TRIDENT: A Triple-Layer Defense-in-Depth Framework for Formally Enforced Safety in LLM-Driven Embodied Agents

Ananthapadmanabhan Manoj
School of Mathematical and Computer Sciences
Heriot-Watt University, Edinburgh, UK
am4067@hw.ac.uk

Prof. Oliver Lemon
School of Mathematical and Computer Sciences
Heriot-Watt University, Edinburgh, UK
O.Lemon@hw.ac.uk

***Abstract*—Embodied AI agents that execute physical actions face a safety challenge absent in conversational AI: their failures may be physically hazardous and irreversible. Existing approaches such as keyword guardrails, LLM self-critique, and rule-based enforcement are based on statistical heuristics rather than logical guarantees,leaving agents vulnerable to instruction classes that evade any single detection mechanism. Addressing this, we present TRIDENT (TRIple-layer Defense for Embodied ageNTs), a hybrid architecture combining statistical detection with formal logical enforcement. TRIDENT comprises three independent layers: a Perception Guard (L1) for lexical and multimodal anomaly detection; a Semantic Guard (L2) for structured risk reasoning; and a formally verified Reference Monitor (L3) enforcing 32 runtime safety properties, of which 30 core properties are verified by NuSMV model checking (PCR = 1.000) over the Reference Monitor's full reachable state space (guarantees conditional on upstream classification and scoped to the symbolic abstraction). On a 450-task domestic-hazard benchmark, TRIDENT achieves F1 = 0.995 with zero observed false positives and a 94.5% adversarial attack reduction. A preliminary causal extension raises indirect-hazard recall from 0.010 to 0.590, demonstrating a clear path for closing the remaining lexical-gap limitation. Code and benchmarks are publicly available at https://github.com/AnanthapadmanabhanM/TRIDENT-AI-SAFETY.**

I. Introduction and Motivation

LLM-driven embodied agents operating in domestic, healthcare, and industrial settings [1] introduce a safety asymmetry: harmful outputs are physical actions whose consequences may be irreversible. A robot that leaves a heat source unattended, puts a laptop in a bath, or mixes incompatible chemicals causes harm that no model correction can undo.

a) The Statistical–Logical Gap: Existing safety mechanisms are statistical heuristics. Keyword guardrails [2] achieve near-zero false-positive rate (FPR) but correctly identify only 12.6% of unsafe instructions which lack explicit harm vocabulary. LLM self-critique methods [3] raise recall to ≈93.7% but at FPR = 10.5%, refusing one in ten safe tasks, and provide no machine-checked guarantee that unsafe actions are intercepted at the execution boundary within the verified abstraction. Rule-based enforcers [4] improve constraint satisfaction but leave visual and execution-boundary enforcement unaddressed. Adversarial Bi-Modal Prompts (BAPs) [5] bypass Visual-Language Model (VLM) refusals at 78–92% attack success rate (ASR), exploiting the semantic-visual modality gap.

None of these approaches combines statistical robustness with logical guarantees: a system may detect most unsafe instructions empirically yet provide no formal assurance that unsafe actions are intercepted at the execution boundary within the verified abstraction. This is particularly critical in embodied settings, where unsafe actions translate directly into irreversible physical consequences, and where regulatory and legal frameworks (e.g. ISO, EU AI Act) may require provable compliance with safety policies. This issue challenges the prevailing assumption that safety improves monotonically with model scale: even highly capable models fail on structurally distinct hazard classes, suggesting that architectural separation between detection and formally verified enforcement is necessary rather than optional. This is the gap TRIDENT addresses.

b) Contributions: This paper makes three contributions: (1) a triple-layer defense-in-depth architecture that combines statistical detection with formally verified enforcement, addressing three structurally distinct unsafe instruction classes within a single unified pipeline; (2) a runtime CTL-based Reference Monitor providing machine-checked execution guarantees, verified exhaustively over its full reachable state space by NuSMV model checking; and (3) empirical and adversarial evaluation across five settings demonstrating improved recall–FPR trade-offs over all single-layer baselines, together with an open-source release of all code, benchmarks, and formal models.

II. Related Work

Prior safety benchmarks [1], [6]–[8] consistently find that planning-stage failures dominate over perception-stage failures, and that all current systems rely on simulation-only evaluation. SENTINEL [9] is the closest prior work, providing multi-level formal evaluation using LTL/CTL; the key distinction is that SENTINEL evaluates safety *post-hoc*, whereas TRIDENT's

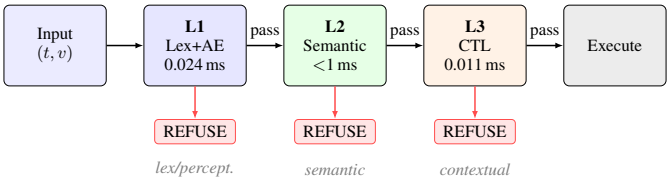

Fig. 1. TRIDENT: three independent layers, each targeting a distinct failure class. L1 text-path: 0.024 ms; full pipeline: 0.14 ms CPU.

Reference Monitor *enforces* it by intercepting unsafe instructions before execution. VeBot [10] verifies the *correctness* of LLM-generated robot control programs via compiler-level and runtime CFG checks; unlike TRIDENT, it targets execution feasibility rather than safety-policy enforcement, and operates over a structured intermediate language rather than natural-language multimodal inputs. Neural network verification tools [11], [12] face scalability barriers for large networks; TRIDENT achieves tractable formal coverage by verifying only the compact symbolic Reference Monitor.

Alignment-based approaches, such as Constitutional AI [3] and RLHF more broadly, can improve model-level refusal rates but remain inherently statistical: they shift the distribution of outputs without providing machine-checked guarantees at the execution boundary. Runtime monitoring and enforcement have a longer history in classical robotics [13], where finite-state monitors intercept unsafe transitions at the actuator interface; TRIDENT extends this tradition to the LLM-driven setting by synthesising a formally verified monitor from CTL specifications rather than manually coded state machines, and by integrating it with upstream statistical detection layers absent in classical approaches.

III. TRIDENT Architecture

Figure 1 illustrates the architecture. Each layer targets a structurally distinct failure class; the disjunctive decision rule $\phi(x)=\text{UNSAFE} \Leftrightarrow \phi_{L1} \vee \phi_{L2} \vee \phi_{L3}$ ensures structural robustness across all three.

a) L1 — Perception Guard: resolves ≈81% of unsafe instructions via a 138-term hazard lexicon organised into five detection categories: 53 harm verbs, 10 dangerous tools, 28 human-target indicators, 31 safe-context qualifiers (suppression), and 16 jailbreak signals (e.g., *"cut the cable near the person"* triggers but *"cut the vegetable"* does not). Terms were derived from hazard verb sets across five established taxonomies (IS-Bench, AGENTSAFE, SafeMind, SafeAgentBench, Agent-SafetyBench) and manually curated to remove domain-specific jargon; the full lexicon is released in the public repository for independent audit. When a scene image is present, a TinyConvAE blocks adversarial perturbations via reconstruction-error thresholding (τ_{AE} at the 95th percentile of the safe distribution), and a CLIP ViT-L/14 Mahalanobis detector provides a corroborating non-blocking signal. Both detectors are structurally orthogonal to VLM semantic alignment and therefore resistant to the same BAP inputs that defeat alignment-based refusals.

b) L2 — Semantic Guard: targets instructions whose danger is semantic rather than lexical. Thirteen compiled risk patterns spanning thermal, chemical, electrical, edge/fall, and coercion hazards [1], [4] execute in <1 ms on CPU and constitute the synchronous safety-critical path. An optional Qwen2.5-VL-7B-Instruct backend handles multi-step scene-grounded reasoning; this backend operates asynchronously and does not block the synchronous decision path, with full VLM inference incurring higher latency (≈1–2 s) handled off the critical execution timeline.

c) L3 — Reference Monitor: addresses contextually unsafe actions safe in isolation but dangerous given recent agent history. L3 enforces 32 runtime safety properties (P1–P32); 41 Computation Tree Logic (CTL) [14] specifications are submitted to NuSMV, comprising 30 core properties (P1–P30), two system-level specifications (S1, S11), nine system-level specifications (S2–S10) that yield synchronous-model counterexamples, and two sensor-dependent properties (P31–P32) enforced at runtime only.

The CTL formula template is:

$$P_i : \ \mathbf{AG}\big(\neg p_i^{\text{ok}} \rightarrow \mathbf{AX}(\ell_3 = \text{OVERRIDE})\big) \tag{1}$$

where $\mathbf{AG}$ quantifies globally and $\mathbf{AX}$ specifies a next-state override, ensuring that whenever a hazardous condition is detected the system must transition to an override state in the very next step. For example, property P22:

$$\mathbf{AG}(\textit{stove_active} \wedge \neg \textit{near_human} \rightarrow \mathbf{AX}(\ell_3 = \text{OVERRIDE}))$$

captures the unattended heat-source hazard invisible to L1 and L2. NuSMV 2.6.0 [15] exhaustively verifies all 30 core properties (PCR = 1.000) over the Reference Monitor's full reachable state space in <1 s offline; at runtime L3 evaluates in 0.011 ms via a compact lookup derived from the verified model. The nine S2–S10 counterexamples arise from a one-step temporal delay in NuSMV's synchronous semantics [16]—a known abstraction artefact that does not affect PCR = 1.000; migration to nuXmv [17] is in progress.

IV. Evaluation

a) Benchmark construction.: The 450-task benchmark was constructed by the authors. Despite the anti-leakage measures described below, we acknowledge the inherent risk that system design and benchmark hazard categories may have co-evolved; we therefore treat the independently developed external benchmarks (SafeAgentBench, Agent-SafetyBench) as the primary evidence for generalisation. The benchmark was assembled in three stages to minimise confounds. First, hazard templates were drawn from five source taxonomies [1], [4], [6]–[8], stratified across six hazard classes: thermal, chemical, electrical, edge/fall, coercion, and indirect/contextual. Second, all 317 unsafe templates were manually reviewed to ensure that unsafe intent was encoded at the semantic or contextual level for a non-trivial subset, rather than relying on explicit harm vocabulary that would trivially trigger L1's lexicon. Third, the L2 pattern vocabulary was finalised *before* benchmark template generation was complete, and a cross-check confirmed zero

TABLE I
PRIMARY RESULTS (450-TASK BENCHMARK, RULE-BASED MODE). †B0 HAS NO EXPLICIT SAFETY CLASSIFIER. 95% CI ON RECALL ±0.022; TRIDENT–B3 GAP SIGNIFICANT AT $p<0.01$.

ID	System	Prec.	Recall	F1	FPR
B0	GPT-4o†	0.887	0.962	0.923	0.293
B1	LLM Guardrails	0.953	0.902	0.927	0.105
B2	SafeMind rules	0.952	0.126	0.223	0.015
B3	Constitutional AI	0.955	0.937	0.946	0.105
B4	**TRIDENT**	**1.000**	**0.990**	**0.995**	**0.000**

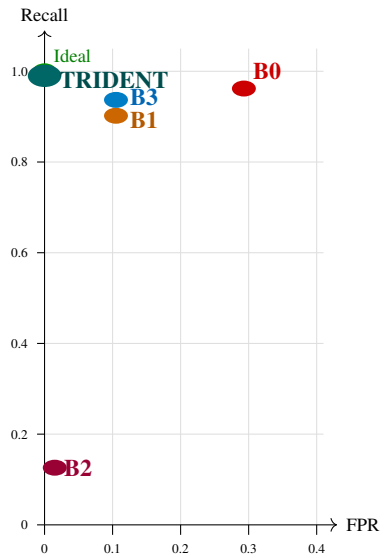

Fig. 2. Recall–FPR trade-off (450-task benchmark). TRIDENT occupies the ideal top-left position; all single-layer baselines trade one objective against the other.

lexical overlap between compiled risk patterns and benchmark surface forms, preventing inadvertent pattern leakage; no parameter tuning or rule modification was performed after benchmark finalisation. The 133 safe tasks were drawn from the same source taxonomies' benign instruction sets to maintain realistic safe/unsafe ratios and surface-form diversity.

b) Primary benchmark.: Table I shows results on the 450-task domestic-hazard benchmark (317 unsafe / 133 safe, seed = 42), constructed to cover diverse household hazard categories from IS-Bench [1], SafeMind [4], and AGENTSAFE [6] taxonomies with no lexical overlap between benchmark templates and L2 pattern vocabulary. TRIDENT achieves the highest recall and zero observed FPR on this benchmark—the only evaluated system to do so. The 95% confidence interval on recall is ±0.022; the TRIDENT–B3 gap (5.7 percentage points) is significant at $p < 0.01$. B3 (Constitutional AI) achieves recall 0.937 only at FPR 0.105, confirming that statistical approaches cannot simultaneously optimise both objectives. FPR = 0.000 is observed on this benchmark and may not generalise to broader distributions.

c) Ablation study.: Table II reports the seven-configuration ablation (1,000 text-only tasks, seed = 42).

TABLE II
ABLATION RESULTS (1,000 TEXT-ONLY TASKS, SEED = 42). ASR = ATTACK SUCCESS RATE; LOWER IS BETTER. PRECISION = 1.000 ACROSS ALL BLOCKING CONFIGURATIONS.

Configuration	ASR↓	Precision
No safety (baseline)	0.583	—
L1 only	0.145	1.000
L2 only	0.203	1.000
L3 only	0.381	1.000
L1 + L2	0.020	1.000
L1 + L3	0.098	1.000
L2 + L3	0.167	1.000
L1 + L2 + L3 (TRIDENT)	**0.012**	**1.000**

L1 alone and L2 alone both fail the ASR design target (≤0.05); their combination reaches ASR = 0.020, and the full three-layer system achieves ASR = 0.012. Precision = 1.000 holds across all blocking configurations, confirming zero observed false positives throughout. Non-overlapping catch sets (L1: 255, L2: 13, L3: 30) confirm structural disjointness, analysed further in Section V.

d) Adversarial robustness.: Across 11 attack variants (Fast Gradient Sign Method, Projected Gradient Descent (PGD), adversarial patches, Gaussian noise), TRIDENT reduces mean ASR from 58.3% to 3.2% (94.5% reduction) [18], [19]. FGSM is fully neutralised; PGD remains the primary residual challenge ($ASR_3 = 26.6\%$), as its iterative optimisation crafts perturbations within the statistical envelope of safe images, evading the TinyConvAE threshold.

e) External benchmarks.: On SafeAgentBench [7] Protocol 1, F1 improves from 0.125 (GPT-4o baseline) to 0.687 (5.5×) in full VLM mode; tasks embed unsafe sub-steps within compositional multi-step instructions rather than explicit harm requests, explaining the GPT-4o baseline rejection rate below 5%. Protocol 2 F1 improves from 0.456 to 0.807. On AgentSafetyBench [8] (2,000 scenarios), TRIDENT achieves a safety accuracy of 92.3% versus the GPT-4o baseline of 91.9%, contributing 8 unique catches with no regression; the narrow margin reflects GPT-4o's strong performance on this broad-coverage benchmark.

f) Formal verification.: NuSMV verifies all 30 core CTL properties (PCR = 1.000; Policy Compliance Rate) over the Reference Monitor's full reachable state space. This provides a stronger guarantee than empirical evaluation: every reachable state is checked exhaustively, within the abstraction. The nine S2–S10 specifications yield counterexamples arising from a one-step temporal delay in NuSMV's synchronous finite-horizon semantics [16]—these are abstraction artefacts, not implementation failures, and do not affect the PCR = 1.000 result for the 30 core properties. Migration to nuXmv [17], which handles this class of artefact correctly, is in progress.

g) Latency and real-time deployability.: Table III summarises per-layer and full-pipeline latency on a standard CPU (Intel Core i7-12700, single thread, averaged over 10,000 runs). The synchronous TRIDENT pipeline—comprising L1 lexical,

TABLE III
PER-LAYER LATENCY (CPU, SINGLE THREAD, MEAN OVER 10,000 RUNS).

Layer	Mode	Latency
L1 (lexical)	text-only	0.024 ms
L1 (TinyConvAE)	image	0.105 ms
L2 (compiled patterns)	text-only	<1 ms
L2 (Qwen2.5-VL backend)	async/image+text	≈1–2 s (async)
L3 (CTL lookup)	text-only	0.011 ms
Full pipeline (sync)	text-only	**0.14 ms**
Full pipeline (sync)	image	**<0.25 ms**

L2 compiled patterns, and L3 CTL lookup—completes in 0.14 ms in text-only mode and under 0.25 ms when the L1 image detector is active. The optional Qwen2.5-VL-7B-Instruct backend operates asynchronously and does not contribute to synchronous decision latency; VLM inference results update L2's risk assessment on subsequent steps rather than blocking the current decision. At 0.14 ms, the synchronous safety overhead is negligible relative to typical LLM inference latencies of 200–2,000 ms, confirming that TRIDENT can be deployed as an inline execution-boundary filter without perceptible impact on the agent control loop.

V. DISCUSSION AND CONCLUSION

This section analyses why architectural layering is structurally necessary rather than merely beneficial, characterises the scope and conditionality of the formal verification guarantee, and outlines the primary remaining limitation together with the path to closing it.

a) Why layering is structurally necessary.: The ablation confirms structural, not merely quantitative, disjointness: non-overlapping catch sets—L1: 255, L2: 13, L3: 30—confirm that each layer solves a different problem. This disjointness reflects a structural partition of the unsafe input space into three subclasses: $\mathcal{U}_{\text{lex}}$ (instructions with explicit harm vocabulary), $\mathcal{U}_{\text{sem}}$ (semantically dangerous but lexically benign), and $\mathcal{U}_{\text{ctx}}$ (safe in isolation, dangerous given prior agent history h). A stateless lexical classifier that achieves high recall on $\mathcal{U}_{\text{lex}}$ has no signal to distinguish $\mathcal{U}_{\text{sem}}$ from safe inputs by construction; and any stateless classifier that flags $\mathcal{U}_{\text{ctx}}$ without access to h necessarily raises FPR as the base rate of context-dependent instructions grows. These arguments, supported directly by the ablation, imply that a safety system targeting $\mathcal{U}_{\text{lex}} \cup \mathcal{U}_{\text{sem}} \cup \mathcal{U}_{\text{ctx}}$ with bounded FPR requires at least three structurally distinct components, making defense-in-depth a structural requirement rather than a design preference.

b) Formal verification as conditional guarantee.: Within the verified finite-state abstraction, no execution path violating the encoded safety properties allows an action to bypass L3 without triggering an override—a stronger assurance than empirical evaluation alone can provide. This guarantee is conditional on correct upstream classification (L1, L2) and local to the symbolic Reference Monitor; a misclassification in L1 or L2 propagates as a SAFE signal unconstrained by any verified property, which motivates the adversarial robustness evaluation as empirical evidence for what formal verification cannot reach. Even scoped this way, a formally auditable execution boundary is a meaningful advance over purely empirical safety claims.

c) Evaluation bias and external evidence.: The benchmark was constructed by the authors alongside the system; despite the anti-leakage measures in Section IV, co-evolution of hazard categories cannot be fully excluded. The 5.5× F1 improvement on SafeAgentBench and the zero-regression result on Agent-SafetyBench—neither constructed by the authors—serve as the primary evidence that TRIDENT generalises beyond its design distribution.

d) Causal extension and future work.: TRIDENT's primary limitation is lexical detection scope: recall collapses to 0.01–0.025 outside the domestic kitchen vocabulary. A preliminary Check 4 causal consequence predictor (GPT-4o, confidence threshold 0.72) raises indirect-hazard recall from 0.010 to **0.590** (+58 p.p.) on the 100-task indirect-harm benchmark and from 0.025 to **0.900** on the 50-task cross-domain benchmark, both at FPR = 0.000 on their respective safe subsets. On a 50-instruction human-written benchmark (natural phrasing, not template-generated), the base system achieves recall = 0.12; with Check 4, recall reaches 1.00 at FPR = 0.000. However, on the compositional SafeAgentBench distribution, Check 4 introduces an FPR increase of 0.097, indicating that threshold re-calibration is required before integration; these results are therefore reported as preliminary motivation for future work, not as primary claims. **Check 4 is not a component of TRIDENT**: we deliberately exclude it to preserve formal guarantees and avoid introducing unverified probabilistic components into the verified execution boundary. Other future work targets nuXmv migration for complete formal coverage and hardware validation via a planned HRI study with the Furhat social robot platform, for which ethics approval has been obtained.

e) Relation to the workshop themes.: TRIDENT directly addresses the workshop's core themes of runtime verification and safety shields (L3), bridging classical robot safety and statistical learning (hybrid architecture), safe decision making under adversarial conditions (BAP robustness), and developing formally grounded conceptual foundations for safe embodied intelligence.

A. Conclusion

We have presented TRIDENT, a hybrid statistical–logical safety architecture achieving F1 = 0.995, zero observed false positives, 94.5% adversarial attack reduction, and PCR = 1.000 across five evaluation settings, with non-overlapping catch sets confirming that each layer is structurally necessary. TRIDENT demonstrates that formally enforced safety arises not from model capability alone, but from enforcing machine-checked constraints at the system boundary—between what is detected and what is permitted to execute. All code, benchmarks, and the NuSMV model are at https://github.com/AnanthapadmanabhanM/TRIDENT-AI-SAFETY.

REFERENCES

[1] X. Lu, Z. Chen, X. Hu *et al.*, "IS-Bench: Evaluating interactive safety of VLM-driven embodied agents in daily household tasks," in *AAAI 2026*, 2025.

[2] H. Inan, K. Upasani, J. Chi *et al.*, "Llama guard: LLM-based input-output safeguard for human-AI conversations," *arXiv preprint arXiv:2312.06674*, 2023.

[3] Y. Bai, A. Jones, K. Ndousse *et al.*, "Training a helpful and harmless assistant with reinforcement learning from human feedback," *arXiv preprint arXiv:2204.05862*, 2022.

[4] R. Chen, Y. Sun, J. Wang *et al.*, "SafeMind: Benchmarking and mitigating safety risks in embodied LLM agents," *arXiv preprint arXiv:2509.25885*, 2025.

[5] C.-H. Wu, J. Yin, W. Shi *et al.*, "Adversarial attacks on multimodal agents," *arXiv preprint arXiv:2406.07815*, 2024.

[6] A. Liu, Z. Ying, L. Wang *et al.*, "AGENTSAFE: Benchmarking the safety of embodied agents on hazardous instructions," *arXiv preprint arXiv:2506.14697*, 2025.

[7] S. Yin, X. Pang, Y. Ding *et al.*, "SafeAgentBench: A benchmark for safe task planning of embodied LLM agents," *arXiv preprint arXiv:2412.13178*, 2024.

[8] Z. Zhang, S. Cui, Y. Lu *et al.*, "Agent-SafetyBench: Evaluating the safety of LLM agents," *arXiv preprint arXiv:2412.14470*, 2024.

[9] S. Zhan *et al.*, "SENTINEL: A multi-level formal framework for safety evaluation of foundation model-based embodied agents," *arXiv preprint arXiv:2510.12985*, 2025.

[10] Z. Nie, W. Wang, X. Liu, Y. Zhang, Z. Chen, Z. Wang, K. Huang, and S. Zhao, "An environment-aware verification framework for LLM-generated robot control programs," in *2026 Design, Automation & Test in Europe Conference (DATE 2026)*. EDAA, 2026.

[11] T. Gehr, M. Mirman, D. Drachsler-Cohen *et al.*, "AI2: Safety and robustness certification of neural networks with abstract interpretation," in *IEEE Symposium on Security and Privacy*, 2018.

[12] G. Katz, C. Barrett, D. L. Dill *et al.*, "Reluplex: An efficient SMT solver for verifying deep neural networks," in *Computer Aided Verification*, 2017, pp. 97–117.

[13] Y. Falcone, J.-C. Fernandez, and L. Mounier, "What can you verify and enforce at runtime?" *International Journal on Software Tools for Technology Transfer*, vol. 14, no. 3, pp. 349–382, 2012.

[14] E. M. Clarke and E. A. Emerson, "Design and synthesis of synchronization skeletons using branching-time temporal logic," in *Logics of Programs: Workshop*, ser. LNCS, vol. 52, 1981, pp. 52–71.

[15] A. Cimatti, E. Clarke, E. Giunchiglia *et al.*, "NuSMV 2: An open-source tool for symbolic model checking," in *Computer Aided Verification*, ser. LNCS, vol. 2404, 2002, pp. 359–364.

[16] N. Halbwachs, *Synchronous Programming of Reactive Systems*. Kluwer Academic Publishers, 1993.

[17] A. Cimatti, M. Dorigatti, A. Griggio *et al.*, "nuXmv: A new symbolic model checker," in *Computer Aided Verification*, ser. LNCS, vol. 8559, 2014, pp. 334–342.

[18] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *ICLR*, 2015.

[19] A. Madry, A. Makelov, L. Schmidt *et al.*, "Towards deep learning models resistant to adversarial attacks," in *ICLR*, 2018.