AUTOMATIC ATTACK DISCOVERY FOR FEW-SHOT CLASS-INCREMENTAL LEARNING VIA LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Few-shot class-incremental learning (FSCIL) updates a deployed classifier from only a few examples of each new class, making its training interface unusually sensitive to poisoned support data. We present ACraft, a feedback-guided framework that searches executable clean-label poisoning programs rather than tuning one fixed attack objective. An LLM proposes rationale–code pairs; an FSCIL evaluator measures cross-session damage on a validation split; and a PPO-guided evolutionary controller selects initialization, refinement, or synthesis actions. Our contribution is this FSCIL-grounded program-search formulation and its empirical analysis, rather than a new LLM, evolutionary operator, or PPO algorithm. Across miniImageNet, CIFAR-100, and CUB-200, ACraft discovers dataset-specific programs that degrade several FSCIL learners and transfer from a CNN to a ViT setting. Our primary partial-control setting poisons 50% of current-session support samples and reduces average miniImageNet accuracy by 20.95 points; 25% control still causes a 14.16-point drop, while 75% control causes a 36.07-point drop. We additionally report held-out evaluation, validation-tuned baselines, explicit run provenance, and offline search costs.

1 INTRODUCTION

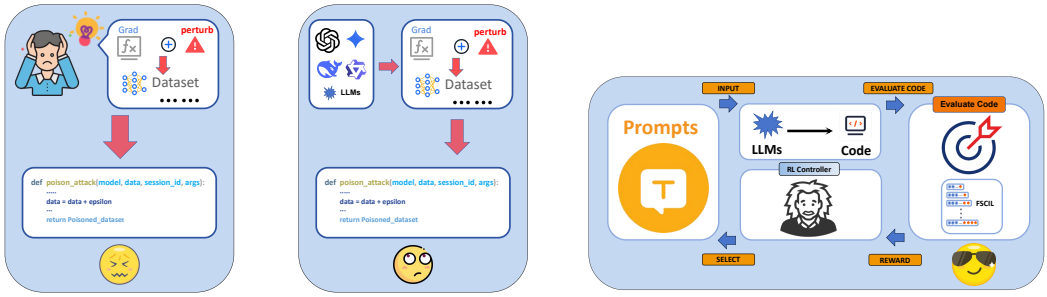
Deep neural networks have demonstrated remarkable success across computer vision applications, but this success commonly assumes that the training and testing categories are fixed after deployment. Real systems often violate this assumption: categories emerge over time, data distributions shift, and deployed models must incorporate new classes without erasing old knowledge. Class-Incremental Learning (CIL) De Lange et al. (2021); Verwimp et al. (2023) addresses this dynamic setting by enabling models to acquire new concepts while retaining previous ones. Few-Shot Class-Incremental Learning (FSCIL) further makes the setting practical by requiring each new class to be learned from only a handful of labeled samples Zhao et al. (2025); Liu & Yang (2025); Chen et al. (2025); Li et al. (2024); Oh et al. (2024b).

Although recent FSCIL methods improve adaptation and reduce catastrophic forgetting, their security risk remains underexplored. The few-shot and sequential nature of FSCIL creates a distinctive attack surface: small perturbations introduced during incremental training can be repeatedly amplified by later updates, while the limited support set makes the model highly sensitive to individual poisoned examples. Existing handcrafted attacks such as FGSM, PGD, C&W, and DeepFool were mainly designed for static image classifiers. When directly applied to FSCIL, they inherit fixed objectives and manually chosen update rules that do not account for evolving decision boundaries, prototype drift, or the asymmetric damage between old and new classes.

1.1 CHALLENGES

Attacking FSCIL is not a straightforward extension of attacking a static classifier. First, a fixed objective must be chosen and retuned when the target protocol, backbone, or session schedule changes; implementing canonical FGSM or PGD itself is not laborious. Second, FSCIL models are non-stationary: feature spaces and classifier boundaries evolve as classes arrive, so a perturbation

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Fixed objective (b) One-shot LLM (c) Feedback search

Figure 1: Attack-design paradigms. (a) A fixed handcrafted objective. (b) One-shot LLM generation without task feedback. (c) ACraft searches executable programs through measured FSCIL feedback.

Table 1: **Program-level comparison under 50% support-set poisoning.** T is the number of attack steps, d the input dimensionality, and $\Pi_{\epsilon,x}$ the common projection defined in Section 4. *Attack Acc* is attacked average accuracy (lower is stronger).

Method	Formula	Searched program	Original target	Attack Acc↓	Cost (Program)
FGSM Goodfellow et al. (2014a)	$x_{adv} = \Pi_{\epsilon,x}(x + \epsilon\,\mathrm{sign}(\nabla_x \mathcal{L}(f(x), y)))$	✗	Static classifier	60.39	$O(d)$
PGD Madry et al. (2017a)	$x_{adv}^{t+1} = \Pi_{\epsilon,x}(x_{adv}^{t} + \alpha\,\mathrm{sign}(\nabla_x \mathcal{L}(f(x_{adv}^{t}), y)))$	✗	Static classifier	60.25	$O(T \cdot d)$
C&W Carlini & Wagner (2017)	$\min_\delta \|\delta\|_2^2 + c\max\{m(x+\delta) + \kappa, 0\}$	✗	Static classifier	61.44	$O(T \cdot d)$
DeepFool Moosavi-Dezfooli et al. (2016)	$\delta^{(t)} = \frac{\lvert f_k(x^{(t)}) - \max_{j\neq k} f_j(x^{(t)})\rvert}{\lVert \nabla_x f_k(x^{(t)}) - \nabla_x f_l(x^{(t)})\rVert_2^2}\left(\nabla_x f_k(x^{(t)}) - \nabla_x f_l(x^{(t)})\right)$	✗	Static classifier	61.35	$O(T \cdot d)$
ACraft	$x_{adv}^{t+1} = \Pi_{\epsilon,x}\left(x_{adv}^{t} - \alpha\,\mathrm{sign}\left(\mu m^{(t)} + (1-\mu)\lambda_{rev}\nabla_x \mathcal{L}_{comb}^{(t)}\right)\right)$	✓	FSCIL	41.83	$O(T \cdot d)$

optimized for one session can become ineffective later. Third, each support example has disproportionate influence on prototypes and decision boundaries. These properties motivate searching over loss composition, feature targets, and update state rather than only increasing the steps of a fixed attack.

Figure 1 contrasts three paradigms. Canonical attacks use a fixed objective; one-shot LLM prompting can generate code but receives no target-task feedback; ACraft closes the loop among generated programs, FSCIL evaluation, and evolutionary control. The human still specifies the threat model, legal program interface, and prompt prior. "Discovery" therefore means feedback-guided search under these choices, not generation without human knowledge.

1.2 Contributions

We propose **ACraft**, a feedback-guided attack-program search framework for FSCIL. It repeatedly generates candidates, evaluates them under a common white-box clean-label poisoning protocol, and uses a PPO-based controller to choose whether to initialize, refine, or synthesize the next candidates. Programs may change dynamic loss composition, feature-layer targets, and momentum-style state rather than merely retuning a fixed loss.

Our contributions are: (i) an FSCIL-specific formulation of attack design as task-grounded search over executable programs; (ii) an integrated LLM/evaluator/PPO loop whose components are established individually but are evaluated here as a closed system; and (iii) a study with strict validation/test separation, partial-poisoning rates, tuned canonical and poisoning baselines, an automated baseline, cross-learner/backbone transfer, run provenance, and search-cost reporting.

2 Rethinking the Design of Attack Algorithms

Table 1 compares representative handcrafted attacks with ACraft. Standard attacks are primarily designed to optimize fixed losses for static image classification models under bounded perturbation constraints. Although formulas may share generic operators such as sign, clip, and projection, these operators only enforce the perturbation budget. The algorithmic identity of ACraft lies in the generated attack program: it can alter the symbolic loss composition, emphasize different feature layers, and maintain momentum-style states across update steps. Therefore, ACraft is not merely a retuned PGD variant, but a program-level search framework for FSCIL-specific attack mechanisms.

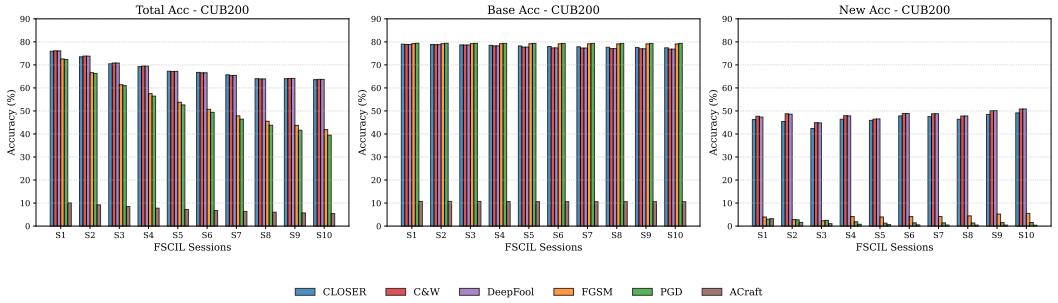


Figure 2: ACraft *v.s.* existing attacks for the FSCIL task on CUB-200 under the separate full-control clean-label diagnostic ($\rho = 100\%$). This figure is an upper-bound analysis and is not the primary 50% setting.

The distinction is important because FSCIL simultaneously requires old-knowledge preservation and few-shot adaptation to new classes. A perturbation that is mildly harmful in static classification can become persistent in FSCIL if it corrupts the few support samples used to update prototypes or classifier weights. Conversely, a handcrafted attack optimized for the current model state may be neutralized by later incremental updates. The separate full-control CUB-200 diagnostic in Figure 2 illustrates this gap: standard attacks produce limited degradation, while ACraft strongly damages both base-class retention and novel-class adaptation. It is not used as the primary 50% miniImageNet result.

We further examine whether LLMs can support this search. A one-shot strategy (Appendix A) asks an LLM to output attack ideas and code but receives no task feedback. ACraft instead couples generation with executable FSCIL evaluation and evolutionary control. The component study supports the value of this feedback within the evaluated system; it does not prove that PPO is superior to every matched-budget controller. Section 4 gives the formulation.

3 Related Work

Adversarial attacks are fundamental to assessing the robustness of deep neural networks by exposing their intrinsic vulnerabilities. Attacks are broadly categorized based on the attacker's knowledge of the target model. White-box attacks Goodfellow et al. (2014b); Madry et al. (2017b) operate under the assumption of full access to the model's architecture and parameters, primarily leveraging gradient information to craft perturbations. Conversely, black-box attacks are conducted in limited-knowledge settings, typically employing either query-based strategies, which can be score-based or decision-based Chen et al. (2017); Brendel et al. (2018), or transfer-based techniques Dong et al. (2018). Besides inference-time evasion attacks, training-time poisoning and backdoor attacks have also been studied to compromise model behavior during training Huang et al. (2021); Gu et al. (2017); Abbasi et al. (2024); Li & Ditzler (2022), which provides important baselines for evaluating attacks under the poisoning setting. However, a foundational assumption in much of the existing literature is that the target model is static, possessing fixed parameters and a stable decision boundary. The security implications for dynamic learning models, whose parameters and representations evolve throughout their lifecycle, remain largely underexplored.

Few-Shot Class-Incremental Learning (FSCIL) epitomizes such a dynamic learning paradigm. This setting requires a model to sequentially learn novel classes from a few examples while mitigating the catastrophic forgetting of previously learned ones. This process inherently involves continuous shifts in the model's decision boundaries and internal representations. Despite the development of sophisticated FSCIL strategies Zhao et al. (2025); Liu & Yang (2025); Chen et al. (2025); Li et al. (2024); Oh et al. (2024b); Zhou et al. (2023); Wang et al. (2024); Ahmed et al. (2024), the transient vulnerabilities, or attack surfaces, that emerge during this dynamic learning process have been largely overlooked by conventional attack methodologies. This oversight constitutes a significant research gap.

Our Positioning. Our framework relates to AutoML and LLM-guided program search, which generate or optimize algorithms, code, or reward functions under task feedback He et al. (2021);

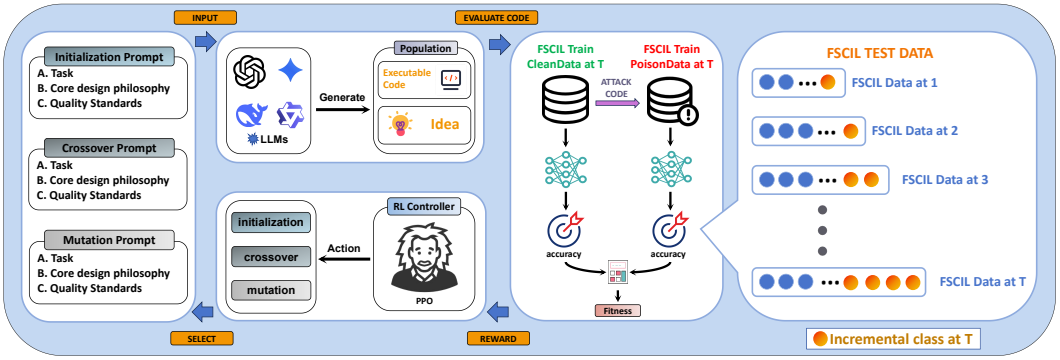


Figure 3: Overview of ACraft.

Chen et al. (2021); Yang et al. (2024); Romera-Paredes et al. (2024); Ma et al. (2024). ACraft does not claim novelty for LLM generation, evolutionary operators, or PPO individually. Its contribution is the FSCIL-grounded search problem and integration: candidates are executable clean-label poisoning programs whose symbolic loss, feature targets, and stateful update rule may change, and they are retained only after evaluation in a non-stationary incremental learner. The prompt defines the threat model and legal interface and therefore remains a human prior; the search automates candidate generation and evaluation under that prior.

4 Automatic Attack Discovery for FSCIL

4.1 Overview of ACraft

ACraft uses an LLM to propose candidate attack programs together with executable code, while avoiding the absence of task feedback in one-shot generation. The framework contains three co-operating components: an *Adversarial Algorithm Generator*, an *Attack Efficacy Evaluator*, and a *PPO-based Evolution Controller*. The generator proposes programs, the evaluator measures their effect on a target FSCIL model, and the controller uses that feedback to guide future generations. LLM generation, evolutionary search, and PPO are established techniques; our technical focus is their FSCIL-grounded executable-program loop.

Adversarial Algorithm Generator. The generator asks an LLM to output two coupled artifacts for each attack algorithm: a strategic rationale R, which explains the intended mechanism, and executable code C, which can be evaluated in the FSCIL training pipeline. This design keeps the generated attack interpretable while allowing the search to modify more than hyperparameters. Candidate programs may change the loss terms, layer emphasis, update rule, or state variables used by the attack.

Attack Efficacy Evaluator. The evaluator deploys each generated program against the target FSCIL model and summarizes its effect with a scalar fitness score. The score primarily measures accuracy degradation and catastrophic forgetting, while also accounting for execution-side cost and perturbation constraints. It is the only performance signal returned to the search loop, so it defines what kinds of attack behaviors are rewarded.

PPO-based Evolution Controller. The controller learns a meta-policy over evolutionary actions. Given the current population statistics, it decides whether the generator should initialize a new attack family, refine a strong parent, or synthesize multiple parents. PPO is an implementation choice for the noisy reward landscape, not a new optimization algorithm or a claim of superiority to every matched-budget controller.

4.2 Mathematical Formulation

We consider a sequential FSCIL task $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_N\}$, where session s provides a few-shot support set $D_s^{sup} = \{(x_{i,s}, y_{i,s})\}_{i=1}^{n_s}$. ACraft searches a latent program space $\mathcal{G}$ encoded by the LLM. Each $g \in \mathcal{G}$ is a rationale–code pair (R, C) whose code may modify a subset I_s, $|I_s| = \lfloor \rho n_s \rfloor$, subject to

$$\|x_{i,s}^{adv} - x_{i,s}\|_\infty \le \epsilon, \qquad y_{i,s}^{adv} = y_{i,s}. \tag{1}$$

The poisoned support data enter only through the learner's ordinary session update. Thus ρ controls attacker capacity, labels remain clean, and $\Pi_{\epsilon,x}$ denotes projection onto the intersection of the L_∞ ball around x and the valid image domain.

Let $P(M, D)$ denote the performance of model M on dataset D. We distinguish the clean model M_{clean} from M_g, the model obtained after poisoned training specified by algorithm g. To avoid leakage during discovery, ACraft explicitly separates a search-time validation split D_{val} from a held-out test split D_{test}. All fitness and reward signals are computed only on D_{val}, while D_{test} is invisible until final reporting. The search-time attack objective is

$$L_{attack}(M_g, D_{val}) = P(M_{clean}, D_{val}) - P(M_g, D_{val}), \tag{2}$$

and ACraft searches for

$$g^* = \arg\max_{g \in \mathcal{G}} \mathbb{E}_{D_{val}}[L_{attack}(M_g, D_{val})]. \tag{3}$$

After search, the fixed best program is evaluated once on the held-out test split:

$$\text{Eval}(g^*) = \mathbb{E}_{D_{test}}[P(M_{clean}, D_{test}) - P(M_{g^*}, D_{test})]. \tag{4}$$

This validation/test separation prevents direct test-label tuning, although selection on one validation split can still overfit; we state this limitation explicitly.

4.3 Core Mechanisms of Attack Generation

We model the generator as a set of discrete transformation functions F_{trans}. Each transformation maps a context C, consisting of instructions and optionally parent algorithms, to a new executable candidate g':

$$F : C \to \mathcal{G}, \qquad C = (\Pi_{sub}, G_{sub}). \tag{5}$$

The action space contains three transformation types. **Genesis** F_{init} creates diverse seed algorithms from the initial instruction when no parent is available. **Refinement** F_{refine} locally improves a promising parent while preserving its core strategy. **Synthesis** F_{synth} combines multiple high-performing parents to encourage exploration and mechanism-level innovation.

This search is broader than tuning a fixed PGD-style objective. In practice, the discovered attacks may synthesize a dynamic composite loss, assign larger weights to input-proximal feature layers, and combine these objectives with momentum-style states:

$$\begin{aligned} \mathcal{L}_{comb}^{(t)} &= \lambda_f^{(t)} \mathcal{L}_{feat}^{(t)} + \lambda_o^{(t)} \mathcal{L}_{old}^{(t)} + \lambda_b^{(t)} \mathcal{L}_{bdry}^{(t)}, \\ m^{(t+1)} &= \mu m^{(t)} + (1-\mu) \nabla_x \mathcal{L}_{comb}^{(t)}. \end{aligned} \tag{6}$$

Such mechanisms are suitable for FSCIL because many methods freeze or stabilize feature extractors to preserve old knowledge. Once early or input-proximal features are destructively distorted by poisoned support samples, later few-shot updates cannot easily repair them, which amplifies catastrophic forgetting across sessions.

4.4 Fitness and PPO Control

The evaluator computes a multi-attribute fitness score $\phi(g)$ for each generated algorithm. We use an objective vector $\vec{J}(g) = [J_{succ}(g), J_{cost}(g)]^T$, where J_{succ} measures attack success and J_{cost} measures execution-side expense and perturbation cost. The success term balances damage to old and new classes:

$$\begin{aligned} J_{succ}(g) = \alpha\big(P(M_{clean}, D_{old}) - P(M_g, D_{old})\big) \\ + (1-\alpha)\big(P(M_{clean}, D_{new}) - P(M_g, D_{new})\big). \end{aligned} \tag{7}$$

The scalar fitness is

$$\phi(g) = w_{succ} J_{succ}(g) + w_{cost} J_{cost}(g). \tag{8}$$

The cost term above concerns candidate execution, not the complete discovery loop. We therefore report the measured offline wall-clock and token accounting separately in the supplementary material; deployment executes the selected program without repeating LLM/PPO search. The retained logs do

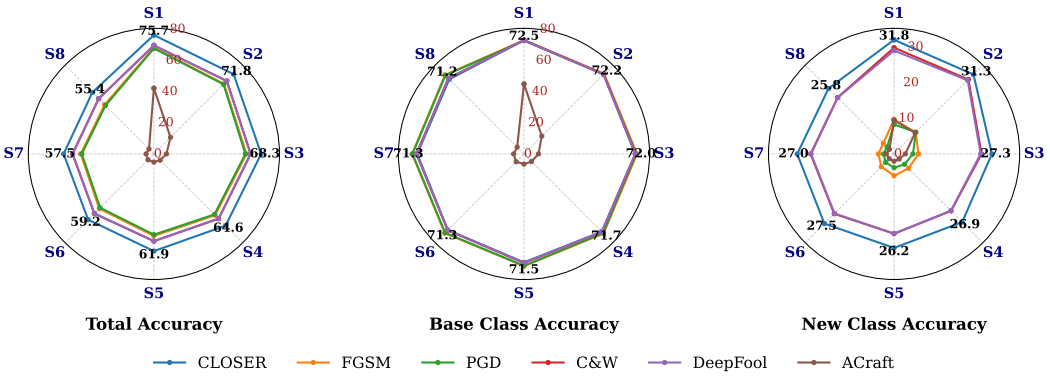


Figure 4: ACraft *v.s.* existing attacks for the FSCIL task on CIFAR-100 under the separate full-control clean-label diagnostic ($\rho = 100\%$). This figure is an upper-bound analysis and is not the primary 50% setting.

not support a reliable decomposition into request, candidate-evaluation, PPO, and I/O time, so we do not characterize the search as inexpensive.

The controller learns a policy $\pi_\theta(a_t|s_t)$ over evolutionary actions $a_t = (F, C)$, where s_t summarizes the current population and historical actions. After executing an action, the average offspring fitness is used as reward. PPO updates the controller with the clipped objective

$$L_{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right], \tag{9}$$

where $r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_{old}}(a_t|s_t)$. The clipping term prevents unstable policy shifts, which is important because generated code quality and FSCIL attack effectiveness can vary sharply across iterations.

4.5 Closed-Loop Workflow

For comparison, FGSM, PGD, C&W, and DeepFool retain their canonical static-classifier objectives. At each session we apply each operator to the same current support data under its stated perturbation constraint, keep the labels unchanged, and feed the result through the same FSCIL update. These are canonical static-objective data-attack baselines in a shared training-time pipeline; we do not add old/new-class, forgetting, or cross-session terms and do not present them as FSCIL-specific attacks.

ACraft proceeds as follows. First, the controller selects F_{init} and the generator creates N seed algorithms to form the initial population. Second, at iteration t, the controller observes s_t, samples an action $a_t = (F, C)$, and asks the generator to create offspring programs. Third, the evaluator executes each offspring against the target FSCIL pipeline and assigns a fitness score, penalizing invalid or failed code. Fourth, PPO uses the average offspring fitness as reward and updates the policy. Finally, the union of parents and offspring is ranked by fitness, the top N candidates survive, and the loop repeats until the iteration budget is exhausted or the population converges. The final output is $g^* = \arg\max_{g \in \mathcal{G}} \phi(g)$.

5 Experiments and Discussion

5.1 Experimental Settings

We evaluate miniImageNet Vinyals et al. (2016), CIFAR-100 Krizhevsky et al. (2009), and CUB-200 Wah et al. (2011), using CLOSER Oh et al. (2024a) as the main target. The white-box clean-label attacker can access model parameters, gradients, and activations during incremental training and perturb a fraction ρ of current-session support samples under $\epsilon \leq 8/255$. Our primary operating point is partial control at $\rho = 50\%$; we additionally evaluate 25% and 75% control. For session accuracies A_s, every table uses $\bar{A} = (S+1)^{-1} \sum_{s=0}^{S} A_s$ and $\Delta_{avg} = \bar{A}_{clean} - \bar{A}_{attack}$.

Table 2: Attack performance under partial support-set control on miniImageNet.

ρ	Clean Avg.	Attacked Avg.	Drop
25%	62.78	48.62	14.16
50% (primary)	**62.78**	**41.83**	**20.95**
75%	62.78	26.71	36.07

Table 3: Session-wise miniImageNet accuracy under 50% clean-label support-set poisoning.

Method	**S0**	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**Avg**	**Drop**
CLOSER (clean)	76.02	71.61	67.99	64.69	61.70	58.94	56.23	54.52	53.33	62.78	–
FGSM	76.02	69.88	65.91	62.34	59.11	56.06	53.27	51.18	49.78	60.39	2.39
PGD	76.02	69.72	65.74	62.15	58.98	55.92	53.12	51.04	49.59	60.25	2.53
C&W	76.02	70.34	66.59	63.22	60.19	57.44	54.76	52.86	51.51	61.44	1.34
DeepFool	76.02	70.26	66.48	63.14	60.07	57.36	54.69	52.71	51.42	61.35	1.43
ACraft (ours)	**76.02**	**45.51**	**41.71**	**39.38**	**38.11**	**35.30**	**34.97**	**33.43**	**32.04**	**41.83**	**20.95**

During discovery, fitness and rewards use only a validation split; the held-out test split is invisible until final reporting. Validation-tuned baselines use the same split and poisoning interface, while evasion rows are diagnostic rather than the main threat model. Main tables are single-run point estimates unless explicitly marked; the four-run LLM study is a separate experimental batch. Detailed prompts and settings are in Appendices B–C.

5.2 Main Results

Table 2 gives the primary result: controlling half of the current-session support samples reduces average accuracy from 62.78% to 41.83%, a 20.95-point drop. The effect remains substantial at 25% control and increases consistently with attacker capacity.

Partial-control protocol. At each incremental session, the attacker can modify at most a fraction ρ of the incoming support samples, while labels and the remaining samples are unchanged. The perturbation budget remains $\epsilon = 8/255$, and poisoned samples enter the learner only through its ordinary incremental update. We choose $\rho = 50\%$ as the main setting because it models partial compromise of the support-data interface rather than complete control. Plausible access paths include a compromised ingestion stage, a malicious data curator, or an untrusted contributor supplying part of the session data.

The discovered program is fixed before the poisoning-rate evaluation. Changing ρ therefore varies only attacker capacity rather than triggering a new LLM search. This design separates the effectiveness of the selected attack program from the amount of support data available to the attacker.

The partial-control trajectory shows that ACraft causes an early drop after the first incremental update and maintains lower accuracy through the remaining sessions. The canonical baselines remain close to the clean trajectory under the same 50% interface and produce drops of 1.34–2.53 points.

These comparisons use the same 50% clean-label support-set interface and the common average-accuracy definition. ACraft remains the strongest degradation among the listed methods, while the table does not claim that every baseline has an FSCIL-specific objective.

CIFAR-100 and CUB-200. Figures 4 and 2 use separate dataset-specific searches and show the separate full-control clean-label diagnostic ($\rho = 100\%$). They are upper-bound analyses rather than the primary 50% setting, and therefore demonstrate automatic re-discovery/adaptation rather than zero-shot transfer of one program. Across both datasets, the selected programs suppress base-class retention and novel-class learning. We reserve *transfer* for the fixed-program cross-learner and CNN-to-ViT tests below.

5.3 Generalizability and Ablation

Table 5 evaluates whether the discovered attack transfers across FSCIL methods under the same 50% support-set control. ACraft reduces average accuracy by 21.97–23.43 points across the four

Table 4: Tuned baselines and poisoning/backdoor comparisons under the 50% support-set protocol. S0, S4, and S8 are shown for compactness; Avg is computed over the complete S0–S8 trajectory.

Method	**S0**	**S4**	**S8**	**Avg**	**Drop**
BadNets	76.02	61.63	52.95	62.69	0.09
Unlearnable Examples	76.02	57.80	47.31	59.10	3.68
MLE-STAR	76.02	54.72	46.36	56.67	6.11
FGSM, $\epsilon = 8/255$	76.02	59.30	50.00	60.59	2.19
FGSM, $\epsilon = 16/255$	76.02	58.26	47.58	59.47	3.31
PGD, $K = 20, \alpha = 2$	76.02	59.33	49.91	60.59	2.19
PGD, $K = 150, \alpha = 2$	76.02	57.75	46.98	59.05	3.73
PGD, $K = 150, \alpha = 1$	76.02	56.97	46.12	58.40	4.38
C&W, $c = 0.1$	76.02	60.36	51.86	61.55	1.23
C&W, $c = 1.0$	76.02	57.20	46.52	58.61	4.17
ACraft (ours)	**76.02**	**38.11**	**32.04**	**41.83**	**20.95**

Table 5: **Generalizability of our ACraft for more FSCIL methods in miniImageNet dataset.**

FSCIL methods	**Year**	**Acc in each session (%)↓**									**Avg↓**	**Drop↓**
		0	1	2	3	4	5	6	7	8		
Limit Zhou et al. (2023)	TPAMI2023	84.13	78.80	74.21	69.64	66.01	63.32	61.50	58.17	55.88	67.96	–
Limit + ACraft		84.13	49.95	46.75	43.54	41.12	39.12	37.79	35.65	34.06	45.79	22.17
Approximation Wang et al. (2024)	ECCV2024	84.13	75.25	70.25	67.54	64.85	61.34	59.74	56.38	53.45	65.88	–
Approximation + ACraft		84.13	47.51	43.79	41.63	39.66	37.01	35.89	33.68	31.89	43.91	21.97
OrCo Ahmed et al. (2024)	CVPR2024	84.13	77.84	73.42	69.22	65.85	62.75	60.56	57.34	55.33	67.38	–
OrCo + ACraft		84.13	49.66	46.25	43.27	40.61	38.48	36.84	34.69	33.23	45.24	22.14
Tri-WE Zhou et al. (2023)	CVPR2025	84.13	81.41	76.65	73.59	70.1	65.13	63.42	61.02	60.13	70.62	–
Tri-WE Zhou et al. (2023) + ACraft		84.13	52.19	48.53	45.85	43.19	39.83	38.20	36.83	35.96	47.19	23.43

learners: 22.17 points for Limit, 21.97 for Approximation, 22.14 for OrCo, and 23.43 for Tri-WE. This indicates that ACraft is not specific to the main CLOSER target: it captures vulnerabilities shared by FSCIL protocols, especially their dependence on limited support samples and prototype or classifier updates. Beyond FSCIL algorithms, we apply the fixed discovered program to a ViT-based setting without a new search. Table 6 reports attacked accuracies under the same 50% support-set control, with ACraft reaching 45.79% average accuracy across the reported sessions.

We also study LLM backbones in the supplementary material. Those four-run results form a separate ablation batch and must not be conflated with the main-table point estimate. They show variation across model families under the same evaluation protocol; this does not disentangle pretrained LLM knowledge from prompt design or establish invariance to arbitrary prompts.

Finally, Table 7 shows limited sensitivity to nearby PPO settings within ACraft; it is not a matched-budget comparison against every controller. Additional prompts, ablations, and limitations are in Appendices A–G.

5.4 Transfer and controller sensitivity

The transfer experiments probe two distinct questions. The cross-learner experiment keeps the discovered program fixed and changes only the FSCIL learner. Under 50% support-set control, the average drops are 22.17, 21.97, 22.14, and 23.43 points for Limit, Approximation, OrCo, and Tri-WE, respectively. The comparable clean averages are 67.96, 65.88, 67.38, and 70.62. Thus, the absolute effect is similar across learners even though their clean trajectories differ; this is evidence of transfer across update rules, not a claim that all learners share identical representations.

The CNN-to-ViT experiment changes the backbone while keeping the discovered program fixed and does not perform a second LLM search. On the ViT setting, the attacked averages are 61.28 (FGSM), 60.17 (PGD), 60.72 (C&W), 59.71 (DeepFool), and 45.79 (ACraft). The comparison is intended to test whether the program remains executable and damaging after a representation change. It does not

Table 6: Transferability of ACraft to ViT-based FSCIL settings under 50% support-set control. Avg is computed over the complete S0–S8 trajectory.

Method / Setting	S0	S1	S2	S3	S4	S5	S6	S7	S8	Avg
FGSM (on ViT)	76.02	71.10	66.80	63.55	60.19	57.65	54.55	51.01	50.65	61.28
PGD (on ViT)	76.02	70.35	65.65	62.10	59.04	56.55	53.55	49.31	48.96	60.17
C&W (on ViT)	76.02	70.65	66.25	62.85	59.63	57.20	54.10	50.00	49.78	60.72
DeepFool (on ViT)	76.02	69.85	65.15	61.50	58.53	55.90	52.65	49.61	48.18	59.71
ACraft (on ViT)	**76.02**	**51.80**	**46.80**	**43.63**	**42.47**	**40.60**	**38.15**	**36.85**	**35.79**	**45.79**

Table 7: Sensitivity analysis of PPO hyperparameters under the primary 50% support-set poisoning protocol.

Configuration	LR	Clip	Attack Drop (%)
LR (Lower Bound)	1×10^{-4}	0.2	20.28
LR (Upper Bound)	5×10^{-4}	0.2	20.61
Clip (Lower Bound)	3×10^{-4}	0.1	20.49
Clip (Upper Bound)	3×10^{-4}	0.3	20.68
Default	$\mathbf{3 \times 10^{-4}}$	**0.2**	**20.95**

establish zero-shot transfer across arbitrary architectures or datasets, because the target configuration and data pipeline remain fixed.

Finally, the PPO sensitivity study varies one controller hyperparameter at a time around the default setting. The resulting drops are 20.28 and 20.61 for the lower and upper learning-rate bounds, and 20.49 and 20.68 for the lower and upper clipping bounds; the default configuration gives 20.95. These values indicate that the observed effect is not tied to one exact nearby hyperparameter choice. They are not a matched-budget comparison against random search, evolutionary search without PPO, or another controller, and therefore do not support a universal claim that PPO is optimal.

All transfer and sensitivity results are reported as separate point-estimate experiments. The primary 50% result remains the miniImageNet/CLOSER run in Table 3; the additional experiments characterize scope and robustness rather than increasing the number of independent runs behind that headline number.

6 Conclusion and Limitations

ACraft formulates FSCIL attack design as feedback-guided search over executable clean-label poisoning programs. The contribution is the task-grounded search loop and empirical security analysis, not a new LLM, evolutionary operator, or PPO method. At the primary 50% partial-control setting, the fixed miniImageNet program causes a 20.95-point average-accuracy drop; 25% control still causes 14.16 points. The session-wise trajectory shows that the degradation begins during incremental adaptation and persists through later sessions. Dataset-specific searches show rediscovery rather than zero-shot cross-dataset transfer, whereas fixed-program tests provide evidence across FSCIL learners and from CNN to ViT.

The study remains limited by white-box access, a human-specified prompt and program interface, costly candidate evaluation, and incomplete uncertainty and action-trace records for some runs. Canonical attacks are static-objective data-attack baselines in the shared FSCIL pipeline, not purpose-built FSCIL attacks. Validation/test separation prevents direct test-label tuning but not selection-induced overfitting. Future work should evaluate gray-/black-box and limited-session access, matched-budget controllers and FSCIL-specific baselines, multiple seeds for every headline result, and auditable compute traces.

Reproducibility statement

We report the validation/test separation, threat model, perturbation and poisoning-rate budgets, and run provenance in the main paper. Prompts, implementation details, search hyperparameters, discovered programs, and additional results are provided in the supplementary material. Point estimates are identified as such; no unrecorded variance statistics are inferred.

Ethics statement

ACraft is intended for controlled robustness evaluation of FSCIL systems. It can expose vulnerabilities but could also facilitate denial-of-service attacks on compromised training pipelines. We therefore restrict the study to bounded clean-label perturbations on public benchmarks, disclose the strong access assumptions, and recommend isolated execution of generated code. The work uses no human-subject data.

AI USE STATEMENT

Large language models are an integral component of ACraft: they propose rationale–code candidate attack programs that are executed and selected by the reported FSCIL validation loop. The authors designed the methodology, verified the generated programs and reported results, and take responsibility for the final submission.

REFERENCES

Ali Abbasi, Parsa Nooralinejad, Hamed Pirsiavash, and Soheil Kolouri. Brainwash: A poisoning attack to forget in continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24057–24067, 2024.

Noor Ahmed, Anna Kukleva, and Bernt Schiele. Orco: Towards better generalization via orthogonality and contrast for few-shot class-incremental learning. In *41st IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2024.

Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *International Conference on Learning Representations (ICLR)*, 2018.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pp. 39–57. Ieee, 2017.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, et al. Evaluating large language models trained on code. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 12103–12119, 2021.

Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *ACM Workshop on Artificial Intelligence and Security (AISec)*, pp. 15–26, 2017.

Yiyang Chen, Tianyu Ding, Lei Wang, Jing Huo, Yang Gao, and Wenbin Li. Enhancing few-shot class-incremental learning via training-free bi-level modality calibration. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 9881–9890, 2025.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021.

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9185–9193, 2018.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014a.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014b.

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

Xin He, Kaiyong Zhao, and Xiaowen Chu. Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622, 2021.

Hanxun Huang, Xingjun Ma, Sarah Monazam Erfani, James Bailey, and Yisen Wang. Unlearnable examples: Making personal data unexploitable. In *ICLR*, 2021.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Dong Li, Aijia Zhang, Junqi Gao, and Biqing Qi. An efficient memory module for graph few-shot class-incremental learning. *Advances in Neural Information Processing Systems*, 37:130084–130108, 2024.

Huayu Li and Gregory Ditzler. Targeted data poisoning attacks against continual learning neural networks. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2022.

Ye Liu and Meng Yang. Sec-prompt: Semantic complementary prompting for few-shot class-incremental learning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 25643–25656, 2025.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017a.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017b.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2574–2582, 2016.

Junghun Oh, Sungyong Baik, and Kyoung Mu Lee. Closer: Towards better representation learning for few-shot class-incremental learning. In *ECCV*, 2024a.

Junghun Oh, Sungyong Baik, and Kyoung Mu Lee. Closer: Towards better representation learning for few-shot class-incremental learning. In *European Conference on Computer Vision*, pp. 18–35. Springer, 2024b.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, et al. Mathematical discoveries from program search with large language models. *Nature*, 625:468–475, 2024.

Eli Verwimp, Rahaf Aljundi, Shai Ben-David, Matthias Bethge, Andrea Cossu, Alexander Gepperth, Tyler L Hayes, Eyke Hüllermeier, Christopher Kanan, Dhireesha Kudithipudi, et al. Continual learning: Applications and the road forward. *arXiv preprint arXiv:2311.11908*, 2023.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. 2016.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

Xuan Wang, Zhong Ji, Xiyao Liu, Yanwei Pang, and Jungong Han. On the approximation risk of few-shot class-incremental learning. In *European Conference on Computer Vision*, pp. 162–178. Springer, 2024.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations (ICLR)*, 2024.

Li-Jun Zhao, Zhen-Duo Chen, Yongxin Wang, Xin Luo, and Xin-Shun Xu. Attraction diminishing and distributing for few-shot class-incremental learning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 25657–25666, 2025.

Da-Wei Zhou, Han-Jia Ye, Liang Ma, Di Xie, Shiliang Pu, and De-Chuan Zhan. Few-shot class-incremental learning by sampling multi-phase tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45 (11):12816–12831, 2023. doi: 10.1109/TPAMI.2022.3200865.