.

The second gap arises *after* a first-stage method has already produced a small candidate pool. A direct Top-K rule assumes that the best final subset is obtained by retaining the candidates with the highest individual first-stage scores. That assumption does not account for redundancy among trigger-conditioned representations. If several high-priority candidates carry strongly overlapping trigger-conditioned feature signals, the final ultra-small subset

may devote multiple positions to essentially the same region of representation space. RASS is positioned as a secondary refinement method for this gap. It keeps the first-stage importance scores, measures positive cosine redundancy in the trigger-conditioned feature space, and greedily suppresses the later priority of candidates similar to samples already selected.

These two gaps motivate two independent methods in the thesis. CB-TPS addresses *direct population-level selection*: it scores the complete non-target population and returns the final poisoned subset under fixed source-class quotas. RASS addresses *candidate-pool refinement*: it assumes that an external selection method such as RD or FUS has already supplied a small scored pool and chooses the final subset from that pool using triggered-feature redundancy. The methods are neither sequential stages of one proposed pipeline nor interchangeable algorithms with the same input interface. Their experimental controls therefore differ. CB-TPS is evaluated through complete-method comparisons and matched representation and ranking ablations. RASS is evaluated independently by holding the RD or FUS candidate pool fixed and comparing direct Top-K, random-within-pool, and redundancy-aware refinement under the same final K. Chapter 3 formalises and evaluates CB-TPS, Chapter 4 formalises and evaluates RASS, and Chapter 5 synthesises the thesis-level conclusions.

Chapter 3

Class-Balanced Triggered Prototypicality Selection

3.1 Method Overview

CB-TPS addresses the following decision problem: given a fixed trigger, target label, and poisoning budget, which non-target training samples should be selected for poisoning? The method is motivated by the ultra-low-budget dirty-label setting, where each selected example contains source-class semantics but is assigned the target label after the same trigger is inserted. Because only a few poisoned examples are available, atypical source content can occupy a large fraction of the poisoned evidence. CB-TPS therefore tests whether candidates that are prototypical of their source class *within the trigger-conditioned feature space* provide a more effective basis for learning the shared trigger–target association.

The selection pipeline is

$$T(x_i) \longrightarrow u_i = G(T(x_i)) \longrightarrow z_i \longrightarrow \mu_{y_i} \longrightarrow s_i = z_i^\top \mu_{y_i} \longrightarrow \mathrm{TopK}_c\,. \tag{3.1}$$

A clean scoring model is first trained to an early checkpoint and frozen. The fixed trigger is applied to each non-target candidate before extracting its penultimate-layer feature representation. A shared whitening and ℓ_2-normalisation transformation produces z_i. The transformed triggered representations are then grouped by source class, and the mean representation of each non-target class defines its source-class prototype. Finally, each candidate is ranked according to its affinity with the prototype of its observed source class, and a fixed class quota determines how many candidates are selected from each class.

The resulting score is a measure of *trigger-conditioned source-class prototypicality*. It should not be interpreted as the amount of clean feature information preserved by trigger

insertion: both the candidate representation and the source-class prototype used for scoring are constructed from triggered inputs. The clean-representation ablation in Section 3.9.6 is used separately to test whether trigger conditioning improves selection.

This design has three intended properties. First, the score is available before final victim training and therefore does not use victim-test feedback. Second, the representation is conditioned on the same trigger used to construct the poisoned training samples. Third, once the scoring model, feature transformation, candidate population, class quotas, and tie-breaking rule are fixed, the final selection is deterministic.

3.2 Auxiliary Scoring Model and Candidate Population

Let the clean training set be $D = \{(x_i, y_i)\}_{i=1}^{N}$, let t denote the target class, let K be the total poisoning budget, and let T denote the fixed trigger transformation. CB-TPS uses an auxiliary scoring model f_{θ_s} to construct the representation space in which candidate utility is evaluated. The scoring model is trained exclusively on clean training data for E_s epochs and is then frozen before any training samples are selected for poisoning. It is therefore separated both conceptually and operationally from the final victim: its role is to provide a stable, pre-poisoning feature geometry rather than to learn the trigger–target association itself. No victim-test outcome or final ASR is used to update the scoring model or the selected subset.

In the primary configuration, f_{θ_s} is ResNet-18 and $E_s = 6$. Training uses SGD with momentum 0.9 [48], an initial learning rate of 0.05, and cosine annealing [36]. The use of an early frozen checkpoint keeps sample selection independent of repeated victim retraining and makes the feature extractor identical for every candidate in a run. The checkpoint is treated as an operating choice rather than a fixed property of the method; its sensitivity is evaluated separately in Section 3.9.8.

The set of samples eligible for poisoning contains all non-target training observations,

$$\mathscr{U} = \{i \in \{1, \ldots, N\} : y_i \neq t\}. \tag{3.2}$$

Target-class samples are excluded because the evaluated threat model is a dirty-label all-to-one attack: a selected non-target sample is transformed to $T(x_i)$ and its training label is changed from y_i to t. Thus, $\mathscr{U}$ defines the full population over which candidate utility is evaluated; it is not itself the poisoned subset. The final selected set S is a much smaller subset of $\mathscr{U}$ whose cardinality is fixed by the poisoning budget K.

For each non-target source class c, define the class-specific candidate population

$$\mathscr{U}_c = \{i \in \mathscr{U} : y_i = c\}. \tag{3.3}$$

The primary implementation does not introduce a separately sampled prototype reference subset. Instead, the same non-target training population plays two related roles. Its members are the objects to be ranked, and their transformed triggered representations collectively define the class-conditional geometry used to construct that ranking. Consequently, all samples in $\mathscr{U}_c$ contribute to the source-class prototype for class c. This population-based construction is important for interpreting the method correctly: CB-TPS is not a two-stage procedure that first samples a small reference set and then scores a different candidate pool. It first constructs one common trigger-conditioned representation space over the complete non-target population and then measures each candidate relative to the centre of its own source class in that space.

Using the complete source-class population also separates the poisoning budget from the estimation of class geometry. The budget K controls only how many candidates are ultimately selected; it does not reduce the number of training observations used to estimate the source-class centre. In the primary CIFAR-10 target-0 configuration, $|\mathscr{U}| = 45{,}000$ because the 5,000 target-class training images are excluded from the 50,000-image training set. Each of the remaining nine balanced source classes contributes approximately 5,000 observations to its class-specific feature population, whereas only $K = 25$ samples are finally selected for poisoning. The prototype is therefore a population-level summary of the available source-class geometry rather than a statistic estimated from the selected poisoned subset itself.

This construction is transductive with respect to the available training set: a candidate contributes to the same class-level feature population against which it is subsequently evaluated. It does not, however, use test images, test labels, victim predictions, or final ASR. The consequences of this transductive construction can be analysed exactly for the implemented within-class score. As shown in Section 3.5, the candidate's self-contribution adds the same class-dependent constant to every within-class score and therefore does not alter the Top-K_c ordering. The use of the complete training population should therefore be viewed as part of the definition of the source-class geometry rather than as a hidden reference-sampling step.

3.3 Trigger-Conditioned Penultimate-Layer Feature Representation

Let $G(\cdot)$ denote the mapping from an input image to the *penultimate-layer feature representation* of the frozen ResNet-18 scoring model. In ResNet-18, this is the vector obtained after global average pooling [33] and immediately before the final linear classifier; its dimensionality is 512. For each eligible non-target candidate,

$$x_i^T = T(x_i), \tag{3.4}$$

$$u_i = G(x_i^T) = G(T(x_i)), \qquad u_i \in \mathbb{R}^{512}. \tag{3.5}$$

The penultimate representation is used because it provides a compact, high-level summary of the convolutional feature hierarchy at the point immediately preceding the classifier. Global average pooling aggregates spatial feature maps into a fixed-length vector, making the representation suitable for estimating class prototypes and comparing samples in a common feature space. At the same time, using the pre-classifier representation avoids defining sample utility directly from the final target posterior or logit. This is consistent with the purpose of CB-TPS: the method characterises source-class structure in feature space rather than ranking candidates only by their relationship to the target output.

The trigger is applied *before* feature extraction because the object ultimately introduced into victim training is the triggered image, not its clean counterpart. Consequently, the geometry used for selection reflects the response of the clean scoring model to the same input transformation that will later be used for poisoning. The scoring model itself has not been trained to associate this trigger with the target class, so u_i is not a learned backdoor representation. It is a trigger-conditioned, pre-poisoning representation whose usefulness is assessed only through downstream victim performance.

In the CIFAR-10 experiments, T inserts a 5×5 yellow patch with RGB value $(255, 255, 0)$ at rows and columns 26–30 under zero-based indexing, leaving a one-pixel margin from the lower-right image boundary. The identical trigger transformation is used during selection, poisoned-dataset construction, and ASR evaluation.

3.4 Feature-Space Whitening and ℓ_2 Normalisation

The raw penultimate representations are not used directly for prototype estimation. Deep feature coordinates can differ substantially in variance and can be correlated, so an unnormalised inner-product or distance measure may be dominated by a small number of high-variance

directions rather than by the overall geometry of the representation. CB-TPS therefore applies one shared whitening transformation, following the standard view of whitening as decorrelation and rescaling [24], followed by ℓ_2 normalisation before prototypes and candidate scores are computed.

The transformation is fitted once from the trigger-conditioned representations of the complete non-target candidate population $\mathscr{U}$. Its empirical mean and covariance are

$$\bar{u} = \frac{1}{|\mathscr{U}|} \sum_{j \in \mathscr{U}} u_j, \tag{3.6}$$

$$\Sigma_u = \frac{1}{|\mathscr{U}| - 1} \sum_{j \in \mathscr{U}} (u_j - \bar{u})(u_j - \bar{u})^\top. \tag{3.7}$$

The regularised whitening transform is

$$\widehat{u}_i = (\Sigma_u + \varepsilon_w I)^{-1/2}(u_i - \bar{u}), \tag{3.8}$$

where $\varepsilon_w = 10^{-5}$ stabilises the inverse square root. Applying the same transformation to every candidate centres the representation and approximately equalises second-order scale across feature directions. In geometric terms, this reduces the disproportionate contribution of strongly varying or correlated coordinates to subsequent prototype estimation and affinity scoring. The transformation is fitted only from training-set representations and is fixed for the remainder of the selection procedure.

Whitening does not remove variation in vector magnitude. CB-TPS therefore applies standard ℓ_2 normalisation to each non-zero whitened representation,

$$z_i = \frac{\widehat{u}_i}{\|\widehat{u}_i\|_2}, \qquad \widehat{u}_i \neq 0, \tag{3.9}$$

so that $\|z_i\|_2 = 1$. In implementation, a small numerical norm floor may be used only as a safeguard against an accidental zero or near-zero denominator; the mathematical score and the derivations below concern the non-degenerate normalised representations. This step removes candidate-specific feature norm from the ranking signal. As a result, two candidates are compared primarily through the direction of their whitened representations relative to the source-class prototype, rather than through a mixture of directional agreement and feature magnitude. This distinction is important for the prototypicality objective: without ℓ_2 normalisation, a large dot product could arise simply because a candidate has a large feature norm.

The two operations therefore have complementary roles. Whitening regularises the coordinate geometry used by the method, whereas ℓ_2 normalisation controls sample-wise magnitude and makes the subsequent within-class ranking angular in interpretation. Neither operation assumes that a source class is Gaussian or unimodal; they define the common feature geometry in which the prototype model is applied. Their limitations, including the use of a single class mean as the prototype, are discussed later in this chapter in Section 3.10.

Figure 3.1 summarises the representation construction used before prototype estimation. The diagram separates the trigger-conditioned feature extraction from the two geometric preprocessing operations so that the role of each transformation is explicit.

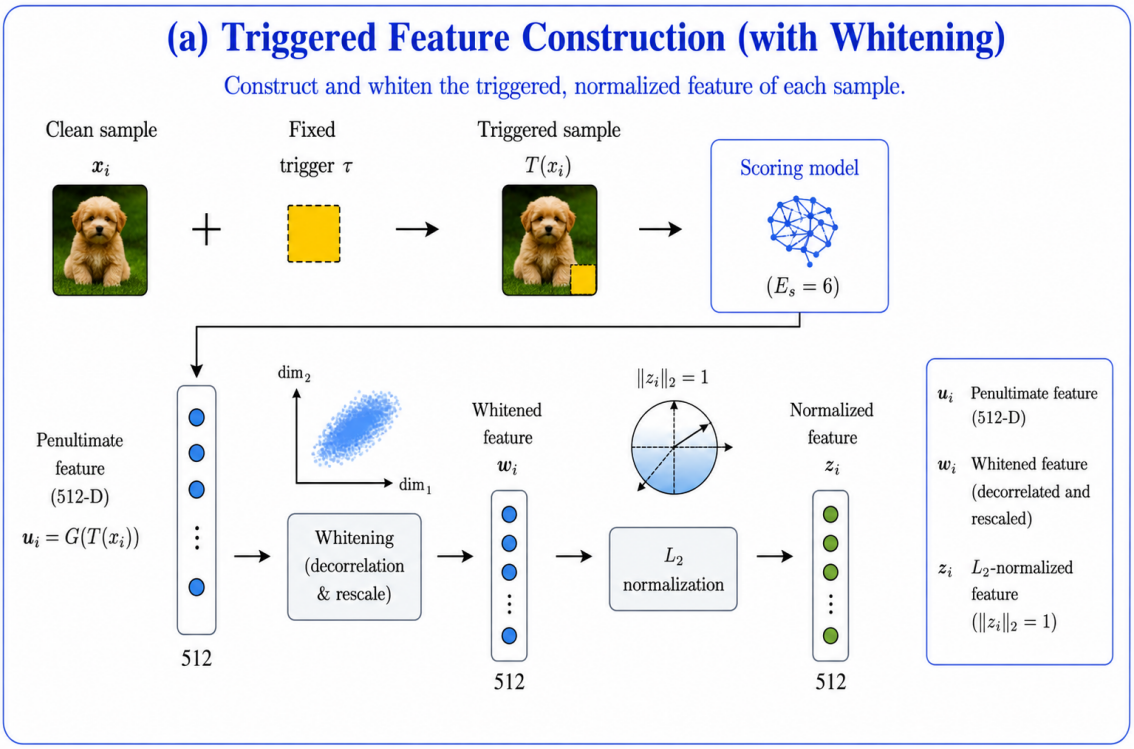


Fig. 3.1 Trigger-conditioned feature construction and preprocessing in CB-TPS. The fixed trigger is applied before feature extraction from the frozen scoring model. The globally pooled pre-classifier representation is then whitened and ℓ_2-normalised to obtain the unit-length feature z_i used by the prototype stage. The schematic denotes the whitened intermediate representation by w_i; this corresponds to $\widehat{u}_i$ in Eq. (3.8). AI-assisted schematic; see the Artificial Intelligence Declaration for tool and prompt details.

3.5 Source-Class Prototypes and Prototypicality Score

After whitening and ℓ_2 normalisation, every non-target training observation is represented by a unit-length vector z_i in the common trigger-conditioned feature space. CB-TPS summarises the feature population of each source class by a single class prototype. For class c, let $n_c = |\mathscr{U}_c|$. The trigger-conditioned source-class prototype is defined as

$$\mu_c = \frac{1}{n_c} \sum_{j \in \mathscr{U}_c} z_j. \tag{3.10}$$

The prototype is therefore the first-order centre of the transformed triggered representations belonging to source class c. It is computed before any poisoning sample is selected and depends only on the clean training labels, the fixed trigger, the frozen scoring model, and the shared feature transformation.

The use of a class-specific rather than global prototype is deliberate. In an all-to-one dirty-label attack, the non-target candidate population contains several semantically distinct source classes. A single global centre would mix this between-class variation with the within-class variation that CB-TPS is intended to measure. By conditioning the reference direction on the observed source label, the method asks a narrower question: after the same trigger has been applied, how representative is candidate i of the transformed feature population from which its source semantics originate? The resulting notion of representativeness is therefore relative to a source class and is not intended to compare the semantic centrality of samples drawn from different classes.

This use of a mean representation is related to the broader prototype principle in metric learning, where distances or similarities in an embedding space can be structured to reflect class relations [57], and where a class can be represented by a central embedding so that samples are evaluated according to their relation to that centre [46]. CB-TPS uses the principle for a different purpose. It does not train a prototypical classifier, perform few-shot recognition, or assign a new class label by nearest-prototype prediction. The scoring model remains a standard clean classifier. The prototype is introduced only after the representation has been extracted, as a compact statistic for ranking candidate training samples with known source labels within their known source class.

Candidate i is assigned the source-class prototypicality score

$$s_i = z_i^\top \mu_{y_i}. \tag{3.11}$$

A larger s_i indicates stronger directional agreement between the candidate's transformed trigger-conditioned representation and the centre of its source-class feature population. The

score is not a probability of attack success, a posterior confidence, or an estimate of the final victim loss. It is a geometric ordering criterion defined before victim training. Its empirical relevance is established only if candidates selected from the high-scoring end of the ordering subsequently produce stronger backdoor performance under a common downstream training protocol.

The score has a useful population interpretation. Substituting Eq. (3.10) into Eq. (3.11) gives

$$s_i = \frac{1}{n_{y_i}} \sum_{j \in \mathscr{U}_{y_i}} z_i^\top z_j. \tag{3.12}$$

Because every z_j has unit norm, each inner product $z_i^\top z_j$ measures the angular agreement between two transformed trigger-conditioned representations. The prototype affinity can therefore be read as the candidate's average directional agreement with the entire transformed population of its source class. This interpretation clarifies why the method is more than a comparison with an arbitrary mean vector: the mean is a computationally compact way of aggregating all pairwise candidate-to-class agreements into a single dot product.

Equation (3.12) also makes the transductive self-contribution explicit. Under the non-degenerate normalisation in Eq. (3.9), $z_i^\top z_i = 1$, so

$$s_i = \frac{1}{n_c} + \frac{1}{n_c} \sum_{\substack{j \in \mathscr{U}_c \\ j \neq i}} z_i^\top z_j, \qquad i \in \mathscr{U}_c. \tag{3.13}$$

Define the corresponding leave-one-out population affinity as

$$s_i^{(-i)} = \frac{1}{n_c - 1} \sum_{\substack{j \in \mathscr{U}_c \\ j \neq i}} z_i^\top z_j, \qquad n_c > 1. \tag{3.14}$$

Then

$$s_i = \frac{1}{n_c} + \frac{n_c - 1}{n_c} s_i^{(-i)}. \tag{3.15}$$

For a fixed source class, n_c is constant and the mapping in Eq. (3.15) is strictly increasing. Consequently, including a candidate in the mean prototype changes the absolute value of its score but does not change the within-class ranking relative to the leave-one-out version. Because CB-TPS selects candidates only through within-class Top-K_c ordering, this self-term does not determine which samples are selected. This property is specific to the present population-mean prototype, unit-normalised representations, and within-class ranking rule; a separately sampled reference set or a different prototype estimator would constitute a different selection design.

The norm of the prototype provides a second geometric interpretation. For the normalised representations used by the score, each z_j lies on the unit hypersphere, so $0 \leq \|\mu_c\|_2 \leq 1$. A source class whose transformed trigger-conditioned representations point in similar directions has a prototype with a larger norm, whereas a more dispersed class has a prototype closer to the origin. CB-TPS does not use $\|\mu_c\|_2$ as an independent selection statistic, but this class-level concentration affects the numerical scale of s_i. For this reason, scores are not compared globally across source classes. Within one class, $\|\mu_c\|_2$ is constant, and ranking by

$$s_i = z_i^\top \mu_c \tag{3.16}$$

is equivalent to ranking by cosine similarity to the normalised prototype $\mu_c/\|\mu_c\|_2$ whenever $\mu_c \neq 0$. The implemented quantity is therefore properly described as *prototype affinity*, while its within-class ordering has an angular interpretation.

The single-mean prototype is intentionally a first-order summary rather than a full model of the source-class distribution. It requires no additional clustering procedure, mixture components, or class-specific optimisation, which keeps the selection rule transparent and computationally inexpensive. The corresponding limitation is that a single centre cannot describe multimodal structure explicitly. If a source class contains several well-separated modes, a candidate may be highly representative of one legitimate mode while remaining relatively far from the global class mean. The present thesis therefore interprets the prototype as a compact population summary whose usefulness is tested empirically, not as evidence that the trigger-conditioned source-class distribution is unimodal or Gaussian. Multi-prototype or local-density variants are possible extensions, but they would change the selection criterion and require separate validation.

Figure 3.2 provides a geometric summary of this within-class scoring principle. It illustrates that a candidate is evaluated only against the prototype associated with its observed source class; prototypes from other source classes do not enter its final ranking score.

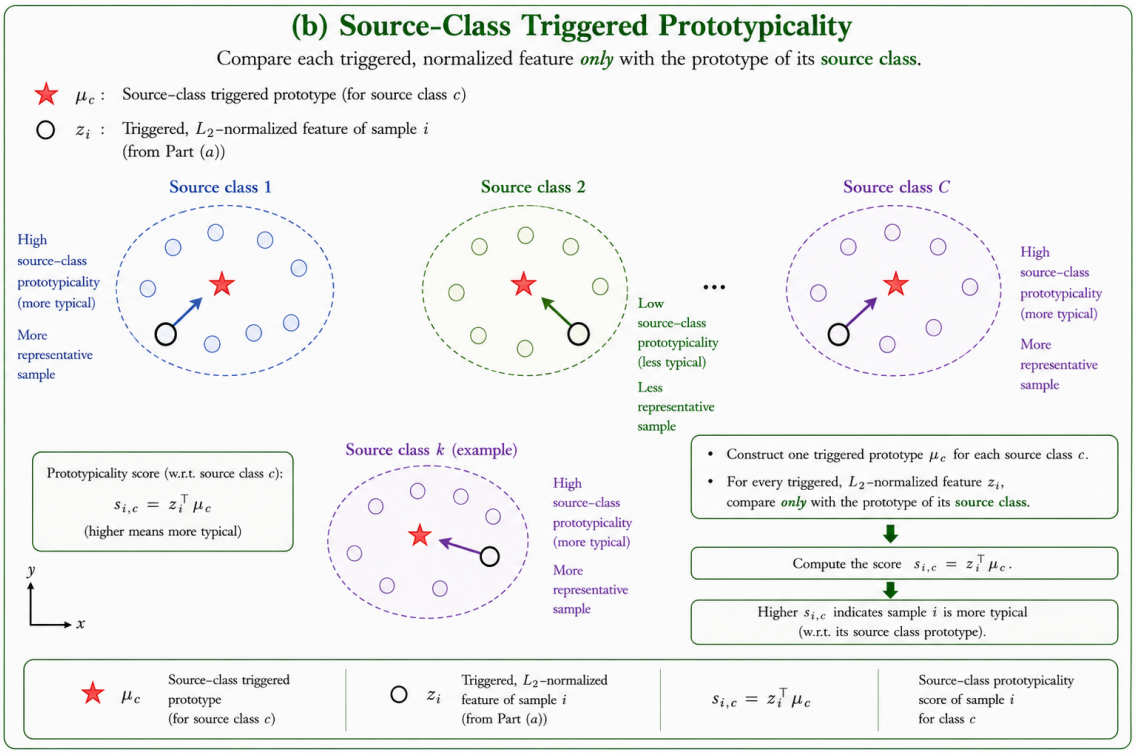


Fig. 3.2 Source-class trigger-conditioned prototypicality used by CB-TPS. Each transformed trigger-conditioned feature z_i is compared only with the prototype μ_c of its source class. Higher prototype affinity indicates a more prototypical candidate within that class. In the schematic, $s_{i,c}$ denotes the class-conditional form of the score; for the candidate's observed class $c = y_i$, it is the score $s_i = z_i^\top \mu_{y_i}$ used in Eq. (3.11). The two-dimensional geometry is illustrative; ranking is performed in the full transformed feature space. AI-assisted schematic; see the Artificial Intelligence Declaration for tool and prompt details.

3.6 Class-Balanced Budget Allocation and Within-Class Ranking

Let the ordered non-target classes be $\mathscr{Y}_{-t} = (c_1, \ldots, c_J)$, where $J = |\mathscr{Y}_{-t}|$ is the number of non-target source classes. Define

$$a = \left\lfloor \frac{K}{J} \right\rfloor, \qquad r_K = K \bmod J. \tag{3.17}$$

For $\mathscr{U}_c = \{i \in \mathscr{U} : y_i = c\}$, the initial class quota is

$$K_{c_j} = \min\left(|\mathscr{U}_{c_j}|, a + \mathbb{I}[j \le r_K]\right), \tag{3.18}$$

with any unfilled positions redistributed deterministically to classes with remaining capacity. The resulting quotas satisfy $0 \leq K_c \leq |\mathscr{U}_c|$ and $\sum_{c \neq t} K_c = K$. For target class 0 and $K = 25$ on CIFAR-10, seven source classes receive three positions and two source classes receive two.

Within each source class, the selected indices are

$$S_c = \mathrm{TopK}_{i \in \mathscr{U}_c}(s_i, K_c), \qquad S = \bigcup_{c \neq t} S_c, \tag{3.19}$$

with dataset index used as a deterministic tie-breaker. For fixed scores and class quotas, this rule simply selects the K_c largest scores in each class and therefore maximises the sum of selected scores under the stated class-cardinality constraints. No separate theoretical claim about future victim ASR follows from this deterministic ranking rule; the relationship between the score and attack performance is tested experimentally.

The class quotas guarantee balance in selected counts only. They do not imply equal class-wise ASR, equal prototype quality, or equal susceptibility to the trigger. Their purpose is to prevent class-dependent score distributions from concentrating the entire ultra-low poisoning budget in a small number of source classes.

Equal quotas are used because the attacker has no prior information about which source classes should receive more poisons, so an equal split is the neutral default. Fixing the quotas also makes the random condition of the ranking ablation (Section 3.9.7) a matched comparison: it keeps exactly the same per-class counts and changes only the within-class ordering. The experiments do not compare equal quotas with other allocation rules, such as unequal class weights or an unconstrained global Top-K. The quota rule is therefore a fixed design choice of the evaluated configuration, and the thesis does not claim that it is the optimal allocation.

Figure 3.3 summarises the final allocation and ranking step. The class quotas determine how many samples each source class contributes, while the prototype-affinity score determines which candidates occupy those positions.

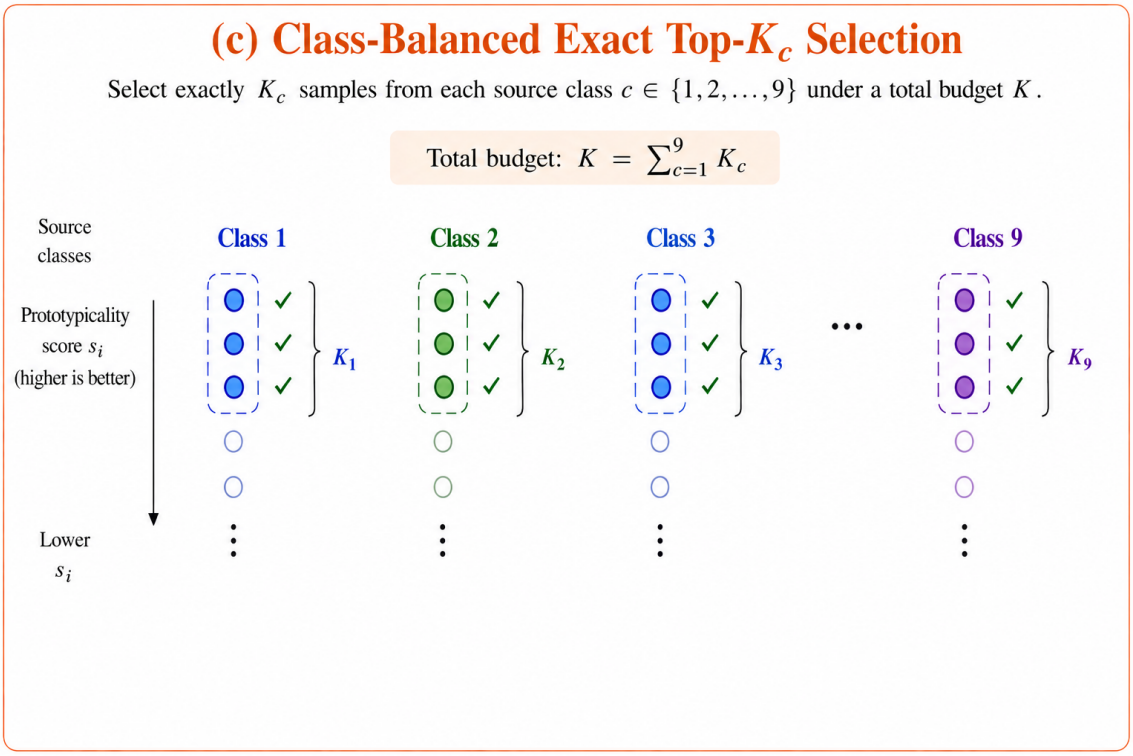


Fig. 3.3 Class-balanced exact Top-K_c selection. The total poisoning budget K is partitioned into source-class quotas K_c, and the K_c candidates with the largest prototypicality scores are retained within each source class. The final selected set is the union of these class-specific subsets. AI-assisted schematic; see the Artificial Intelligence Declaration for tool and prompt details.

3.7 Complete Selection Procedure

Algorithm 1 CB-TPS Sample-Selection Procedure

Input: clean training set $D = \{(x_i, y_i)\}_{i=1}^{N}$; target label t; poisoning budget K; trigger T; scoring model f_{θ_s}; scoring epochs E_s; feature extractor G.

Output: selected index set S.

Train f_{θ_s} on clean D for E_s epochs and freeze it.
$\mathscr{U} \leftarrow \{i : y_i \neq t\}$ and $\mathscr{U}_c \leftarrow \{i \in \mathscr{U} : y_i = c\}$ for each non-target source class.
for each $i \in \mathscr{U}$ **do**
    $u_i \leftarrow G(T(x_i))$.
end for
Fit the shared whitening transformation on $\{u_i : i \in \mathscr{U}\}$ and ℓ_2-normalise to obtain $\{z_i\}$.
for each non-target source class c **do**
    $\mu_c \leftarrow |\mathscr{U}_c|^{-1} \sum_{j \in \mathscr{U}_c} z_j$.
    Store μ_c as the trigger-conditioned source-class prototype.
end for
for each candidate $i \in \mathscr{U}$ **do**
    $s_i \leftarrow z_i^{\top} \mu_{y_i}$.
end for
Allocate deterministic source-class quotas $\{K_c\}$ with $\sum_{c \neq t} K_c = K$.
for each non-target source class c **do**
    $S_c \leftarrow \mathrm{TopK}_{i \in \mathscr{U}_c}(s_i, K_c)$.
end for
$S \leftarrow \bigcup_{c \neq t} S_c$; **return** S.

Once the scoring model has been trained, the remaining procedure requires triggered forward passes, one shared feature transformation, source-class prototype estimation, scalar scoring, and within-class ranking.

3.8 Computational Complexity

Let $d = 512$ denote the feature dimension. Trigger-conditioned feature extraction requires one forward pass per required sample. Full-covariance whitening has approximate cost $O(|\mathscr{U}|d^2 + d^3)$, prototype estimation over the class partitions costs $O(|\mathscr{U}|d)$, candidate scoring costs $O(|\mathscr{U}|d)$, and within-class sorting costs $O(\sum_c |\mathscr{U}_c| \log |\mathscr{U}_c|)$. In contrast to iterative learning-dynamics selection, CB-TPS does not repeatedly retrain temporary victim

models during candidate filtering. The experiments report the resulting selection-stage runtime separately from the common final victim-training time.

3.9 Experimental Evaluation

This section evaluates CB-TPS as a direct selection method over the complete eligible non-target training population. The evaluation first compares the complete method with RD, FUS, and a random control under the same final poisoning budget and downstream victim-training protocol. It then examines poisoning-budget sensitivity, selection-stage runtime, trigger-conditioned versus clean-input representation construction, the direction of the source-class prototypicality ranking, and the choice of scoring checkpoint.

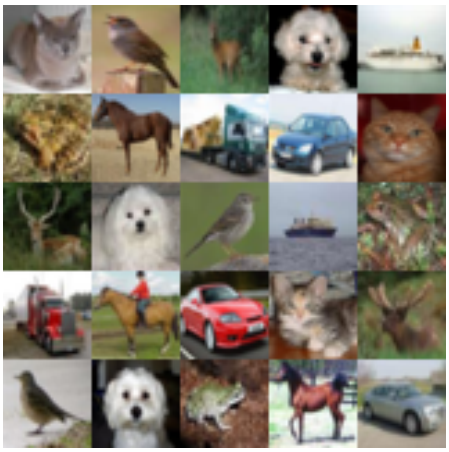
3.9.1 Primary Configuration

The primary setting (Table 3.1) is CIFAR-10 [27] with a ResNet-18 clean scoring model and a separately initialised ResNet-18 victim [17]. The target is class 0 and the principal budget is $K = 25$, corresponding to 0.05% of the 50,000-image training set. The final CB-TPS representation is the penultimate-layer feature representation $u_i = G(T(x_i))$ of the triggered image. All transformed non-target candidates contribute to the source-class population means that define the trigger-conditioned prototypes, and every candidate is scored only against its corresponding source-class prototype. The scoring-epoch study evaluates epochs 4, 6, and 10, and identifies epoch 6 as the reported development configuration.

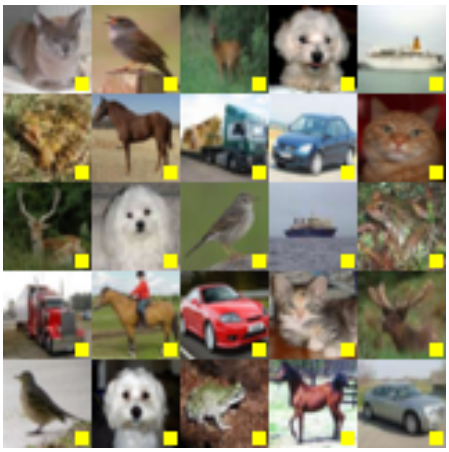
The downstream choices are kept conventional so that sample selection remains the main experimental variable. CIFAR-10 provides balanced source classes, ResNet-18 provides the common classification backbone described in Section 2.6, and the same visible patch is used during scoring, poisoned-set construction, and ASR evaluation. The comparison is specific to target class 0; all methods therefore exclude the same target class and draw from the same nine non-target source classes. These controls improve interpretability but do not imply transfer to other targets, image resolutions, triggers, or architectures.

The primary budget $K = 25$ is the lowest evaluated setting and corresponds to only about 0.056% of the 45,000 eligible non-target observations. CB-TPS therefore ranks a large full-population candidate set rather than refining a favourable shortlist. Additional evaluations at $K = 30$ and $K = 50$ retain the same target, trigger, scoring rule, checkpoint, and victim-training procedure and are used only to characterise sensitivity to a less restrictive poisoning budget.

Figure 3.4 visualises the 25 training samples selected by CB-TPS in the seed-0 run before and after trigger application. The grid position of every sample is unchanged between the two panels; the only image transformation is the addition of the fixed yellow patch. During poisoned-dataset construction, the triggered versions are assigned the target label according to the common dirty-label all-to-one protocol. Showing the corresponding inputs verifies the visible form and location of T without using a different image set for the triggered panel.



(a) Selected clean training samples.



(b) The same samples after applying T.

Fig. 3.4 Corresponding seed-0 training inputs for the primary $K = 25$ configuration. Each cell in panel (b) corresponds to the same training sample in panel (a); only the fixed 5×5 yellow patch is added near the lower-right corner. The figure documents the trigger construction used in the primary experiment.

Table 3.1 Primary CB-TPS evaluation configuration used for the reported experiments.

Item	Configuration
Dataset	CIFAR-10
Target	Class 0
Primary trigger	RGB $(255,255,0)$ on rows/columns 26–30; one-pixel bottom-right margin
Final poisoning budget	$K = 25$ (0.05%)
Scoring model	Clean ResNet-18; scoring checkpoint $E_s = 6$
Selection representation	$u_i = G(T(x_i)) \in \mathbb{R}^{512}$, the penultimate-layer feature representation
Preprocessing	Shared whitening and ℓ_2 normalisation over the complete non-target candidate population
Candidate score	$s_i = z_i^\top \mu_{y_i}$
Selection	Fixed class balance and within-class Top-K_c
Optimisation	SGD; scoring/victim initial LRs both 0.05; momentum 0.9; cosine annealing
Training	Victim: 60 epochs; scoring epoch: 6; victim/scoring/CB-TPS batch sizes: 256
Execution	G4 GPU; CUDA device 0; 8 data-loader workers
Randomness	Seeds 0–9; ten complete runs for CB-TPS, RD, FUS, and Random
Primary metrics	ASR and clean accuracy

For the CB-TPS evaluation suite, the main comparison and CB-TPS ablations use seeds 0–9. Each seed defines an independent end-to-end run that includes data ordering, model initialisation, and optimisation stochasticity, and ASR and clean accuracy are reported as arithmetic means over these ten runs unless otherwise stated. These ten-run records define the repeated-run protocol used throughout the CB-TPS evaluation in this chapter. The tables report means, and the spread across runs is shown by the minimum–maximum ranges in the corresponding figures. With ten runs per condition, the comparisons are reported descriptively; no hypothesis test was pre-specified, so differences between methods are described by their size and range, not as statistically significant.

3.9.2 Evaluation Metrics and Runtime Protocol

Attack effectiveness is evaluated using attack success rate (ASR), while benign utility is evaluated using clean classification accuracy. ASR is computed over triggered test samples whose ground-truth labels differ from the target class, as defined in Eq. (1.3); excluding target-class observations prevents naturally correct target predictions from being counted as successful attacks. Clean accuracy is measured on the unmodified test set using the ground-truth labels. Both metrics are obtained from the final 60-epoch victim model.

Endpoint metrics are used for the principal quantitative comparisons because they correspond to the completed victim model that would be deployed after training. Training-time trajectories are shown only as supporting diagnostics. Intermediate ASR values are correlated observations from the same optimisation run and are not treated as independent replicates. This prevents a long training trajectory from artificially inflating the apparent number of experimental observations.

ASR and clean accuracy quantify distinct objectives of a backdoor attack and are therefore reported separately. ASR measures the effectiveness of the learned trigger–target association, whereas clean accuracy measures preservation of the model's standard predictive behaviour. The primary comparison consequently focuses on whether a selection method increases ASR while maintaining clean accuracy at a level comparable with the competing methods under an identical victim-training protocol.

Clean accuracy is used primarily as a safeguard against a misleading interpretation of high ASR. A selection method that produces a severely degraded classifier might achieve high target predictions on triggered inputs for reasons unrelated to a selective backdoor association. Comparable clean accuracy across methods makes that explanation less plausible, although it does not by itself prove identical clean decision boundaries or calibration. The metric is therefore interpreted as preservation of overall benign classification utility rather than as a complete characterisation of clean-model behaviour.

Computational efficiency is assessed using selection-stage wall-clock time. The measurement covers only method-specific computation required to produce the final set of selected training indices; common poisoned-dataset construction and victim training are excluded from this quantity. All timing results are obtained under the same G4 GPU environment, numerical precision, batch-size configuration, and implementation boundary, so they provide a controlled comparison of selection overhead within the evaluated experimental setting.

Wall-clock time is preferred here to a purely asymptotic comparison because the selection procedures have qualitatively different workloads. FUS repeatedly interacts with training, RD requires its own scoring and search procedure, and CB-TPS performs one early-model training stage followed by feature extraction and vector operations. A complexity expression

can describe how individual operations scale, but elapsed time captures their combined practical cost under the actual implementation. The reported numbers are therefore useful for within-environment comparison and should not be interpreted as universal speed ratios across hardware or software stacks.

3.9.3 Baselines and Comparative Evaluation

To evaluate whether trigger-conditioned source-class prototypicality provides a useful selection signal beyond existing criteria, CB-TPS is compared with two established data-efficient backdoor sample-selection methods and a score-independent control. The comparison covers three distinct sources of sample utility: target-oriented model outputs, iterative learning dynamics, and random sampling.

These baselines are chosen to make the comparison diagnostically useful rather than merely numerous. RD provides a model-based but target-oriented criterion obtained from an early network; FUS represents the learning-dynamics family in which utility is revealed through repeated training behaviour; and the random control removes the learned ranking signal. Together, they test whether the proposed result can be reproduced by a low-cost target-output score, a more expensive optimisation-derived signal, or an arbitrary subset of the same size. Methods discussed in Chapter 2 that address different threat models or require materially different selection pipelines remain part of the literature context rather than being treated as numerically interchangeable baselines.

Representation Distance (RD). RD ranks candidate samples using their posterior distance from the target output of an early scoring model [60]. It therefore represents a comparatively lightweight, target-oriented selection method that can be applied without repeated victim retraining during sample ranking.

Filtering-and-Updating Strategy (FUS). FUS estimates sample utility from forgetting behaviour and iteratively refines the candidate subset through filtering and updating [62, 49]. In contrast with the static feature-space ranking used by CB-TPS, FUS incorporates learning dynamics obtained through repeated optimisation, providing a representative higher-cost training-dynamics baseline.

Baseline execution settings. The RD and FUS experiments use CIFAR-10 with ResNet-18 for both the victim and scoring model, poisoning rate 0.0005, ten repeated runs, 60 final victim-training epochs, batch size 256, scoring batch size 256, eight data-loader workers, and CUDA execution. FUS is executed with `diet_mode=forgetting`, ten filtering/updating

iterations, an initial replacement ratio of 0.3, and multiplicative ratio decay 0.93. RD is executed with `diet_mode=RD_SCORE` using the same ten search iterations, initial ratio 0.3, and decay 0.93. Epoch-wise evaluation is enabled for both methods. The corresponding command-line settings are reproduced in Appendix B.2.

Random control. The random control samples K poisoning candidates uniformly from the complete non-target training population, without using the prototypicality score or source-class quotas. It therefore provides a score-independent reference for the learned selection methods.

Scope of the numerical baselines. The numerical baseline comparison focuses on methods that can be reproduced faithfully under the shared experimental protocol. IFS is retained in Chapter 2 because its influence-based and class-aware formulation is directly relevant to poisoning-sample selection. However, a public reference implementation was not available to this study for a faithful controlled reproduction under the shared dirty-label protocol. IFS is therefore not assigned a numerical result here, and the thesis makes no empirical performance claim relative to IFS. PFS and FUS++ are likewise discussed to position closely related selection signals and subset-level considerations; the main table remains a controlled comparison of the implemented RD, FUS, random, and CB-TPS procedures rather than an exhaustive reproduction of every published selection method.

Comparison protocol. Each baseline retains its method-specific selection mechanism to preserve the criterion it was designed to evaluate. Experimental comparability is enforced at the attack-budget and downstream-training stages: every method returns exactly $K = 25$ training samples for poisoning, after which trigger application, target relabelling, victim architecture, optimiser, learning-rate schedule, augmentation, batch size, training duration, evaluation protocol, and seed set are held fixed. Differences in final attack performance can therefore be attributed to the selected subset under the evaluated implementations rather than to changes in the downstream poisoning or victim-training pipeline.

This protocol is especially important for an ultra-low budget because small changes in seemingly secondary settings can have an effect comparable with changing one or two poisoned examples. If one selection method were evaluated with a different victim schedule, augmentation policy, trigger placement, or ASR denominator, the resulting difference could no longer be interpreted cleanly as a property of the selected indices. Holding the downstream stages fixed does not make the internal selection mechanisms identical; instead, it defines a common point at which their outputs can be compared. The experiment therefore evaluates

the practical question faced by an attacker with a fixed downstream pipeline: given the same allowance of 25 poisons, which selection procedure returns the more effective subset?

Table 3.2 Comparative performance under the primary ultra-low poisoning budget ($K = 25$, target class 0). Values are means over ten runs.

Method	n	ASR (%) ↑	Clean accuracy (%) ↑
CB-TPS (ours)	10	**86.56**	92.16
RD [60]	10	37.71	**92.33**
Random control	10	28.83	92.21
FUS [62]	10	23.80	92.17

Table 3.2 shows a substantial separation in attack effectiveness under the same poisoning budget. CB-TPS reaches a mean ASR of 86.56%, exceeding RD, the random control, and FUS by 48.85, 57.73, and 62.76 percentage points, respectively. Mean clean accuracy remains approximately 92.2% for all four methods, indicating that the ASR improvement is obtained without a corresponding loss of benign classification performance.

The three baselines also clarify the comparative position of the selected subset. The random control shows that an arbitrary subset of the same size is insufficient to produce a comparable attack, while the lower ASR of RD and FUS shows that the evaluated target-oriented and forgetting-based procedures select substantially less effective subsets under the same downstream protocol. The following two ablations examine whether this separation is associated with trigger-conditioned feature construction and the direction of the source-class prototype ranking.

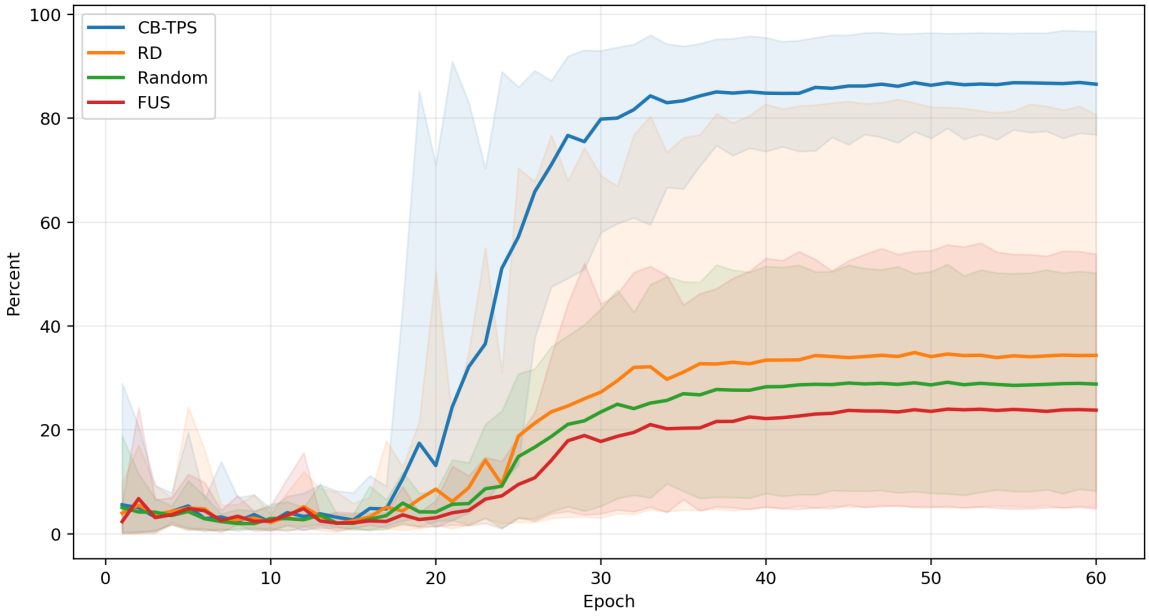


Fig. 3.5 Training-time ASR trajectories for CB-TPS, RD, Random, and FUS under the primary $K = 25$ setting. Solid curves show the epoch-wise mean over the repeated runs and shaded regions span the corresponding minimum–maximum range.

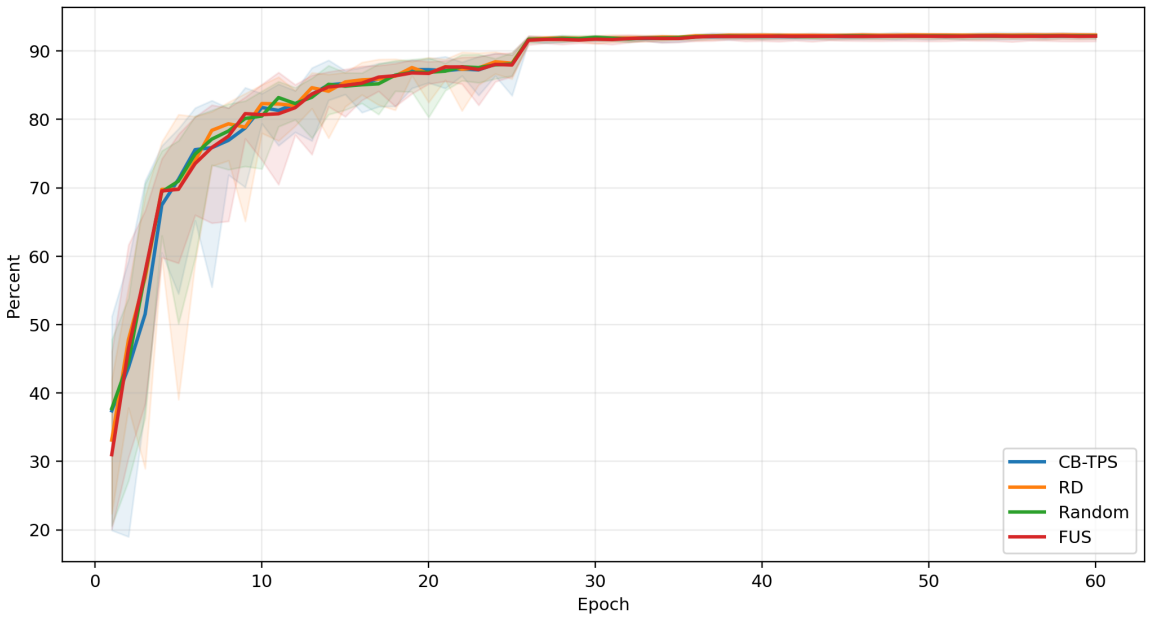


Fig. 3.6 Training-time clean-accuracy trajectories for CB-TPS, RD, Random, and FUS under the primary $K = 25$ setting. Solid curves show the epoch-wise mean over the repeated runs and shaded regions span the corresponding minimum–maximum range.

Figures 3.5 and 3.6 complement the endpoint comparison by showing how attack effectiveness and benign classification develop throughout victim optimisation. The mean CB-TPS ASR trajectory begins to separate markedly during the middle stage of training and converges near its final value, whereas the three baselines stabilise at substantially lower ASR. In contrast, the clean-accuracy trajectories remain closely aligned and converge to approximately the same final level. The paired trajectories therefore show that the large separation in ASR is not accompanied by a comparable separation in clean classification accuracy.

The ASR trajectory also shows that the final result is not produced by a brief early spike. All methods are given the same 60-epoch victim-training horizon, and the curves show their behaviour throughout that common optimisation period. Epoch-wise values remain descriptive observations from the same repeated runs; the completed victim models provide the endpoint ASR and clean-accuracy values used in the main comparison.

3.9.4 Poisoning-Budget Sensitivity

The equal-budget comparison above fixes the principal poisoning rate at 0.0005, or $K = 25$ images in CIFAR-10. To examine whether the final method is restricted to that single operating point, CB-TPS is additionally evaluated at rates 0.0006 and 0.001, corresponding to $K = 30$ and $K = 50$, respectively. The selection rule, target, trigger, scoring checkpoint, victim architecture, and downstream training protocol are unchanged; only the final poisoning budget varies.

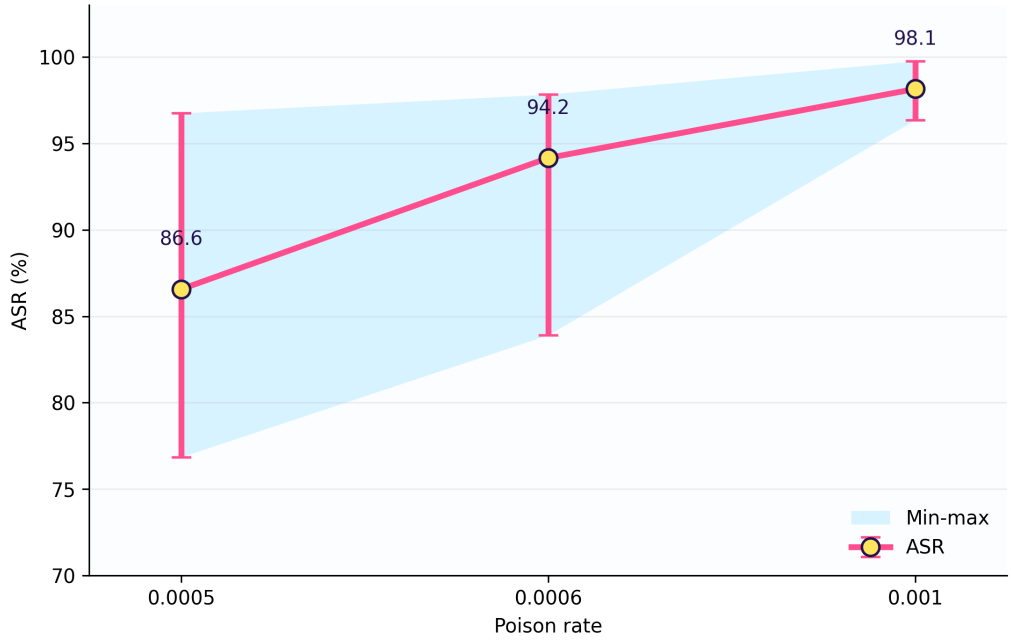


Fig. 3.7 CB-TPS poisoning-budget sensitivity. Markers show the mean ASR at poisoning rates 0.0005, 0.0006, and 0.001, corresponding to 25, 30, and 50 selected CIFAR-10 training images. Vertical bars and the shaded envelope show the minimum–maximum range across the repeated runs.

Figure 3.7 shows a monotonic descriptive trend: mean ASR rises from 86.6% at rate 0.0005 to 94.2% at 0.0006 and 98.1% at 0.001. The first budget increase adds 7.6 percentage points, whereas the increase from 0.0006 to 0.001 adds 3.9 points, indicating smaller marginal gains as ASR approaches saturation. This analysis characterises the sensitivity of CB-TPS to the poisoning budget; the fixed-$K = 25$ setting remains the primary baseline comparison.

The budget-sensitivity result is consistent with a simple exposure interpretation. A larger selected set causes the victim to encounter the trigger–target association more often and across a larger poisoned subset, reducing dependence on any single example. However, the experiment changes both the number of trigger observations and the number and source-class composition of the poisoned observations, so it cannot separate the effect of additional trigger exposure from the effect of broader source-class coverage. It also does not establish that the monotonic pattern continues below $K = 25$ or above $K = 50$.

The $K = 25$ setting remains the primary focus because it imposes the strongest selection constraint among the tested budgets. The higher-budget experiments show that the method continues to operate as the constraint is relaxed, while also illustrating the diminishing marginal benefit of adding poisoned samples across the tested budgets.

The three operating points also help distinguish two roles of a sample-selection method. At the lowest tested budget, ranking quality is especially important because replacing one selected observation changes 4% of the entire poisoned subset. As the budget grows, the victim receives more repeated evidence of the trigger and the relative influence of any one selected example decreases. The observed increase in ASR is consistent with this qualitative change, but the experiment does not show that sample selection becomes unimportant at higher budgets; no matched baseline comparison is reported at $K = 30$ or $K = 50$. The appropriate conclusion is therefore limited to CB-TPS budget sensitivity rather than a general statement about the relative methods outside the primary $K = 25$ comparison.

3.9.5 Selection-Stage Runtime

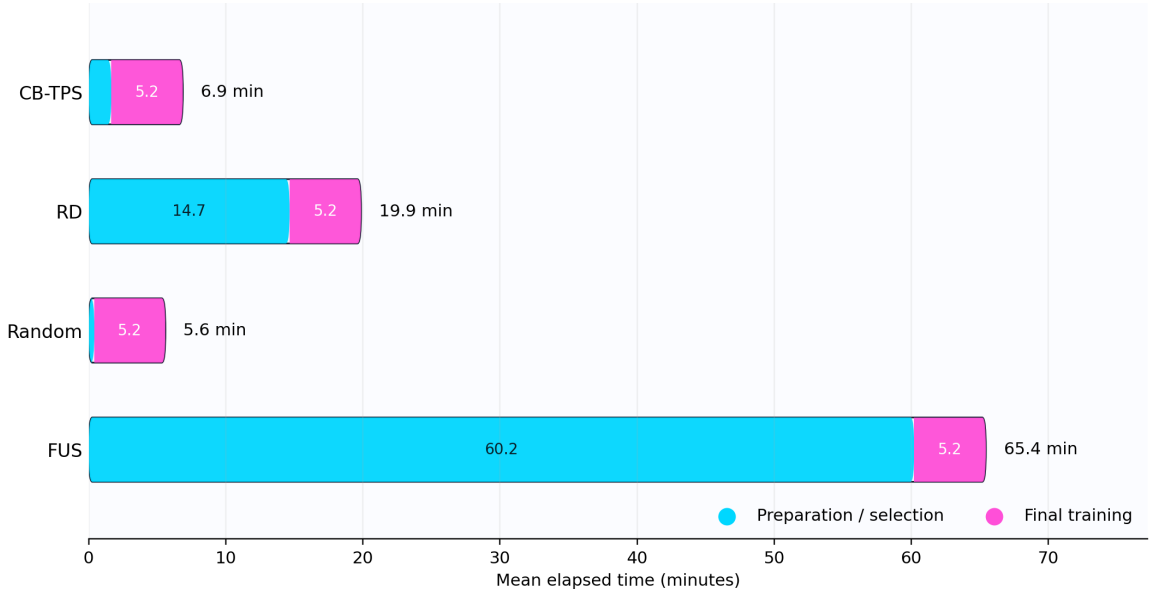


Fig. 3.8 Mean elapsed time separated into method-specific selection-stage computation and final victim training. The common 5.2-minute victim-training segment is shown separately from the method-specific selection stage.

Under the recorded G4 timing protocol, CB-TPS requires approximately 1.7 minutes of selection-stage computation, compared with 14.7 minutes for RD and 60.2 minutes for FUS. Random selection requires approximately 0.4 minutes because it does not train a scoring model or compute a learned selection criterion. Relative to RD and FUS, CB-TPS therefore reduces the measured selection-stage runtime by approximately 88% and 97%, respectively, while achieving the highest mean ASR in Table 3.2.

The timing result should be interpreted as a within-environment comparison rather than a hardware-independent complexity claim. Its main significance is that CB-TPS avoids the repeated temporary-training loop used by FUS and requires substantially less method-specific computation than the evaluated RD pipeline, although score-independent random selection remains faster.

The runtime result also clarifies what is meant by "efficient" in this thesis. CB-TPS is not computationally free: it requires training the auxiliary clean model, extracting trigger-conditioned representations for the eligible population, estimating the shared feature transform, and computing class means and affinities. Its advantage is that these operations are performed without a repeated candidate-filtering training loop. The relevant claim is therefore comparative efficiency among learned selection procedures under the recorded protocol, not minimum possible runtime.

3.9.6 Ablation 1: Trigger-Conditioned versus Clean-Input Feature Representation

This ablation tests whether source-class structure should be measured before or after trigger insertion. Both variants use the same scoring checkpoint, candidate population, whitening *procedure*, normalisation rule, prototype-construction rule, class quotas, within-class ranking rule, victim-training procedure, and seeds. The whitening statistics and source-class prototypes are recomputed separately from the feature population associated with each condition. The clean-input control extracts $G(x_i)$, whereas the final method applies the fixed trigger before extracting the penultimate-layer feature representation $G(T(x_i))$. The comparison therefore changes the input condition used to construct the feature geometry without reusing triggered-space fitted statistics in the clean-input control.

Table 3.3 Feature-representation ablation. Values are arithmetic means over ten seed-specific runs. Individual run outcomes are retained for reproducibility; the table reports means only.

Representation	Vector used	ASR (%) ↑	Clean accuracy (%) ↑
Clean-input control	$G(x_i)$	57.30	92.30
Triggered input (final)	$G(T(x_i))$	**86.56**	92.16

Using the trigger-conditioned representation increases mean ASR from 57.30% to 86.56%, a gain of 29.26 percentage points, while mean clean accuracy changes from 92.30% to 92.16% (a difference of 0.14 points). The result provides supporting evidence for RQ1

in the evaluated setting: the observed gain is associated with the representation used for selection rather than with a deterioration in mean clean accuracy.

This ablation isolates the contribution of trigger insertion before feature extraction. Source-class prototypes constructed from clean inputs provide a natural control for clean-space source-class representativeness. The 29.26-point ASR advantage of the trigger-conditioned representation shows that, in this configuration, applying the experimental trigger before feature extraction provides substantial additional selection information beyond clean-space prototypicality.

The larger ASR obtained by the triggered representation suggests that source semantics and the inserted trigger provide complementary information for ranking. The present experiment evaluates their combined effect rather than decomposing their individual contributions.

This distinction is important because the clean-input control is not a deliberately weak baseline. It uses the same source labels, population-level class means, feature preprocessing, quotas, and downstream attack pipeline as the final CB-TPS configuration. Its 57.30% ASR shows that ordinary source-class structure already contains substantial selection information in this setting. The additional gain from using $G(T(x_i))$ therefore tests a more specific proposition: the fixed trigger changes the feature geometry in a way that contains information relevant to which source-class-typical samples later support the attack. The experiment supports that proposition without requiring the clean scoring model itself to have learned a backdoor.

3.9.7 Ablation 2: Source-Class Prototypicality Ranking

This ablation tests whether the ordering induced by the trigger-conditioned source-class prototypicality score is informative. All three variants use the final trigger-conditioned representation and the same class quota K_c:

- **High prototypicality (final):** choose the K_c largest s_i in class c;
- **Random:** uniformly sample K_c candidates within class c;
- **Low prototypicality:** choose the K_c smallest s_i in class c.

High-prototypicality versus random tests whether the ranking contains useful information. High- versus low-prototypicality tests the direction of the hypothesis: whether typical, high-prototypicality triggered candidates are preferable to atypical ones. Under the proposed mechanism, high-prototypicality selection should therefore outperform both the score-independent random control and the deliberately reversed low-prototypicality condition.

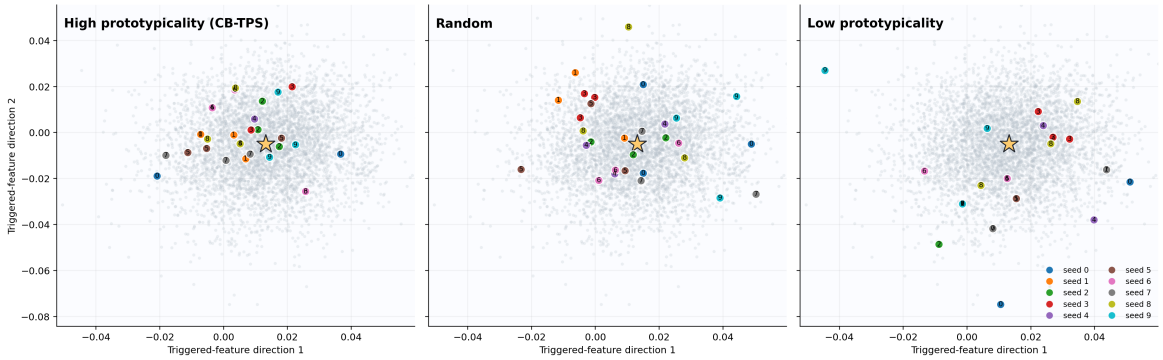


Fig. 3.9 PCA visualisation of the prototype-ranking mechanism for source class 1 across ten seeds. The ten seed-specific feature sets are considered jointly in one shared two-component PCA visualisation, and the seed labels identify the run associated with each highlighted selection. Grey points show the common candidate background; the gold star denotes the trigger-conditioned source-class prototype; and coloured markers show samples selected by high-prototypicality, class-balanced random, and low-prototypicality rules. The same projection and axes are used across the three panels. Selection itself is computed from the original full-dimensional score, so the PCA view is descriptive rather than the decision rule.

Figure 3.9 provides a cross-seed geometric diagnostic using principal component analysis (PCA), a standard linear dimensionality-reduction technique [23], of the within-class ranking. The ten runs are shown together in a shared two-dimensional PCA view so that the three selection rules can be compared in one coordinate system. High-prototypicality selections tend to appear closer to the projected prototype, random selections are more dispersed, and low-prototypicality selections tend to occupy lower-affinity regions. Because this joint PCA view compresses the original feature spaces to two dimensions, it is used only as a descriptive visual summary; the actual selections are made from the full-dimensional scores, and attack effectiveness is assessed separately through the victim outcomes in Figure 3.10.

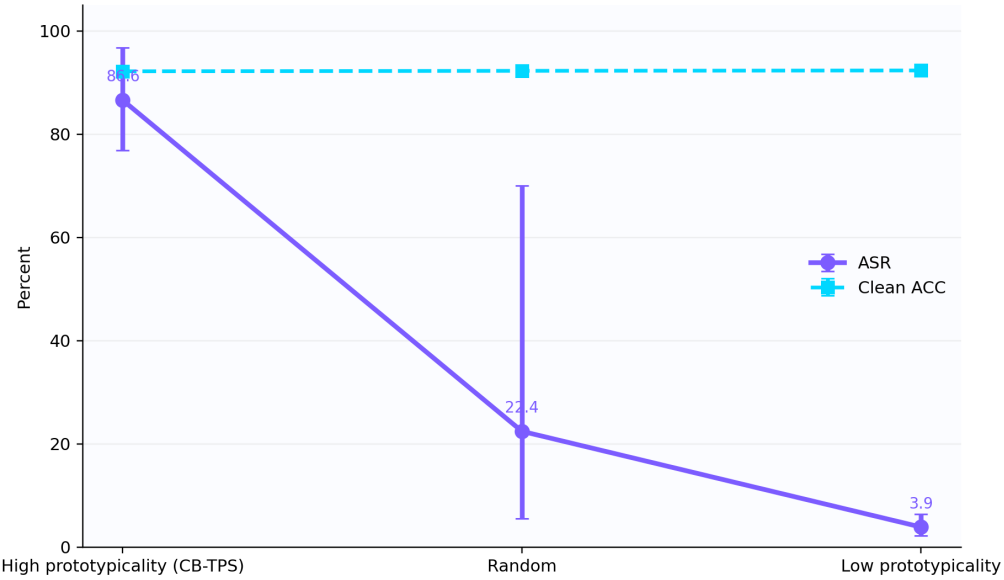


Fig. 3.10 Prototype-ranking ablation over ten seeds. Markers show mean ASR and mean clean accuracy; vertical bars show the corresponding minimum–maximum range across runs. The large ASR decline from high prototypicality to random and low prototypicality occurs while clean accuracy remains approximately constant.

Table 3.4 Prototype-ranking ablation with fixed class quotas. Values are means over ten seeds.

Within-class rule	n	Score order	ASR (%) ↑	Clean accuracy (%) ↑
High prototypicality (CB-TPS)	10	largest s_i	**86.56**	92.16
Random	10	score-independent	22.42	92.25
Low prototypicality	10	smallest s_i	3.87	92.32

The random condition in this ablation is an independently generated within-class random control for the prototype-ranking experiment. It therefore uses a different set of randomly sampled indices from the Random baseline reported in Table 3.2; the two random means are not expected to be numerically identical and are used for different comparisons.

At the final epoch-6 scoring checkpoint used for this prototype-ranking study, high-prototypicality selection reaches a mean ASR of 86.56%, exceeding class-balanced random by 64.14 percentage points and low-prototypicality selection by 82.69 points. The three mean clean accuracies span only 0.16 percentage points. Thus, neither fixed class quotas nor clean-accuracy degradation explains the ASR ordering. The result provides supporting evidence for RQ1: the score contains useful within-class ranking information, and selecting

higher-prototypicality candidates is substantially more effective than selecting from the opposite end of the same ranking.

The low-prototypicality condition is particularly valuable because it preserves almost the entire pipeline. It uses the same scoring network, trigger, transformation, prototypes, class quotas, number of selected indices, and victim training; only the order is reversed. Its ASR near 4% shows that the presence of prototype construction alone does not create the main outcome. The direction of the prototype-affinity ranking is therefore a major determinant of the observed outcome.

Random selection has a higher mean ASR than the low-prototypicality condition. This ordering is consistent with random sampling occasionally including moderately prototypical candidates, whereas low-prototypicality selection systematically favours candidates at the opposite end of the source-class ranking. The PCA display supports the intended geometry for one source class, but the outcome table is stronger because selection and evaluation occur in the full feature and victim spaces. Together the two figures connect implementation behaviour with final performance without treating a projection as proof.

The contrast among all three ranking directions gives the ablation more information than a high-versus-random comparison alone. Random selection establishes that the score contains useful information, while the low-prototypicality condition tests whether the sign of that information agrees with the proposed hypothesis. If both extremes had produced similar ASR, prototype affinity could have been merely a correlate with little directional value. Instead, performance decreases sharply as selection moves from the high-affinity end to random and then to the deliberately reversed low-affinity end. Within the evaluated configuration, this ordered outcome is consistent with prototypicality being an informative ranking axis rather than an arbitrary partition of the candidate population.

3.9.8 Scoring-Epoch Sensitivity

The scoring epoch E_s changes the representation learned by the clean scoring model. For each evaluated checkpoint, features, whitening statistics, prototypes, selected indices, and final victim results are recomputed. The tested checkpoints are epochs 4, 6, and 10, with the high-prototypicality CB-TPS rule and all remaining settings held fixed.

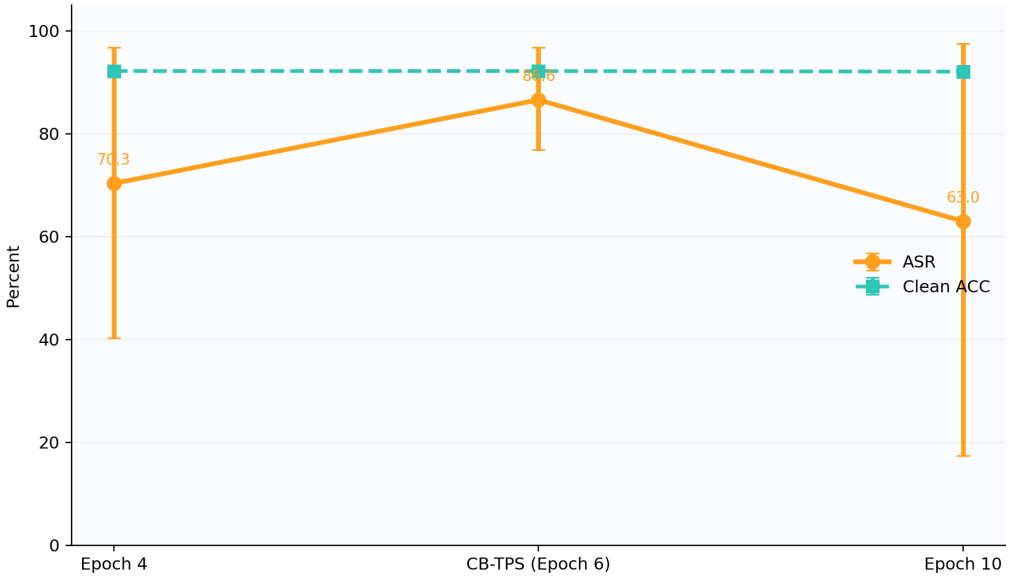


Fig. 3.11 Scoring-epoch sensitivity over ten seeds. Markers show mean ASR and mean clean accuracy for scoring checkpoints at epochs 4, 6, and 10; vertical bars show the corresponding minimum–maximum range across runs.

Table 3.5 Scoring-epoch sensitivity. Values are means over ten seeds.

Scoring epoch E_s	n	ASR (%) ↑	Clean accuracy (%) ↑
Epoch 4	10	70.33	92.17
Epoch 6 (final)	10	**86.56**	92.16
Epoch 10	10	62.96	92.05

Epoch 6 provides the highest mean ASR among the three tested checkpoints: 86.56%, compared with 70.33% at epoch 4 and 62.96% at epoch 10. Mean clean accuracy changes by only 0.12 percentage points across the comparison. Epoch 6 is therefore adopted as the development configuration; because the same experiment is used both to compare checkpoints and choose that setting, the result is not treated as an independently confirmed optimum. The

choice applies only to the three tested checkpoints under the present configuration (CIFAR-10, ResNet-18, target class 0, the fixed patch, and the stated scoring-model schedule). A different dataset, architecture, trigger, or training schedule may favour a different checkpoint, which should then be chosen again on a separate validation setting.

The checkpoint pattern is consistent with an intermediate-representation hypothesis. At epoch 4, source-class structure in the penultimate-layer representation may still be insufficiently developed for stable prototype estimation. By epoch 6, the scoring model appears to provide a more informative trigger-conditioned feature space. At epoch 10, additional clean-task specialisation does not improve the resulting candidate ranking and the mean ASR falls. These explanations remain hypotheses because the experiment measures outcomes rather than representation quality directly.

The sensitivity result also provides a practical warning against treating the scoring model as a passive implementation detail. CB-TPS relies on the geometry of an early representation, so changing the checkpoint changes the object being used to define both whitening and source-class centres. The non-monotonic ASR pattern therefore indicates that "more trained" is not equivalent to "better for selection." The experiment does not identify a universal optimal epoch, but it shows that the auxiliary representation is part of the operating configuration and should be reported explicitly in any reproduction or extension of the method.

3.9.9 Reproducibility and Validity Checks

For each run, the implementation records the method or ablation condition, trigger, target, poisoning budget, scoring checkpoint, seed, selected indices, selected counts by source class, scoring and victim configurations, ASR, clean accuracy, device, and elapsed-time boundary. Before evaluation, selected indices are checked for uniqueness, exclusion of target-class training samples, and consistency with the specified class quotas and total budget. The trigger coordinates and RGB values are held identical across scoring, poisoned-dataset construction, and ASR evaluation, and the victim is independently initialised rather than continued from the scoring model.

Two limitations are particularly relevant to interpretation. First, the final victim remains stochastic, which motivates evaluation over ten independent seeds. Second, the prototype estimator is transductive: each source-class mean is computed from the complete class-specific candidate population, so every candidate contributes to the centre used in its score. This introduces no test-set information and, as shown in Section 3.5, does not change the within-class ranking. An independently estimated prototype population would instead test transfer of the source-class centre itself. External validity is further limited by the single dataset, architecture, target class, and fixed trigger used in the primary study.

Together, these records make the selection stage auditable at the level of concrete indices rather than only aggregate ASR. A reproduction can reconstruct the candidate population, recompute the source-class prototypes and scores, verify the class quotas, and confirm that the saved selected indices are the indices passed to victim training. The same record structure is used for the clean-input versus trigger-conditioned and ranking ablations, so changes in the final subset can be traced to the representation or ordering rule that was varied. The budget, checkpoint, and runtime studies reuse the same configuration records while changing the stated operating quantity.

3.10 Discussion

3.10.1 Principal Findings

The primary comparison in Section 3.9 provides positive evidence for RQ1: under the fixed $K = 25$ budget, CB-TPS produces a substantially higher mean ASR than the evaluated RD, FUS, and random alternatives, while clean accuracy remains closely matched. The discussion therefore concentrates on what the controlled ablations and sensitivity studies add to that endpoint comparison rather than repeating the complete result table.

The two matched ablations provide supporting evidence within RQ1 by separating the two operations that define the CB-TPS selection signal. One changes whether source-class geometry is constructed from clean or triggered inputs; the other keeps the triggered geometry fixed and changes which part of the within-class prototype-affinity ranking is selected. The discussion below therefore focuses on what these chapter-specific experiments support about the CB-TPS selection principle, while keeping the interpretation limited to the evaluated CIFAR-10 target-0 configuration.

3.10.2 Evidence for Trigger-Conditioned Source-Class Prototypicality

The first ablation replaces the trigger-conditioned feature representation $G(T(x_i))$ with the clean representation $G(x_i)$ while retaining the remaining prototype, class-quota, ranking, and victim-training procedure. Mean ASR decreases from 86.56% to 57.30%, whereas mean clean accuracy changes by only 0.14 percentage points. Clean-space source-class structure is therefore informative, but it does not reproduce the effectiveness obtained when the same trigger used for poisoning is applied before feature extraction. This result supports the first component of the hypothesis: candidate utility depends on the representation in which source-class structure is measured.

The second ablation tests the ordering induced by the source-class prototypicality score. Selecting the highest-scoring candidates produces 86.56% mean ASR, compared with 22.42% for an independently generated class-balanced random control and 3.87% when candidates are selected from the low-prototypicality end of the same ranking. Because the trigger, representation, prototype construction, class quotas, poisoning budget, and victim-training procedure are otherwise held fixed, the large high-versus-low separation shows that proximity to the trigger-conditioned source-class prototype contains useful within-class ranking information in this setting.

Taken together, the two ablations provide complementary evidence for the proposed selection principle. First, trigger conditioning adds information beyond clean-space prototypicality. Second, within the trigger-conditioned feature space, selecting candidates that are more prototypical of their source class is markedly more effective than selecting candidates that are less prototypical. Class balance alone cannot explain this result because the random and reversed controls use the same source-class quotas.

These findings are consistent with the dirty-label motivation developed in Chapter 1. Each poisoned example combines source-class semantics with the same trigger and is assigned the same target label. Under an ultra-low budget, unusual properties of a small number of source images can occupy a large fraction of the poisoned evidence. Selecting high-prototypicality candidates may reduce this sample-specific variation, leaving the fixed trigger as a more consistent shared input factor across source classes. The present experiments support this interpretation indirectly through downstream ASR and the two controlled ablations; they do not constitute a direct causal measurement of what the victim internally learns.

The score itself should also be interpreted precisely. Since each transformed candidate z_i has unit norm but the prototype μ_c is not re-normalised, the implemented quantity $s_i = z_i^\top \mu_c$ is a prototype-affinity score rather than a calibrated probability or formal cosine similarity. Substituting the class mean shows that the same quantity is the candidate's average dot-product agreement with all transformed triggered samples from its source class. The criterion therefore measures population centrality in the transformed feature geometry rather than closeness to a separately chosen exemplar. Within a source class, $\|\mu_c\|_2$ is constant across candidates, so the induced ranking is equivalent to cosine-similarity ranking with the normalised source-class prototype. Across classes, however, prototype norms can differ because the transformed feature populations can have different directional concentration. This is one reason the method deliberately performs ranking within source class instead of interpreting raw affinity values as globally calibrated scores.

3.10.3 Relation to Existing Selection Signals

The empirical findings position trigger-conditioned source-class prototypicality as a different source of candidate information from the signals emphasised by the main baselines. FUS derives utility from forgetting behaviour observed during training [62, 49]. Its strength is that the signal is connected directly to optimisation dynamics, but obtaining that signal requires repeated model interaction. RD instead evaluates candidates through their relationship to an early target-oriented output [60]. This is computationally lighter, but the score is expressed in prediction space and asks a different question from source-class representativeness. CB-TPS does not attempt to approximate either signal. It asks whether the triggered candidate remains aligned with the central direction of its own source class in a frozen feature space.

The distinction matters because "useful sample" is not a representation-free concept. A forgetting-based selection method favours samples according to how they behave as training proceeds; a target-distance selection method favours samples according to a particular output relation; and a prototype-based selection method favours samples according to a class-conditional geometric relation. None of these definitions is universally correct independent of the attack objective. The present results show that the source-class geometric criterion is highly informative in the evaluated dirty-label setting, not that source-class prototypicality should replace all other data-valuation criteria. Indeed, combining complementary signals is a plausible direction for future work, but it would require a new controlled study because improvements could arise from either the additional information or the added optimisation.

The findings also distinguish CB-TPS from benign coreset selection. General subset selection methods may seek examples that preserve a training distribution, match aggregate gradients, or represent the diversity of a clean dataset [25, 41]. CB-TPS has a narrower objective. It does not attempt to construct a miniature clean training set. It uses the source class only as a conditional reference and then selects a handful of observations that will be relabelled to a common target after the same trigger is inserted. The value of a central sample in this setting therefore derives from its role in a maliciously relabelled, trigger-conditioned subset, not from a general claim that central samples are always the most useful examples for learning.

3.10.4 Efficiency and Sensitivity

The runtime comparison in Section 3.9 places CB-TPS between score-independent random sampling and the more expensive learned baselines. Random selection remains cheapest because it does not train a scoring model or estimate a selection criterion, whereas CB-TPS requires an auxiliary scoring model and population-wide feature extraction. Within

the recorded timing environment, however, its selection overhead is markedly below the evaluated RD and FUS procedures. The result is therefore best read as an effectiveness–cost trade-off rather than as a claim that CB-TPS is the minimum-cost selection method.

This trade-off is important because selection cost is paid before any final victim benefit is known. An expensive selection method may be justified if it consistently reduces the number of poisoned examples required, but that computation should be counted as part of the attack procedure rather than treated as free preprocessing. Its selection operations are largely batched forward passes and linear-algebraic transformations after the early model has been trained. The recorded timing therefore supports the practical claim that a feature-geometric signal can occupy a middle ground between score-independent sampling and repeated training-based selection.

The scoring-checkpoint experiment further shows that the feature representation is a material design choice. Mean ASR is 70.33% at epoch 4, 86.56% at epoch 6, and 62.96% at epoch 10. More scoring-model training therefore does not monotonically improve selection. One plausible explanation is that very early features provide insufficient source-class structure for stable prototype estimation, whereas later features become increasingly specialised to the clean classification task in ways that are less useful for trigger-conditioned ranking. The experiment measures checkpoint sensitivity rather than the internal cause of this non-monotonic pattern, so representation diagnostics across training would be required to test that explanation directly. Epoch 6 should therefore be read as the best of the three checkpoints tested in this configuration, not as a universally optimal choice.

Increasing the poisoning budget from 25 to 30 and 50 samples raises mean ASR from approximately 86.6% to 94.2% and 98.1%, respectively. This trend is consistent with increased exposure to the trigger–target association. However, changing the budget also changes the number and source-class composition of the poisoned observations. The experiment therefore establishes budget sensitivity for CB-TPS within the tested range rather than isolating a general scaling law.

The diminishing absolute improvement as the budget grows is also consistent with a ceiling effect: once ASR is already high, additional poisoned examples have less room to improve the endpoint metric. This means the three reported budgets should not be used to infer a linear relationship between poison count and attack success. More importantly, the sensitivity experiment does not answer whether the advantage of prototypicality over competing selection methods narrows or widens as K changes, because the baselines are not re-evaluated at the additional budgets. A complete scaling study would require matched cross-method comparisons at each budget.

3.10.5 Implications for Direct Ultra-Low-Budget Selection

The results suggest that ultra-low-budget selection benefits from separating "representative of the source class" from "close to the target." In a dirty-label poison, the original visual semantics do not disappear when the label is changed. The poisoned observation still presents the victim with source-dependent content, a shared trigger, and a target label. A source-class criterion explicitly models one part of this composite structure rather than collapsing candidate utility to the target prediction alone. The strong high-versus-low ordering indicates that this source-side information can be useful even though the final malicious objective is target-side behaviour.

A second implication concerns the timing of data valuation. CB-TPS obtains its selection signal from a frozen early clean model, before the final victim is trained. This indicates that a useful ordering can be present in pre-poisoning representation geometry and need not always be discovered through repeated poisoned-model optimisation. That observation is important for computationally constrained attack studies, but it should not be interpreted as evidence that training dynamics are unimportant. The FUS comparison shows only that the evaluated CB-TPS procedure produces a better cost–effectiveness outcome in this configuration; other datasets, triggers, architectures, or budgets may favour different information sources.

A third implication is methodological. When only a handful of samples are modified, the selection rule should be evaluated with controls that preserve the final budget and allocation. A comparison against unrestricted random poisoning can confound the value of a score with the class composition of the subset. By using class-balanced random and reversed-ranking controls, the present study isolates the within-class ordering more directly. This separation between allocation and ranking may also be useful when evaluating future selection methods whose scores have different scales across classes.

3.10.6 Alternative Explanations and Remaining Questions

Several alternative explanations remain plausible. First, high-prototypicality samples may also be easier examples or more stable under data augmentation even without the trigger. The clean-representation ablation partly addresses this possibility: clean-space prototypicality is useful, but it does not reproduce the performance of the trigger-conditioned representation. Measuring score stability under the exact crop and flip augmentations used during victim training would help determine whether augmentation robustness contributes to the effect.

Second, the fixed source-class quotas may interact with the prototypicality score. The within-class random condition of the ranking ablation shows that quotas alone do not explain the final ASR, but the current experiments do not compare class-balanced ranking with an

unconstrained global Top-K rule. Consequently, the evidence supports the prototypicality criterion under fixed class-balanced allocation; it does not establish that approximately equal source-class allocation is itself optimal.

Third, prototype estimation is transductive because the complete non-target training population is used both to define the class centre and to provide the candidates that are ranked against it. This does not involve test data, and, as derived in Eq. (3.15), the direct self-contribution leaves the within-class ranking unchanged. The transductive design therefore does not obtain its Top-K_c ordering merely from a candidate matching itself. A genuinely disjoint prototype population would still be a useful additional experiment, but it would test a different question: whether the source-class centre estimated from one subset transfers to candidates outside that population.

Fourth, a single mean prototype may be inadequate for multimodal source classes. CIFAR-10 classes contain substantial pose and appearance variation, and the limitation is likely to become more pronounced on fine-grained or higher-resolution datasets. Multiple local prototypes or mixture-based reference models are natural extensions, but they would change the definition of the selection criterion and should therefore be evaluated explicitly rather than treated as implementation variants.

3.10.7 Method-Specific Limitations

The principal limitation is external validity. The primary evidence uses one dataset, one architecture family, one target class, and one visible 5×5 patch. The method has therefore not yet been shown to generalise across target labels, architectures, larger datasets, invisible triggers, clean-label attacks, or defence-aware settings. The use of ResNet-18 for both the scoring model and victim also leaves open how well the selected subset transfers when the scoring and victim architectures differ.

Representation dependence is a central part of this limitation. The class centre is defined in the feature space of a particular early scoring network, so a different architecture can change both local geometry and candidate ordering even when the underlying images are unchanged. Cross-architecture transfer would therefore test whether high-prototypicality selections reflect a property of the data that persists across feature extractors or a property specific to one ResNet-18 representation. Similarly, evaluating multiple triggers would test whether the method captures a general principle of trigger-conditioned geometry or a pattern that is unusually favourable for the fixed patch used here.

A second limitation concerns development and confirmation. Epoch 6 is selected from the same target-0 development study that reports epochs 4, 6, and 10. Its strong performance supports the chosen configuration, but an independent validation setting would be needed

to determine how well the checkpoint transfers without retuning. Seed-level outcomes are retained for the repeated experiments, but the present thesis emphasises descriptive run-level means and minimum–maximum trajectory envelopes rather than a pre-specified confirmatory inferential analysis. A future study could use the retained matched-seed records to add such analysis under a separately defined validation protocol.

A third limitation is that the current study does not report per-source-class ASR or prototype coherence. Count-balanced selection controls only the number of selected samples per source class. Class-wise attack success may still differ because source classes can have different feature structure and different susceptibility to the fixed target and trigger. Reporting these diagnostics would clarify whether the global ASR is broadly supported across source classes or dominated by a subset of them.

A fourth limitation is that the current experiments evaluate the prototype criterion primarily through downstream ASR. They do not directly quantify intra-class dispersion, prototype stability, or the degree to which the selected high-affinity samples reduce source-content variability relative to random subsets. Such diagnostics are not necessary to establish that the ranking is empirically useful, but they would strengthen the proposed interpretation and help identify conditions under which a single class mean ceases to be an adequate summary.

Chapter 4

Redundancy-Aware Secondary Selection

Research arising from the RASS study presented in this chapter has also resulted in the paper *Redundancy-Aware Secondary Selection for Data-Efficient Backdoor Attacks*, accepted for publication at PRICAI 2026. The chapter presents the thesis-level formulation, evaluation, and discussion of the method.

4.1 Method Overview and Independent Problem Setting

Chapter 3 presents CB-TPS as a complete direct selection method that maps the eligible non-target population to a final poisoned subset. RASS addresses a different input setting and is developed as an independent method rather than as a continuation of CB-TPS. It assumes that an external selection method has already produced a small scored candidate pool. Within that pool, individual candidate quality is not the only consideration: several high-ranked candidates can occupy similar regions of the trigger-conditioned representation space, so a direct Top-K rule can allocate a severe poisoning budget to repeatedly represented feature directions.

The thesis therefore introduces *Redundancy-Aware Secondary Selection* (RASS) as a standalone candidate-pool refinement method. RASS does not replace the external pool-generating selection method and does not define a new full-dataset importance score. It receives candidate indices, the corresponding first-stage importance values, and candidate-aligned feature representations. It preserves the external importance signal while using a neighbourhood graph in the trigger-conditioned representation space to modify which candidates remain competitive as the final subset is assembled. In the reported experiments, the incoming pools are generated independently by RD and FUS; CB-TPS is not used to generate RASS inputs.

The two methods developed in this thesis consequently solve different selection formulations. CB-TPS asks *which samples should be selected directly from the complete non-target population?* RASS asks *given a small scored candidate pool from an external selection method, which candidates should remain in the final K-sample subset?* CB-TPS uses trigger-conditioned source-class prototypicality for direct selection; RASS preserves externally supplied importance while reducing local redundancy among trigger-conditioned representations.

4.2 Secondary-Selection Problem Formulation

Let

$$D = \{(x_i, y_i)\}_{i=1}^{N} \tag{4.1}$$

denote the clean training set, let t be the attacker-specified target label, and let $T(\cdot)$ be the fixed trigger transformation. For any selected clean subset S, the corresponding poisoned examples are

$$P(S) = \{(T(x_i), t) : (x_i, y_i) \in S\}, \tag{4.2}$$

and the victim is trained on the clean data with the selected observations replaced by their poisoned versions. Exhaustively evaluating all K-subsets would require repeated victim training for a combinatorial number of candidate combinations. RASS instead operates before victim training and refines a fixed external pool using first-stage importance together with feature-level redundancy.

Suppose an existing selection method A_M returns a candidate pool

$$\mathscr{B} = A_M(D), \qquad |\mathscr{B}| = M, \tag{4.3}$$

where each candidate $i \in \mathscr{B}$ has an externally supplied importance score q_i. The reported formulation assumes that the incoming pool contains at least K candidates with strictly positive raw importance, which is satisfied in all experiments reported in this thesis. Under this eligibility condition, the positive-importance mask used by the implementation does not reduce the requested final cardinality. The notation $\mathscr{B}$ distinguishes the small RASS candidate pool from the complete non-target candidate population used by CB-TPS. The secondary-selection task is to choose

$$A \subset \mathscr{B}, \qquad |A| = K, \qquad K < M, \tag{4.4}$$

while retaining high-importance candidates and avoiding unnecessary concentration of the final subset in locally redundant trigger-conditioned feature neighbourhoods.

Figure 4.1 provides the first-stage context used by the RASS experiments. An external selection method assigns an importance score to each training sample and retains a small top-M candidate pool. This stage supplies the candidate indices and incoming scores to RASS; it is not part of the RASS refinement rule itself.

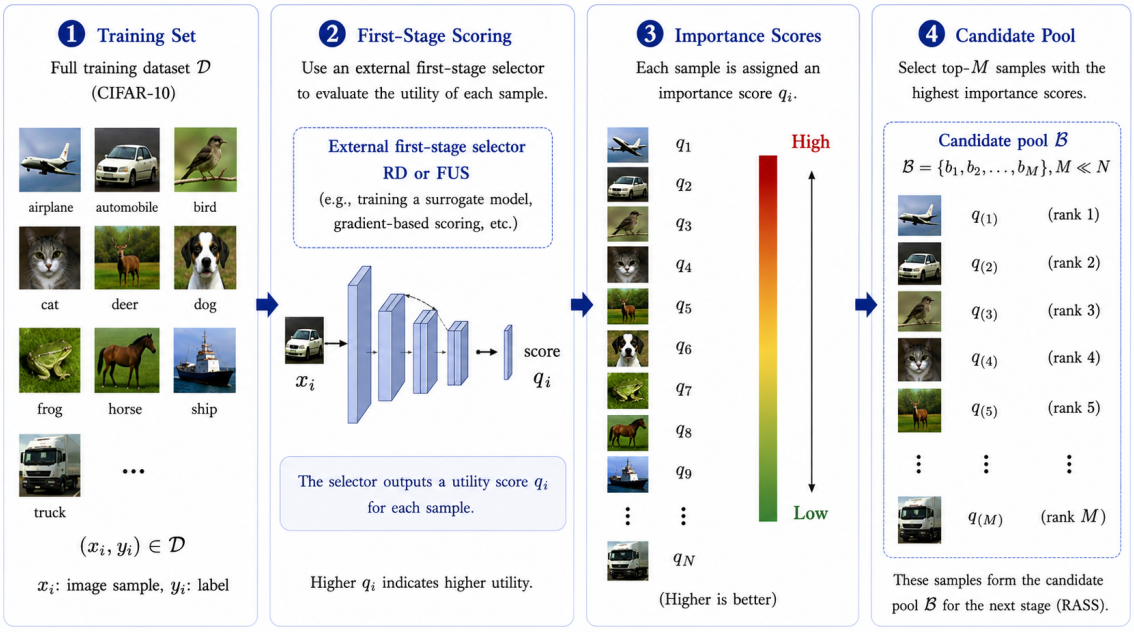


Fig. 4.1 Conceptual first-stage candidate-pool construction used before RASS. An external selection method evaluates the training samples, assigns importance scores q_i, and retains the top-M candidates as the scored pool $\mathscr{B}$. In the reported experiments, this external stage is instantiated by RD or FUS; RASS begins from the resulting candidate pool rather than from the complete training set. AI-assisted schematic; see the Artificial Intelligence Declaration for tool and prompt details.

4.3 First-Stage Importance Normalisation

The RASS graph sampler assumes that the incoming score vector has already been oriented so that larger values represent greater first-stage importance. This is the interface used for the RD- and FUS-based candidate pools in the reported experiments. RASS then applies min–max normalisation inside the current candidate pool:

$$I_i = \frac{q_i - \min_{j \in \mathscr{B}} q_j}{\max_{j \in \mathscr{B}} q_j - \min_{j \in \mathscr{B}} q_j + \varepsilon}, \qquad i \in \mathscr{B}, \tag{4.5}$$

where $\varepsilon > 0$ prevents division by zero. Thus $I_i \in [0,1]$ under the default normalisation, and the transformation preserves the ordering of the supplied scores.

This transformation is local to the current pool. RASS neither interprets q_i as a calibrated probability nor compares the numerical scale of RD scores with that of FUS scores. The implementation also applies a positive-importance eligibility mask before greedy selection, so candidates whose raw incoming importance is non-positive are not selected. In the default configuration no additional class quota is imposed by the RASS sampler itself.

4.4 Triggered-Feature Redundancy and Neighbourhood Graph

For each candidate $i \in \mathscr{B}$, let h_i^t denote the candidate-aligned trigger-conditioned feature representation supplied to the RASS graph sampler. The RASS operator itself does not retrain a feature extractor at this stage; it consumes the feature array provided by the upstream experimental pipeline in the same order as the candidate indices and importance scores. If the supplied feature tensor has more than two dimensions, the implementation flattens it to a vector before pairwise comparison.

The full pairwise redundancy matrix is constructed from positive cosine similarity. The implementation first normalises each supplied feature vector using a norm floor,

$$\widetilde{h}_i^t = \frac{h_i^t}{\max(\|h_i^t\|_2, \varepsilon)}, \qquad \varepsilon > 0, \tag{4.6}$$

and then computes

$$R_{ij} = \max\left(0, (\widetilde{h}_i^t)^\top \widetilde{h}_j^t\right), \qquad i \neq j, \qquad R_{ii} = 0. \tag{4.7}$$

Thus the code forms a dot-product matrix after row-wise ℓ_2 normalisation, clips negative similarities to zero, and removes the diagonal. Hence $R_{ij} \in [0,1]$. The constant ε is used to floor the feature norm before division rather than being added to the product of two norms. Negative cosine similarity is not rewarded as diversity; it simply contributes no positive redundancy.

The redundancy matrix and the graph used for selection are distinct objects. After R is computed, the sampler constructs a directed neighbourhood set $\mathscr{N}_i$ for each candidate. Under the implemented default `redundancy_scope=topk`, $\mathscr{N}_i$ contains at most r candidates with the largest positive R_{ij} values, where the default is $r = 10$. If the configured scope is `all`, then every other candidate is treated as a neighbour. The reported method description follows

the default neighbour-restricted graph, so only edges in $\mathcal{N}_i$ contribute to the graph-density prior and to subsequent suppression.

This distinction is important. RASS does not maximise global dissimilarity across all pairs. It uses triggered-feature redundancy to define a local graph around each candidate, while the external importance score remains the primary estimate of candidate utility. Figure 4.2 summarises the second-stage information used by RASS: the fixed trigger is applied to the candidates to obtain aligned trigger-conditioned representations, the incoming first-stage scores are normalised, and pairwise positive cosine similarity expresses feature redundancy.

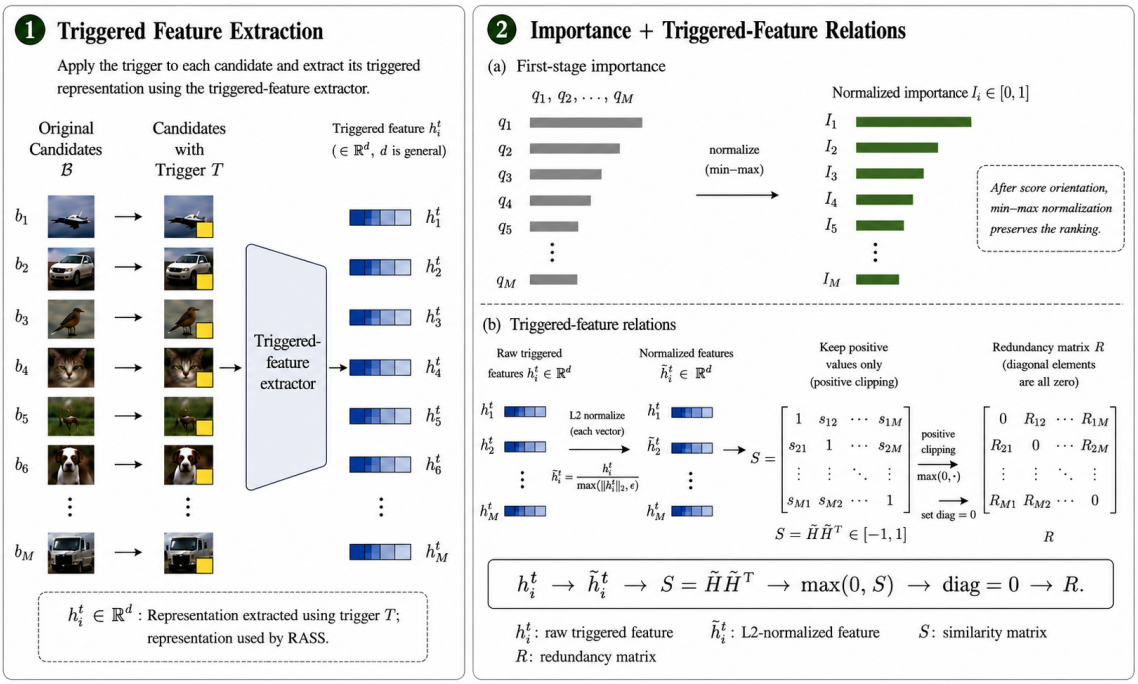


Fig. 4.2 Second-stage information construction for RASS. For each member of the external candidate pool, the trigger-conditioned representation h_i^t is obtained and aligned with the candidate's incoming first-stage score q_i. The scores are normalised to I_i, while positive cosine similarity R_{ij} describes pairwise redundancy in the trigger-conditioned representation space. These quantities provide the inputs to the graph-density prior and greedy neighbourhood suppression defined in the following sections. AI-assisted schematic; see the Artificial Intelligence Declaration for tool and prompt details.

4.5 Graph-Density Prior

For each candidate, the graph sampler first aggregates importance-weighted support from its configured redundancy neighbourhood:

$$\rho_i = \sum_{j \in \mathcal{N}_i} R_{ij} I_j. \tag{4.8}$$

With the implemented additive graph mode, the initial graph-supported score is

$$d_i = I_i + \lambda_g \rho_i, \tag{4.9}$$

where $\lambda_g \geq 0$ controls the contribution of neighbourhood support. The first term preserves the candidate's own first-stage importance. The second term increases the starting priority of candidates located near other candidates that are both positively similar in the trigger-conditioned feature space and important according to the external selection method.

This prior has a different role from the later redundancy penalty. It is not a request to select dense regions indiscriminately; rather, it uses local support as an initial reliability term before sequential suppression starts. The graph sampler also contains alternative implementation modes, but the thesis evaluates the additive `density_suppression` formulation described here.

4.6 Greedy Neighbourhood Suppression

RASS initialises the current score as

$$c_i \leftarrow d_i. \tag{4.10}$$

Let A denote the selected set, initially $A = \varnothing$. At each iteration the eligible unselected candidate with the highest current score is chosen:

$$i^* = \arg\max_{i \in \mathcal{B} \setminus A} c_i. \tag{4.11}$$

After i^* is added to A, suppression is applied only to its currently available graph neighbours. For $j \in \mathcal{N}_{i^*} \cap (\mathcal{B} \setminus A)$,

$$c_j \leftarrow c_j - \lambda_s R_{i^* j} \max(c_{i^*}, \varepsilon_s), \tag{4.12}$$

where $\lambda_s \geq 0$ is the suppression coefficient and the implementation uses a small positive floor ε_s to avoid a vanishing multiplicative term. Candidates outside $\mathcal{N}_{i^*}$ are unchanged at that iteration.

The update is soft rather than eliminative. A redundant neighbour is not permanently removed; its current priority is reduced in proportion to its positive cosine redundancy with the newly selected sample and to the score of that selected sample. If the penalised candidate still has a sufficiently high current score, it may be selected later. The graph topology is fixed for the secondary-selection run, whereas the candidate scores evolve after each selection.

The two coefficients therefore have separate roles. λ_g controls the amount of neighbourhood support present in the initial score, while λ_s controls how strongly selected candidates reduce the later priority of their graph neighbours. A larger λ_g does not mean stronger suppression.

4.7 Complete RASS Procedure

Algorithm 2 Redundancy-Aware Secondary Selection (RASS)

Input: candidate pool $\mathscr{B}$; higher-is-better first-stage scores $\{q_i\}_{i\in\mathscr{B}}$; trigger-conditioned features $\{h_i^t\}_{i\in\mathscr{B}}$; final budget K; graph neighbourhood size r; coefficients λ_g and λ_s; constants $\varepsilon, \varepsilon_s > 0$.
Output: final selected set A with $|A| = K$ under the stated eligibility condition.

Min–max normalise the incoming importance scores using Eq. (4.5).
Set the eligible set to candidates with positive raw importance.
Normalise each trigger-conditioned feature representation using Eq. (4.6).
Compute the full positive-cosine redundancy matrix R using Eq. (4.7).
For each i, construct $\mathscr{N}_i$ from the top-r positive redundancy values (default $r = 10$).
Compute $\rho_i \leftarrow \sum_{j\in\mathscr{N}_i} R_{ij} I_j$ and $d_i \leftarrow I_i + \lambda_g \rho_i$.
Initialise $c_i \leftarrow d_i$ and $A \leftarrow \varnothing$.
for $k = 1, \ldots, K$ **do**
    $i^* \leftarrow \arg\max$ current score among eligible candidates not in A.
    $A \leftarrow A \cup \{i^*\}$.
    for each $j \in \mathscr{N}_{i^*} \cap (\mathscr{B} \setminus A)$ **do**
        $c_j \leftarrow c_j - \lambda_s R_{i^* j} \max(c_{i^*}, \varepsilon_s)$.
    end for
    $c_{i^*} \leftarrow -\infty$.
end for
return A.

In the primary RASS experiments, $M = 30$ and $K = 25$. The refinement therefore removes only five candidates from the externally generated pool. This is a demanding secondary-selection setting because most of the first-stage pool must be retained; RASS has limited

freedom to improve the final subset and must identify which small number of candidates should lose priority once local redundancy is considered.

4.8 Computational Characteristics and Scope

For a pool of size M and feature dimension d, constructing the full positive-cosine redundancy matrix costs $O(M^2 d)$. Building the top-r neighbourhood graph requires ranking redundancy values within each candidate row, while each greedy iteration still requires an $O(M)$ search for the largest current score and updates at most r neighbours under the default top-r scope. The resulting secondary stage is small in the evaluated $M = 30$ configuration.

RASS is conditional on the information supplied by the upstream pipeline. It cannot recover a useful candidate sample that is absent from the incoming candidate pool, and the graph sampler itself does not determine how the upstream trigger-conditioned feature representation is learned. Its contribution is the secondary decision rule applied after candidate indices, first-stage scores, and aligned feature vectors have been provided. The experimental evaluation in Section 4.9 applies this rule to RD- and FUS-generated pools and compares it with direct Top-K and random-within-pool selection under the same final poisoning budget.

4.9 Experimental Evaluation

This section evaluates RASS independently as a candidate-pool refinement method. RD or FUS first produces a scored candidate pool of $M = 30$ samples, after which the pool is reduced to the fixed final budget $K = 25$ by direct Top-K, random-within-pool selection, or RASS. A separate coefficient study examines the graph-density term. The evaluation uses five seeds and is interpreted within this candidate-pool setting rather than as a direct numerical comparison with the CB-TPS full-population evaluation.

4.9.1 Evaluation Protocol

The RASS experiments evaluate the independent candidate-pool refinement problem defined in Chapter 4. The dataset, target, trigger, and final victim model remain CIFAR-10 [27], target class 0, the fixed 5×5 yellow BadNets-style patch, and ResNet-18 [17]. Final victim training uses 60 epochs of SGD with learning rate 0.05 and batch size 256. For the RD-based candidate pool, the scoring model is also ResNet-18 and is trained for six epochs before the first-stage scores are computed. The FUS-based pool uses the same final victim-training procedure.

The key difference from the CB-TPS suite is the selection interface. The first-stage method produces a pool of

$$M = 30$$

scored candidates, and every secondary-selection method must return exactly

$$K = 25$$

final poisoned samples. The comparison therefore holds both the incoming candidate pool and final poison count fixed within each first-stage family. RASS uses the code-default `density_suppression` objective with pool-wise min–max importance normalisation, triggered-feature redundancy, additive graph scoring, and $\lambda_g = 1$, $\lambda_s = 1$ unless otherwise stated. The sampler first constructs the full positive-cosine redundancy matrix over the $M = 30$ pool and then restricts graph support and suppression to the configured top-r neighbourhoods; the implementation default is $r = 10$.

These values are chosen as follows. $K = 25$ matches the primary ultra-low budget of Chapter 3, so both methods are studied at the same final poison count. $M = 30$ keeps the pool only slightly larger than K; RASS therefore removes five candidates, which tests whether redundancy-aware refinement helps even when it can change only a small part of a strong first-stage pool. $r = 10$ restricts each candidate's graph to one third of the pool, so that support and suppression act on local neighbours rather than on the whole pool. $\lambda_s = 1$ and $\lambda_g = 1$ are the implementation defaults; only λ_g is varied, in Section 4.9.3. Refinement performance for substantially larger candidate pools and other suppression strengths remains unverified.

All RASS experiments are repeated over five seeds, using seeds 0–4. For each seed, the first-stage candidate-pool construction, secondary selection when applicable, victim initialisation, data shuffling, augmentation, and final poisoned training are re-run. The reported RASS means are therefore five-run aggregates and, as in Chapter 3, are reported descriptively without statistical significance claims; the shaded regions in the trajectory figures show the spread across seeds.

Three secondary-selection rules are compared for each first-stage pool. *Direct Top-K* keeps the 25 candidates with the best original first-stage scores. *Random-within-pool* uniformly samples 25 candidates from the same 30-sample pool. *RASS* applies the graph-supported importance score and greedy triggered-feature redundancy suppression defined in Chapter 4. An auxiliary full-dataset random reference is also retained to show the attack level recorded when 25 samples are selected uniformly from the full CIFAR-10 training set without first-stage screening. Because this reference is not constrained to the same candidate-pool

interface as the principal RASS comparisons, it is reported for context only and is not treated as a threat-model-matched comparator for RQ2. The evidence for RQ2 is based on direct Top-K, random-within-pool, and RASS applied to the same RD- or FUS-generated candidate pool.

4.9.2 Main Results

Table 4.1 reports the secondary-selection results under the common final budget of 25 poisoned samples. The value $M = 30$ denotes the first-stage candidate-pool size rather than the number of poisoned samples used for victim training.

Table 4.1 RASS secondary-selection results on CIFAR-10 with the final poisoning budget fixed at $K = 25$. Values are mean ASR over five seeds (0–4).

Method	First-stage pool	Final samples	ASR (%) ↑
Auxiliary random (full CIFAR-10)	–	25	17.49
RD direct Top-25	30	25	39.34
Random subset from RD pool	30	25	23.82
RASS on RD pool	30	25	**54.59**
FUS direct Top-25	30	25	20.72
Random subset from FUS pool	30	25	7.61
RASS on FUS pool	30	25	**40.34**

Figures 4.3 and 4.4 provide a common epoch-wise view of the principal RASS comparisons across both RD- and FUS-generated candidate pools. The ASR trajectories show how the advantage of RASS emerges during the shared 60-epoch victim-training schedule, while the clean-accuracy trajectories verify that the differences in attack effectiveness are not accompanied by a corresponding separation in benign classification performance.

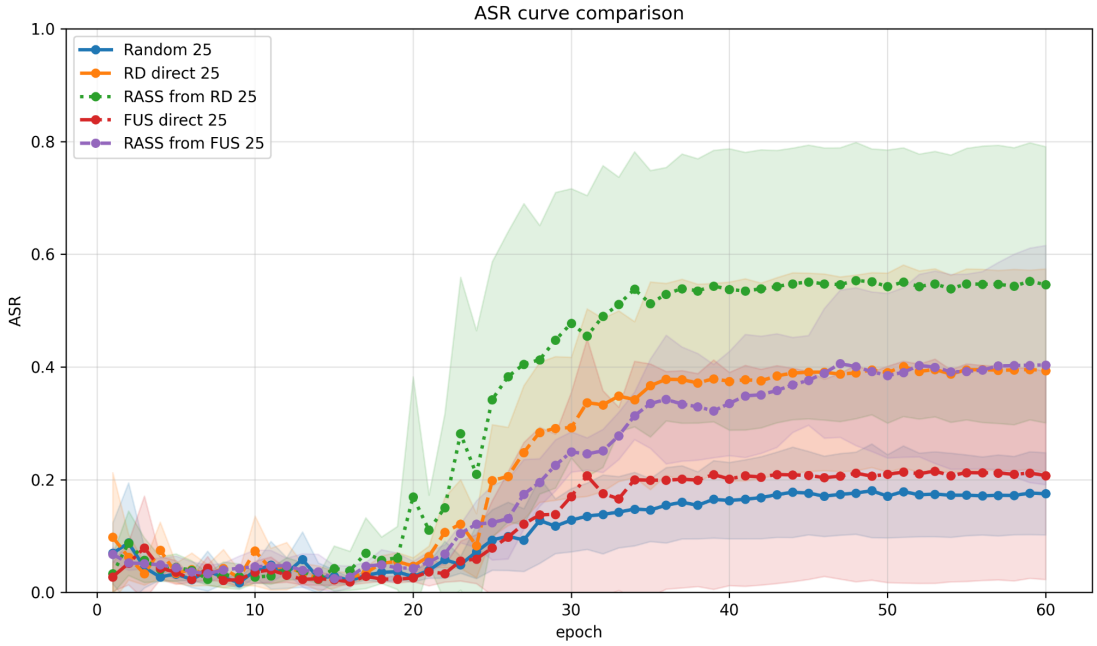


Fig. 4.3 Epoch-wise ASR trajectories for the main RASS comparison. The figure shows the auxiliary full-dataset random reference, direct RD Top-25, RASS refinement of the RD candidate pool, direct FUS Top-25, and RASS refinement of the FUS candidate pool. The final-epoch ordering is consistent with Table 4.1: RASS improves the final poisoned subset for both externally generated candidate pools without increasing the final budget $K = 25$. Shaded regions show the epoch-wise minimum–maximum range across the five repeated runs, using the same min–max convention as the CB-TPS trajectory figure.

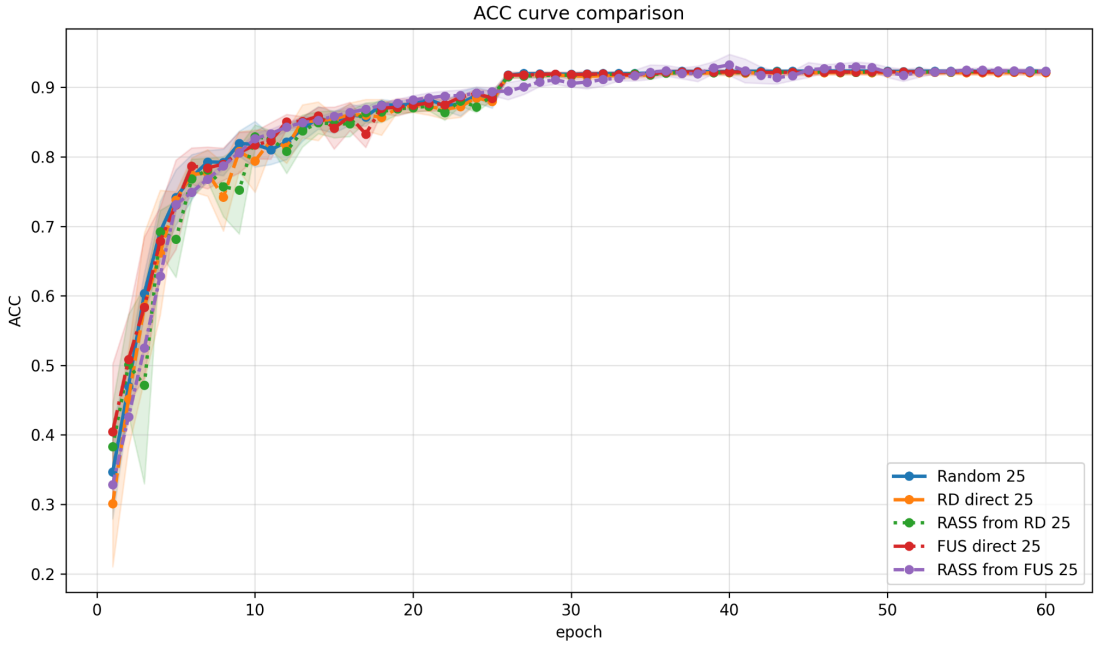


Fig. 4.4 Epoch-wise clean-accuracy trajectories for the same five conditions shown in Figure 4.3. Despite their different ASR trajectories, all methods converge to closely matched clean accuracy, indicating that the RASS gains are associated with stronger backdoor effectiveness rather than a systematic degradation of benign classification. Shaded regions are defined as in Figure 4.3.

For the RD candidate pool, direct first-stage Top-25 selection achieves 39.34% mean ASR. RASS raises this result to 54.59%, an improvement of 15.25 percentage points without changing either the 30-sample candidate pool or the final 25-sample poison budget. Randomly sampling 25 candidates from the same RD pool reaches 23.82% ASR, showing that the presence of an RD-screened candidate pool alone does not explain the RASS improvement.

The same pattern appears with a different first-stage signal. Direct FUS Top-25 selection achieves 20.72% mean ASR, whereas RASS on the same 30-sample FUS pool reaches 40.34%, an improvement of 19.62 percentage points. Random selection from the FUS pool reaches 7.61%. The fact that RASS improves both RD- and FUS-generated pools is important because the incoming scores are based on different notions of candidate utility. RASS does not require those raw scores to share a common scale; it only preserves their within-pool importance after orientation and normalisation.

These comparisons answer RQ2 within the evaluated secondary-selection setting. The improvement cannot be attributed to a larger final poisoning budget, because all candidate-pool methods use exactly 25 final poisoned samples. It also cannot be attributed only to the first-stage pool, because random-within-pool selection performs substantially below RASS

for both RD and FUS. The remaining difference is the rule used to compose the final subset from the same available candidates.

The main RASS experiment should not be compared numerically with Table 3.2 as though both tables were generated from a single shared run protocol. Table 3.2 evaluates complete direct-selection procedures over ten runs, whereas Table 4.1 evaluates five-seed candidate-pool refinement after an $M = 30$ pool has been generated by RD or FUS. The RASS results establish the value of refinement inside a fixed external pool; they do not provide a direct comparison with CB-TPS because the two methods solve different selection formulations and use different experimental protocols.

4.9.3 Graph-Density Ablation

The graph-density coefficient λ_g controls the neighbourhood-support term used to initialise the RASS score before greedy redundancy suppression. To examine its role, the RD candidate pool is evaluated with

$$\lambda_g \in \{0, 0.5, 1, 2\},$$

while the final poisoning budget remains 25 and $\lambda_s = 1$ is held fixed.

Table 4.2 Ablation on the RASS graph-density coefficient λ_g under the RD candidate pool. Values are means over five seeds.

λ_g	Clean accuracy (%) ↑	ASR (%) ↑
0	92.08	46.09
0.5	92.32	51.91
1	92.24	**54.59**
2	92.21	33.48

Figure 4.5 complements the endpoint ablation by showing how the four graph-density settings separate over victim training. This trajectory view is useful because it shows whether the final ordering in Table 4.2 is a late-epoch fluctuation or a persistent difference that emerges during optimisation.

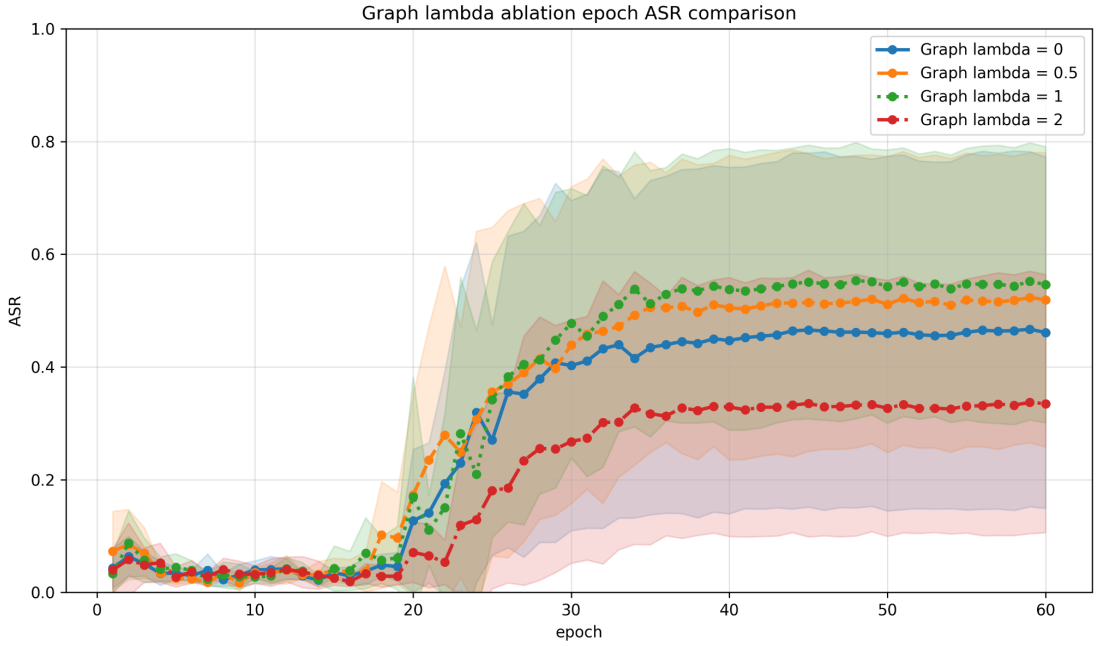


Fig. 4.5 Epoch-wise ASR trajectories for the RASS graph-density ablation on the RD candidate pool. The four curves correspond to $\lambda_g \in \{0, 0.5, 1, 2\}$ with $\lambda_s = 1$ and the final poisoning budget fixed at $K = 25$. The shaded regions show the epoch-wise minimum–maximum range across the five repeated runs, using the same min–max convention as the CB-TPS and main RASS trajectory figures.

The $\lambda_g = 0$ condition is informative because it removes graph-density support while leaving the neighbour-restricted greedy redundancy-suppression stage active at $\lambda_s = 1$. It reaches 46.09% mean ASR, compared with 39.34% for direct RD Top-25 under the same five-seed candidate-pool protocol. Thus, importance-preserving sequential suppression over the trigger-conditioned feature graph provides a 6.75-percentage-point gain over direct first-stage Top-K before any graph-density support is added. Introducing moderate neighbourhood support increases mean ASR further to 51.91% at $\lambda_g = 0.5$ and 54.59% at $\lambda_g = 1$, an additional 8.50-percentage-point gain over the suppression-only $\lambda_g = 0$ condition. Clean accuracy remains close to 92% across these settings. This decomposition does not replace a dedicated λ_s sensitivity study, but it separates the evidence for sequential suppression from the additional contribution of the graph-density prior within the evaluated RD pool.

Increasing λ_g to 2.0 causes a substantial ASR decline to 33.48%. This result should not be interpreted as evidence that stronger redundancy suppression is harmful, because suppression strength is controlled by λ_s rather than λ_g. A large λ_g instead gives excessive weight to the initial graph-density prior, favouring candidates embedded in dense trigger-conditioned feature neighbourhoods before the sequential complementarity penalty is applied. The

ablation therefore supports a balanced use of neighbourhood support rather than maximising the density term itself.

4.10 Discussion

4.10.1 Principal Findings

The main comparisons in Section 4.9 provide positive evidence for RQ2 in the evaluated candidate-pool setting. With the incoming RD or FUS pool and the final poison count fixed, RASS produces a stronger mean ASR than both direct Top-K retention and random-within-pool selection. The discussion therefore focuses on what the refinement rule changes about subset composition rather than repeating the endpoint table.

4.10.2 Redundancy-Aware Refinement and Subset Complementarity

The RASS results highlight the role of subset composition within the candidate-pool refinement setting. A direct first-stage Top-K rule assumes that the marginal value of one candidate is independent of which other candidates have already been retained. The secondary-selection experiments show that this assumption is not always adequate under the $M = 30$ to $K = 25$ setting. On both RD and FUS pools, the RASS subset produces a higher mean ASR than direct Top-25 selection even though the incoming pool and final poison count are unchanged.

The random-within-pool controls are important for interpreting this result. An RD candidate pool contains samples that have already passed a first-stage utility filter, but randomly retaining 25 of the 30 candidates reaches only 23.82% mean ASR. The same control on the FUS pool reaches 7.61%. Thus, simply constraining the search to a small first-stage pool is not sufficient. The secondary rule still determines which five candidates are excluded and, equivalently, which patterns of trigger-conditioned feature similarity remain represented in the final subset.

RASS combines two quantities with different roles. The normalised first-stage score preserves the utility information provided by RD or FUS, while the full positive-cosine trigger-conditioned feature-similarity matrix is used to construct a neighbourhood graph over the candidate pool. The graph-density term initially favours candidates supported by nearby high-importance examples, and the greedy suppression term subsequently reduces the marginal priority of available neighbours that are redundant with a sample already selected. The RD-pool ablation in Section 4.9.3 separates these effects: suppression alone ($\lambda_g = 0$) already improves on direct RD Top-25, and restoring the graph-density prior at $\lambda_g = 1$ improves ASR further. This pattern provides evidence that both stages contribute in the

evaluated RD pool, while the decline at $\lambda_g = 2$ shows that dense-neighbourhood preference should not dominate the original importance signal.

The RASS evidence does not establish that diversity in the trigger-conditioned feature space is universally beneficial. The method is evaluated only for two first-stage candidate-pool families, one pool size, one final budget, and one default suppression coefficient. The empirical contribution is narrower and more precise: under the evaluated RD and FUS pools, accounting for pairwise triggered-feature redundancy improves the composition of the final 25-sample poisoned subset relative to direct Top-K and random-within-pool selection.

4.10.3 Method-Specific Limitations

The RASS evaluation has additional scope constraints. Secondary refinement is tested only with RD and FUS candidate pools, $M = 30$, $K = 25$, five seeds, and the default $\lambda_s = 1$. The candidate-pool size is central to interpreting these results. With $K/M = 25/30$, RASS only decides which five candidates to drop, so the measured gains describe refinement of a pool that is already close to the final budget. The current results do not establish how the redundancy rule behaves when M is much larger than K, where the first-stage ordering constrains the final subset less and redundancy could play a larger or a different role. The main coefficient ablation varies λ_g but does not independently map the sensitivity of λ_s. In addition, RASS depends on first-stage recall: once a useful candidate sample is absent from the incoming pool, the secondary-selection method has no mechanism for recovering it.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This thesis investigated poisoning-sample selection under an ultra-low-budget dirty-label backdoor setting through two independent methods, each corresponding to one research question. RQ1 asked whether trigger-conditioned source-class prototypicality can improve direct selection from the complete eligible non-target population. Chapter 3 addressed this question through Class-Balanced Triggered Prototypicality Selection (CB-TPS), which uses a frozen early clean model to construct trigger-conditioned source-class geometry and ranks candidates within class by prototype affinity. Under the primary CIFAR-10 target-0 setting with $K = 25$, CB-TPS attains 86.56% mean ASR with 92.16% mean clean accuracy across ten runs, while the matched RD, random, and FUS baselines obtain substantially lower mean ASR under the same downstream victim-training protocol. The representation and ranking ablations further support the proposed selection principle: applying the trigger before constructing source-class geometry and selecting from the high-prototypicality end of that geometry are both important within the evaluated configuration. These results provide positive evidence for RQ1.

RQ2 asked whether a small scored candidate pool can be improved by considering redundancy among candidates before the final poisoning budget is fixed. Chapter 4 addressed this question through Redundancy-Aware Secondary Selection (RASS), which retains externally supplied first-stage importance while modifying subset composition according to redundancy among trigger-conditioned candidate representations. In the separate five-seed $M = 30$ to $K = 25$ evaluation, RASS improves the final subset produced from both RD- and FUS-generated candidate pools relative to direct Top-K and random-within-pool alternatives. The graph-density study further shows that neighbourhood support can affect refinement quality, while the fixed-pool comparisons isolate the contribution of the secondary selection

rule from the quality of the first-stage selection method. These results provide positive evidence for RQ2 within the evaluated candidate-pool setting.

Taken together, the thesis contributes two distinct formulations for ultra-low-budget poisoning-sample selection. CB-TPS addresses *candidate valuation* before victim training by using trigger-conditioned source-class prototypicality for direct population-level selection. RASS addresses *subset composition* after an external selection method has already produced a small candidate pool by preserving first-stage importance while suppressing redundant selections. The results show that poison count alone does not determine attack effectiveness: under a severe budget, which samples are selected and how a final subset is composed can materially change downstream ASR while clean classification remains closely matched. The two methods are intentionally evaluated independently and should not be interpreted as sequential stages of one proposed pipeline.

These results are demonstrated on CIFAR-10 with a fixed target class, a visible patch trigger, and ResNet-18. Transfer to other datasets, architectures, targets, triggers, or defence settings is potential rather than demonstrated, and has not been tested in this thesis.

5.2 Future Work

For CB-TPS, confirmation should precede further method expansion. The current scoring checkpoint, feature definition, whitening procedure, source-class score, and quota rule should first be frozen and evaluated across new target labels, datasets, trigger families, and scoring/victim architectures. Cross-architecture tests would clarify whether the selected samples capture a transferable property of poisoned data or a feature peculiarity of one ResNet-18 scoring model. Disjoint or cross-fitted prototype estimation, class-wise ASR, prototype-coherence diagnostics, and multi-prototype variants would then test the limits of the single-centre formulation without mixing basic external validation with new modelling choices.

For RASS, validation should vary candidate-pool size, final budget, neighbourhood size, first-stage selection method, and both graph-density and suppression coefficients. Because the current candidate pool is close to the final budget, larger gaps between M and K would provide a stronger test of whether triggered-feature redundancy remains useful when the refinement stage has more freedom. Additional representation-level diagnostics could test whether gains are consistently associated with reduced redundancy or with other structural properties of the selected subset.

Finally, a combined CB-TPS–RASS pipeline could be studied as a separate method, but it would require its own controlled experimental design. The present thesis keeps the methods

independent so that RQ1 isolates direct candidate valuation and RQ2 isolates secondary subset refinement. A future integrated study could test whether a CB-TPS-generated candidate pool benefits from RASS-style refinement, while comparing the integrated system against each component under identical candidate-pool and final budget constraints. Such an experiment would extend, rather than retroactively alter, the two-method structure evaluated in this thesis.

References

[1] Abad, G., Franch, E., Koffas, S., and Picek, S. (2026). Removing the trigger, not the backdoor: Alternative triggers and latent backdoors. *arXiv preprint arXiv:2603.09772*.

[2] Aghakhani, H., Meng, D., Wang, Y.-X., Kruegel, C., and Vigna, G. (2021). Bullseye polytope: A scalable clean-label poisoning attack with improved transferability. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 159–178.

[3] Bagdasaryan, E. and Shmatikov, V. (2021). Blind backdoors in deep learning models. In *30th USENIX Security Symposium*, pages 1505–1521.

[4] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 1597–1607. PMLR.

[5] Chen, X., Liu, C., Li, B., Lu, K., and Song, D. (2017). Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*.

[6] Coleman, C., Yeh, C., Mussmann, S., Mirzasoleiman, B., Bailis, P., Liang, P., Leskovec, J., and Zaharia, M. (2020). Selection via proxy: Efficient data selection for deep learning. In *International Conference on Learning Representations*.

[7] Fisher, M. L., Nemhauser, G. L., and Wolsey, L. A. (1978). An analysis of approximations for maximizing submodular set functions—II. In *Polyhedral Combinatorics*, volume 8 of *Mathematical Programming Studies*, pages 73–87. Springer.

[8] Gao, Y., Li, Y., Zhu, L., Wu, D., Jiang, Y., and Xia, S.-T. (2023). Not all samples are born equal: Towards effective clean-label backdoor attacks. *Pattern Recognition*, 139:109512.

[9] Gao, Y., Xu, C., Wang, D., Chen, S., Ranasinghe, D. C., and Nepal, S. (2019). STRIP: A defence against trojan attacks on deep neural networks. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pages 113–125.

[10] Garg, S., Kumar, A., Goel, V., and Liang, Y. (2020). Can adversarial weight perturbations inject neural backdoors? In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*, pages 2029–2032.

[11] Garrido-Lucero, F., Heymann, B., Vono, M., Loiseau, P., and Perchet, V. (2024). DU-Shapley: A Shapley value proxy for efficient dataset valuation. In *Advances in Neural Information Processing Systems*, volume 37.

[12] Ghorbani, A. and Zou, J. (2019). Data Shapley: Equitable valuation of data for machine learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 2242–2251. PMLR.

[13] Gu, T., Liu, K., Dolan-Gavitt, B., and Garg, S. (2019). Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244.

[14] Guo, W., Wang, L., Xu, Y., Xing, X., Du, M., and Song, D. (2020). Towards inspecting and eliminating trojan backdoors in deep neural networks. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 162–171.

[15] Hayase, J. and Oh, S. (2023). Few-shot backdoor attacks via neural tangent kernels. In *International Conference on Learning Representations*.

[16] He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9726–9735.

[17] He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

[18] He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Identity mappings in deep residual networks. In *Computer Vision – ECCV 2016*, pages 630–645.

[19] He, P., Xing, Y., Xu, H., Ren, J., Cui, Y., Zeng, S., Tang, J., Yamada, M., and Sabokrou, M. (2024). Stealthy backdoor attack via confidence-driven sampling. *Transactions on Machine Learning Research*.

[20] Huang, B. and Wong, N. (2025). Poisoning-based backdoor attacks for arbitrary target label with positive triggers. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, pages 1134–1142.

[21] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 448–456. PMLR.

[22] Jia, J., Liu, Y., and Gong, N. Z. (2022). Badencoder: Backdoor attacks to pre-trained encoders in self-supervised learning. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 2043–2059.

[23] Jolliffe, I. T. and Cadima, J. (2016). Principal component analysis: A review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202.

[24] Kessy, A., Lewin, A., and Strimmer, K. (2018). Optimal whitening and decorrelation. *The American Statistician*, 72(4):309–314.

[25] Killamsetty, K., Sivasubramanian, D., Ramakrishnan, G., De, A., and Iyer, R. (2021a). GRAD-MATCH: Gradient matching based data subset selection for efficient deep model training. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 5464–5474. PMLR.

[26] Killamsetty, K., Sivasubramanian, D., Ramakrishnan, G., and Iyer, R. (2021b). GLISTER: Generalization based data subset selection for efficient and robust learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8110–8118.

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

[28] Li, Y., Jiang, Y., Li, Z., and Xia, S.-T. (2024a). Backdoor learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(1):5–22.

[29] Li, Y., Lyu, X., Koren, N., Lyu, L., Li, B., and Ma, X. (2021a). Anti-backdoor learning: Training clean models on poisoned data. In *Advances in Neural Information Processing Systems*, volume 34, pages 14900–14912.

[30] Li, Y., Lyu, X., Koren, N., Lyu, L., Li, B., and Ma, X. (2021b). Neural attention distillation: Erasing backdoor triggers from deep neural networks. In *International Conference on Learning Representations*.

[31] Li, Z., Sun, H., Xia, P., Xia, B., Rui, X., Zhang, W., Guo, Q., Fu, Z., and Li, B. (2024b). A proxy attack-free strategy for practically improving the poisoning efficiency in backdoor attacks. *IEEE Transactions on Information Forensics and Security*, 19:9730–9743.

[32] Li, Z., Xia, P., Sun, H., Zeng, Y., Zhang, W., and Li, B. (2025). Explore the effect of data selection on poison efficiency in backdoor attacks. *IEEE Transactions on Dependable and Secure Computing*, 22(6):7430–7447.

[33] Lin, M., Chen, Q., and Yan, S. (2014). Network in network. In *International Conference on Learning Representations*.

[34] Liu, K., Dolan-Gavitt, B., and Garg, S. (2018). Fine-pruning: Defending against backdooring attacks on deep neural networks. In *Research in Attacks, Intrusions, and Defenses*, pages 273–294.

[35] Liu, Y., Ma, X., Bailey, J., and Lu, F. (2020). Reflection backdoor: A natural backdoor attack on deep neural networks. In *Computer Vision – ECCV 2020*, pages 182–199. Springer.

[36] Loshchilov, I. and Hutter, F. (2017). SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*.

[37] Mirzasoleiman, B., Bilmes, J., and Leskovec, J. (2020). Coresets for data-efficient training of machine learning models. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 6950–6960. PMLR.

[38] Nemhauser, G. L., Wolsey, L. A., and Fisher, M. L. (1978). An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming*, 14:265–294.

[39] Nguyen, A. T. and Tran, A. T. (2021). WaNet: Imperceptible warping-based backdoor attack. In *International Conference on Learning Representations*.

[40] Nguyen, T. A. and Tran, T. A. (2020). Input-aware dynamic backdoor attack. In *Advances in Neural Information Processing Systems*, volume 33, pages 3454–3464.

[41] Paul, M., Ganguli, S., and Dziugaite, G. K. (2021). Deep learning on a data diet: Finding important examples early in training. In *Advances in Neural Information Processing Systems*, volume 34, pages 20596–20607.

[42] Saha, A., Subramanya, A., and Pirsiavash, H. (2020). Hidden trigger backdoor attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11957–11965.

[43] Shafahi, A., Huang, W. R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., and Goldstein, T. (2018). Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems*, volume 31.

[44] Shen, G., Liu, Y., Tao, G., An, S., Xu, Q., Cheng, S., Ma, S., and Zhang, X. (2021). Backdoor scanning for deep neural networks through K-Arm optimization. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 9525–9536. PMLR.

[45] Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.

[46] Snell, J., Swersky, K., and Zemel, R. (2017). Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, volume 30.

[47] Souri, H., Fowl, L., Chellappa, R., Goldblum, M., and Goldstein, T. (2022). Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch. In *Advances in Neural Information Processing Systems*, volume 35.

[48] Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pages 1139–1147. PMLR.

[49] Toneva, M., Sordoni, A., des Combes, R. T., Trischler, A., Bengio, Y., and Gordon, G. J. (2019). An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*.

[50] Tran, B., Li, J., and Madry, A. (2018). Spectral signatures in backdoor attacks. In *Advances in Neural Information Processing Systems*, volume 31, pages 8011–8021.

[51] Turner, A., Tsipras, D., and Madry, A. (2019). Label-consistent backdoor attacks. *arXiv preprint arXiv:1912.02771*.

[52] Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., and Zhao, B. Y. (2019). Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 707–723.

[53] Wang, K., Zhao, B., Peng, X., Zhu, Z., Yang, S., Wang, S., Huang, G., Bilen, H., Wang, X., and You, Y. (2022a). Cafe: Learning to condense dataset by aligning features. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12186–12195.

[54] Wang, T., Yao, Y., Xu, F., An, S., Tong, H., and Wang, T. (2021). Backdoor attack through frequency domain. *arXiv preprint arXiv:2111.10991*.

[55] Wang, Z., Ding, H., Zhai, J., and Ma, S. (2022b). Training with more confidence: Mitigating injected and natural backdoors during training. In *Advances in Neural Information Processing Systems*, volume 35, pages 36396–36410.

[56] Wei, Q., He, S., Zhang, J., Feng, L., and An, B. (2025). Influence-based fair selection for sample-discriminative backdoor attack. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 21474–21481.

[57] Weinberger, K. Q. and Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(9):207–244.

[58] Wu, B., Chen, H., Zhang, M., Zhu, Z., Wei, S., Yuan, D., and Shen, C. (2022). Backdoorbench: A comprehensive benchmark of backdoor learning. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, volume 35, pages 10546–10559.

[59] Wu, D. and Wang, Y. (2021). Adversarial neuron pruning purifies backdoored deep models. In *Advances in Neural Information Processing Systems*, volume 34, pages 16913–16925.

[60] Wu, Y., Han, X., Qiu, H., and Zhang, T. (2023). Computation and data efficient backdoor attacks. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4782–4791.

[61] Wu, Z., Lu, Y., Wen, J., Sun, H., Zhou, Q., and Lu, G. (2025). A set of generalized components to achieve effective poison-only clean-label backdoor attacks with collaborative sample selection and triggers. In *Advances in Neural Information Processing Systems*, volume 38.

[62] Xia, P., Li, Z., Zhang, W., and Li, B. (2022). Data-efficient backdoor attacks. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, pages 3992–3998.

[63] Xu, X., Wang, Q., Li, H., Borisov, N., Gunter, C. A., and Li, B. (2021). Detecting AI trojans using meta neural analysis. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 103–120.

[64] Xun, Y., Jia, X., Gu, J., Liu, X., Guo, Q., and Cao, X. (2024). Minimalism is king! high-frequency energy-based screening for data-efficient backdoor attacks. *IEEE Transactions on Information Forensics and Security*, 19:4560–4571.

[65] Yoon, J., Arik, S. O., and Pfister, T. (2020). Data valuation using reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 10842–10851. PMLR.

[66] Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. In *Proceedings of the British Machine Vision Conference*.

[67] Zeng, Y., Chen, S., Park, W., Mao, Z. M., Jin, M., and Jia, R. (2022). Adversarial unlearning of backdoors via implicit hypergradient. In *International Conference on Learning Representations*.

[68] Zeng, Y., Park, W., Mao, Z. M., and Jia, R. (2021). Rethinking the backdoor attacks' triggers: A frequency perspective. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16453–16461.

[69] Zhang, K., Cheng, S., Shen, G., Tao, G., An, S., Makur, A., Ma, S., and Zhang, X. (2024). Exploring the orthogonality and linearity of backdoor attacks. In *2024 IEEE Symposium on Security and Privacy*.

[70] Zhao, B. and Bilen, H. (2021). Dataset condensation with differentiable Siamese augmentation. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 12674–12685. PMLR.

[71] Zhao, B., Mopuri, K. R., and Bilen, H. (2021). Dataset condensation with gradient matching. In *International Conference on Learning Representations*.

[72] Zhao, Z., Chen, X., Xuan, Y., Dong, Y., Wang, D., and Liang, K. (2022). Defeat: Deep hidden feature backdoor attacks by imperceptible perturbation and latent representation constraints. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15192–15201.

[73] Zheng, R., Tang, R., Li, J., and Liu, L. (2022). Data-free backdoor removal based on channel Lipschitzness. In *Computer Vision – ECCV 2022*, pages 175–191. Springer.

[74] Zhu, Z., Zhang, M., Wei, S., Shen, L., Fan, Y., and Wu, B. (2023). Boosting backdoor attack with a learnable poisoning sample selection strategy. *arXiv preprint arXiv:2307.07328*.

Appendix A

Method Verification, Reproducibility, and Statistical Reporting

A.1 Method-to-Implementation Specification

Table A.1 summarises the implementation quantities corresponding to the mathematical definition of CB-TPS.

Table A.1 Implementation specification of the final CB-TPS configuration.

Component	Implementation	Manuscript reference
Candidate set	all non-target indices $y_i \neq t$	Section 1.1
Triggered input	apply the fixed trigger T before feature extraction	Eq. (3.4)
Feature representation	$u_i = G(T(x_i)) \in \mathbb{R}^{512}$, the penultimate-layer feature representation	Eq. (3.5)
Preprocessing	shared whitening followed by ℓ_2 normalisation	Eqs. (3.8)–(3.9)
Source-class prototype	mean transformed representation of all non-target candidates in the corresponding source class	Eq. (3.10)
Prototypicality score	$s_i = z_i^\top \mu_{y_i}$	Eq. (3.11)
Budget allocation	deterministic approximately equal source-class quotas K_c	Eq. (3.18)
Selection	largest K_c scores independently within each source class	Eq. (3.19)

A.2 Ablation Definitions

Trigger-conditioned versus clean-input feature representation. The clean-input control replaces $G(T(x_i))$ with $G(x_i)$ while keeping the scoring checkpoint, whitening *procedure*, prototype-construction rule, class quotas, ranking rule, and downstream victim-training protocol unchanged. Whitening statistics and source-class prototypes are recomputed separately from the feature population associated with each representation condition; fitted triggered-space statistics are not reused for the clean-input control.

Source-class prototypicality ranking. The ranking ablation uses the same trigger-conditioned feature representation and source-class quotas but selects either the largest scores, uniformly random candidates within each class, or the smallest scores.

Scoring checkpoint. The checkpoint study changes only the epoch at which the clean scoring model is frozen. Feature extraction, whitening, prototype estimation, sample selection, and victim training are repeated for each checkpoint.

A.3 Reproducibility Records

For each run, the following information is retained:

- dataset version and split, target class, trigger type, trigger location, trigger size, and poisoning budget;
- method or ablation condition, random seed, selected indices, original source labels, source-class quotas, and realised selected counts;
- scoring-model and victim architecture, checkpoints, optimiser, learning rate and schedule, momentum, weight decay, batch size, data augmentation, numerical precision, and device;
- penultimate-layer feature specification, candidate indices, class-specific population membership, whitening parameters, source-class prototypes, prototypicality scores, and ranking rule; and
- ASR and clean-accuracy numerators and denominators together with the selection-stage timing boundary.

A.4 Consistency Checks

The implementation verifies that the selected set contains exactly K unique non-target indices, that source-class counts match the quota vector, and that the same trigger transformation is used during scoring, poisoned-dataset construction, and ASR evaluation. The recorded class-specific candidate populations reproduce the source-class prototypes, and the saved scores reproduce $z_i^\top \mu_{y_i}$ within numerical tolerance. The final victim is independently initialised and trained using the selected indices rather than continuing from the scoring-model checkpoint.

Appendix B

Implementation Details and Reproducibility

B.1 Final CB-TPS Configuration

The final Class-Balanced Triggered Prototypicality Selection (CB-TPS) configuration used in the primary experiments is summarised below:

- **Feature representation:** the penultimate-layer feature representation of the frozen clean ResNet-18 scoring model, extracted after global average pooling and immediately before the final linear classifier;
- **Trigger-conditioned feature construction:** $u_i = G(T(x_i)) \in \mathbb{R}^{512}$ for every non-target candidate;
- **Prototype estimation:** one mean transformed representation per non-target source class, estimated from all transformed candidates belonging to that source class;
- **Candidate score:** $s_i = z_i^\top \mu_{y_i}$, evaluated only against the corresponding source-class prototype;
- **Budget allocation:** deterministic approximately equal source-class quotas;
- **Selection rule:** the K_c highest-scoring candidates within each non-target source class;
- **Random seeds:** 0–9 for the reported ten-run studies;
- **Optimisation:** SGD with initial scoring-model and victim learning rates of 0.05, momentum 0.9, and cosine annealing; and

- **Execution:** 60 victim-training epochs, scoring checkpoint epoch 6, batch size 256 for scoring and victim training, eight data-loader workers, and a G4 GPU for the recorded primary configuration.

B.2 Baseline Execution Settings

The RD and FUS runs reported in Chapter 3 use the following recorded settings: CIFAR-10; ResNet-18 for both `model_name` and `score_model_name`; poisoning rate 0.0005; ten repeated runs; 60 final training epochs; ten search iterations; initial ratio 0.3; ratio decay 0.93; victim and scoring batch sizes of 256; eight data-loader workers; CUDA execution; and epoch-wise evaluation.

For FUS, the method-specific mode is `forgetting`. The corresponding command is:

```
python3 diet.py
-diet_mode forgetting
-dataset CIFAR10
-model_name resnet18
-score_model_name resnet18
-perc 0.0005
-repeat 10
-epoch 60
-times 10
-ratio 0.3
-decay 0.93
-batch_size 256
-score_batch_size 256
-num_workers 8
-device cuda
-eval_each_epoch
```

For RD, the method-specific mode is `RD_SCORE` with the same shared search and training settings:

```
python3 diet.py
-diet_mode RD_SCORE
-dataset CIFAR10
-model_name resnet18
-score_model_name resnet18
-perc 0.0005
-repeat 10
-epoch 60
-times 10
-ratio 0.3
-decay 0.93
-batch_size 256
-score_batch_size 256
-num_workers 8
-device cuda
-eval_each_epoch
```

The experiment-name and output-directory arguments used to organise result files do not alter the selection method and are omitted from these formatted command blocks.

B.3 Result Recording

Each experiment records the selected indices and their source labels together with the target class, trigger specification, poisoning budget, class quotas, scoring checkpoint, random seed, scoring-model configuration, victim-training configuration, ASR, clean accuracy, and timing boundary. Retaining the selected indices separates the sample-selection stage from downstream victim training and allows the same poisoned subset to be re-evaluated without re-running the selection method.

For repeated-seed experiments, individual run outcomes are retained before aggregation, and the thesis reports the mean over the specified seed set. Comparisons that require paired inference should use the underlying seed-matched run records rather than reconstructing differences from aggregate means.

B.4 Figure Interpretation

The PCA figure used in the source-class prototypicality ablation (Figure 3.9) is a diagnostic visualisation that combines the ten seed-specific feature sets in one shared cross-seed PCA display. A common two-component projection and common axes are used for the three selection-rule panels, while seed-labelled markers identify the selected samples contributed by each run. All ranking decisions are made from the original full-dimensional transformed representations before projection. The PCA display is therefore used to summarise geometric tendencies across seeds rather than as a selection rule or as independent evidence of attack effectiveness; the corresponding victim ASR results provide the outcome-level evaluation.

The training-trajectory and runtime figures (Figures 3.5, 3.6 and 3.8) likewise provide complementary information rather than replacing final run-level evaluation. Final ASR and clean accuracy are measured from the completed victim training runs, while runtime is interpreted under the recorded hardware and measurement boundary.

Appendix C

RASS Implementation Details and Reproducibility

C.1 Final RASS Configuration

The secondary-selection experiments use the following configuration:

- dataset: CIFAR-10;
- target class: 0;
- trigger: fixed 5×5 yellow patch placed in the lower-right corner with a one-pixel margin;
- victim architecture: ResNet-18;
- final victim training: 60 epochs, SGD, learning rate 0.05, batch size 256;
- first-stage selection methods: RD and FUS;
- first-stage candidate-pool size: $M = 30$;
- final poisoning budget: $K = 25$;
- graph selection objective: `density_suppression`;
- importance normalisation: pool-wise min–max normalisation;
- redundancy feature mode: trigger-conditioned features;
- graph mode: additive (`sum`);
- redundancy scope: top-r positive-similarity neighbours, with the code default $r = 10$;
- RASS graph-density coefficient: $\lambda_g = 1$ unless varied in the ablation;

- RASS redundancy-suppression coefficient: $\lambda_s = 1$;
- no additional RASS class-balance quota in the default sampler;
- eligibility condition: every reported candidate pool contains at least K samples with strictly positive raw incoming importance, so the sampler returns exactly K final candidates; and
- repeated runs: seeds 0–4, with means reported over five runs.

The imported graph sampler is an alias of the implementation class used by the project. Its default graph path uses the `density_suppression` objective, min–max importance normalisation, triggered-feature redundancy, and a top-r neighbourhood scope. These defaults are the implementation form represented by Algorithm 2.

For the RD candidate pool, the scoring model used by the first-stage RD procedure is ResNet-18 trained for six epochs before the incoming first-stage scores are computed. RASS itself begins after the candidate pool, aligned importance scores, and feature arrays have been produced. The graph sampler therefore does not infer a feature layer or retrain a feature extractor internally; it consumes the candidate-aligned feature representations supplied by the caller.

C.2 Secondary-Selection Implementation Interface

The RASS wrapper receives a one-dimensional array of candidate indices, an aligned vector of first-stage importance scores, and feature arrays whose first dimension matches the candidate order. In graph mode the wrapper can pass clean features, trigger-conditioned features, or both to the graph sampler. The primary RASS formulation in this thesis uses the trigger-conditioned feature path. If an incoming feature array has more than two dimensions, the graph sampler flattens the per-candidate representation before similarity calculation.

The sampler first normalises the supplied importance vector within the current pool. It then constructs a complete $M \times M$ redundancy matrix by normalising each trigger-conditioned feature representation as

$$\widetilde{h}_i^t = \frac{h_i^t}{\max(\|h_i^t\|_2, \varepsilon)}, \tag{C.1}$$

forming the matrix of dot products, clipping negative values to zero, and setting the diagonal to zero:

$$R_{ij} = \max\left(0, (\widetilde{h}_i^t)^\top \widetilde{h}_j^t\right), \qquad R_{ii} = 0. \tag{C.2}$$

The norm-floor form above follows the implementation exactly; ε protects the row-wise normalisation from a zero feature norm. The full redundancy matrix is retained for diagnos-

tics, but the default selection graph is neighbour-restricted rather than fully connected. For each candidate, the top-r positive redundancy values define its outgoing neighbourhood; the default is $r = 10$.

With additive graph mode, the starting score used by the primary `density_suppression` objective is

$$d_i = I_i + \lambda_g \sum_{j \in \mathcal{N}_i} R_{ij} I_j. \tag{C.3}$$

The greedy selection procedure then chooses the largest current score among available candidates. Only currently available neighbours of the selected sample are penalised, with $\varepsilon_s = 10^{-8}$:

$$c_j \leftarrow c_j - \lambda_s R_{i^* j} \max(c_{i^*}, \varepsilon_s), \qquad j \in \mathcal{N}_{i^*} \cap (\mathcal{B} \setminus A). \tag{C.4}$$

Thus the implementation does not subtract a redundancy term from every remaining candidate after each step. Both graph support and suppression are mediated by the configured adjacency graph. Candidates with non-positive raw incoming importance are masked from selection before the greedy loop.

The wrapper returns local positions inside the candidate pool and maps them back to the original training-set indices before victim training. The implementation also records the raw and normalised importance scores, the clean and triggered redundancy matrices when available, the used redundancy matrix, adjacency matrix, graph-density values, stepwise greedy scores, suppression records, final selected positions, and mapped training indices. These records permit the secondary-selection decision to be audited independently of downstream victim training.

C.3 Comparison Logic

For each first-stage pool, three final-subset rules are compared under the same $K = 25$ budget. Direct Top-K retains the candidates with the best original first-stage scores. Random-within-pool uniformly samples 25 candidates from the same 30-sample pool. RASS uses the same incoming pool and importance values but composes the final subset using the neighbourhood-supported starting score and greedy neighbour-restricted redundancy suppression described above.