Reverse Engineering in Artificial Intelligence Systems: Techniques, Challenges, Security Risks, Interpretability, And Future Directions

Apurva Kolhe
Student, Department of Computer Science and Engineering
Mauli Group of Institutions, College of Engineering and Technology, Shegaon

***Abstract*—Reverse engineering of artificial intelligence (AI) systems has emerged as a critical research frontier that simultaneously threatens intellectual property and privacy while enabling auditing, interpretability, and safety verification. This survey consolidates a decade of work spanning model extraction, adversarial perturbation, gradient inversion, membership inference, and the structural analysis of neural networks and transformers. We organize the field along two orthogonal axes—the adversary's access model (black-box vs. white-box) and the recovery target (functionality, parameters, training data, or architecture)—and show how these axes unify otherwise disparate attack families. Particular attention is given to large language models (LLMs), where prompt injection, jailbreak analysis, and partial weight reconstruction have demonstrated that even commercial API-gated models leak exploitable structure. We connect offensive reverse engineering to its defensive and scientific counterpart, mechanistic interpretability and explainable AI (XAI), arguing that both pursue the same underlying objective: converting opaque parameter tensors into human-legible mechanisms. A comparative analysis of attack–defense pairs, a historical timeline, and a discussion of open-source versus proprietary vulnerability surfaces are provided, followed by enterprise security implications and a forward look toward reverse engineering of autonomous agents and AGI-scale systems. We deliberately mark uncertain citations as placeholders rather than fabricate references, and we close with suggested figures, an implementation architecture, and derived project ideas.**

***Index Terms*—Reverse engineering, model extraction, adversarial machine learning, membership inference, gradient inversion, large language models, prompt injection, mechanistic interpretability, explainable AI, AI security, transformers, autonomous agents.**

I. INTRODUCTION

Modern AI systems are increasingly deployed as opaque artifacts: trained weights are served behind prediction APIs, embedded in edge accelerators, or shipped inside consumer applications. The economic value concentrated in these artifacts—often representing millions of dollars of compute and proprietary data—has made them attractive targets for *reverse engineering*, the systematic recovery of a system's internal structure, parameters, decision logic, or training data from externally observable behavior. Unlike classical software reverse engineering, where the target is a deterministic instruction stream, reverse engineering of AI confronts high-dimensional, continuous, statistically defined functions whose behavior is only partially observable [3], [9].

The motivations are dual-use. On the adversarial side, competitors and attackers seek to clone proprietary models [3], [33], extract memorized private data [10], infer training-set membership [5], or reconstruct architectures via side channels [29], [30]. On the constructive side, the same toolkit underpins auditing, safety verification, and scientific interpretability: researchers "reverse engineer" networks not to steal them but to understand the algorithms they have learned [23], [25]. This survey treats these two communities as studying a single problem from opposite ethical poles, and argues that progress in one necessarily informs the other.

We make three contributions. First, we propose a unifying taxonomy that situates every major attack along an access axis (black-box, gray-box, white-box) and a target axis (functional, parametric, data-centric, architectural). Second, we synthesize the LLM-

specific literature—prompt injection [19], jailbreaks [17], [18], and the 2024 demonstration that the final embedding-projection layer of production models can be extracted through ordinary API queries [11]. Third, we provide comparative tables, a historical timeline, and an enterprise-oriented risk discussion. Throughout, in keeping with the requirement to avoid fabricated sources, any citation we cannot verify with confidence is explicitly marked as a placeholder ([41], [42]).

II. BACKGROUND AND THREAT MODEL

2.1. Formalizing the Target

Let a deployed model be a parameterized function $f\theta: X \rightarrow Y$, where θ denotes the weights, X the input space, and Y the output (logits, probabilities, a top-1 label, or generated text). An adversary interacts through a query interface and observes some projection of the output. Reverse engineering seeks to recover one or more of: (i) a functional surrogate $f\theta\&$ that agrees with $f\theta$ on the input distribution; (ii) the parameters θ themselves (or a subset); (iii) elements of the training set D; or (iv) the architecture A that defines the computational graph [9], [31].

2.2. Access Models

Black-box access exposes only input–output behavior, possibly with confidence scores. Gray-box access adds partial knowledge—architecture family, tokenizer, or a public base model. White-box access exposes θ and gradients, as occurs with open-weight releases or insider compromise [26]. The access model dictates feasibility: gradient inversion [7] requires white-box gradients, whereas functionality stealing [33] needs only labels. A fourth, often-overlooked channel is physical/side-channel access—timing, cache, power, and electromagnetic emanations—which can betray architecture even when the logical interface is sealed [29], [30], [32].

2.3. Adversary Objectives and Cost

Objectives trade off along fidelity, accuracy, and query budget. Jagielski et al. [9] distinguish *accuracy extraction* (matching the victim's task performance) from *fidelity extraction* (matching the victim's exact decisions, including its errors). Fidelity is strictly harder and more valuable to an attacker performing transfer attacks, because a high-fidelity copy preserves the victim's decision boundary and therefore its adversarial vulnerabilities [4], [9].

III. LITERATURE REVIEW

The intellectual lineage of AI reverse engineering can be traced to two seminal observations in 2013–2014: Biggio et al. [27] showed that learned classifiers can be evaded at test time, and Szegedy et al. [2] revealed the existence of adversarial examples—imperceptible perturbations that flip predictions. Goodfellow et al. [1] explained these as a consequence of model linearity and introduced the fast gradient sign method (FGSM), establishing that the decision boundary of a network is itself an exploitable, recoverable object.

Model stealing as an explicit goal was formalized by Tramèr et al. [3] in 2016, who extracted logistic-regression, SVM, and shallow network parameters through equation-solving on prediction-API outputs. Papernot et al. [4] generalized this to deep networks via *substitute training*, showing that a local surrogate trained on victim-labeled synthetic data transfers adversarial examples back to the black-box target. The privacy strand matured in parallel: Fredrikson et al. [6] demonstrated model-inversion reconstruction of training inputs from confidence scores, and Shokri et al. [5] introduced shadow-model membership inference, proving that overfitting leaves a statistically detectable fingerprint of individual training records.

From 2019 onward the field bifurcated into ever-more-practical attacks and a maturing defense literature. Zhu et al. [7] showed *deep leakage from gradients*, reconstructing pixel-accurate images and tokens from shared gradients in federated learning, refined by Zhao et al. [8]. Orekondy et al. [33] (Knockoff Nets) and Correia-Silva et al. [34] (Copycat CNN) demonstrated functionality theft using non-problem-domain data. On hardware, Batina et al. [30] reverse-engineered network architecture through electromagnetic side channels, and Hu et al. [32] (DeepSniffer) reconstructed layer topology from memory-access traces. The LLM era then introduced qualitatively new surfaces, surveyed in Section 5, culminating in Carlini et al. [11], who in 2024 extracted the hidden-dimension and final projection matrix of production language models through their public APIs.

IV. TECHNICAL METHODOLOGIES

4.1. Model Extraction Attacks

Extraction reconstructs a surrogate $f\theta\&$ by querying the victim and training on the returned outputs. For a query set $Q = \{x_i\}$, the attacker minimizes a distillation objective:

$$\theta\& = \text{argmin}\theta\& \ \Sigma_i \, L(f\theta\&(x_i), f\theta(x_i)) \quad (1)$$

where L is cross-entropy on returned probabilities or a label-matching loss when only top-1 labels are exposed [3], [33]. Three regimes are distinguished. Exact/closed-form extraction applies to linear and piecewise-linear models, where a sufficient number of inputs–output pairs yield a solvable system [3]. Learning-based extraction trains a deep surrogate and is the dominant approach for modern networks [9], [33]. Cryptanalytic extraction exploits the ReLU structure: by probing where neurons flip sign, an attacker recovers exact weights layer by layer, an approach Carlini-style analyses extended to recover the final layer of LLMs [11].

Query efficiency is the binding constraint. Active-learning selection of informative queries, Jacobian-based augmentation [4], and the use of public unlabeled corpora reduce the budget by orders of magnitude. Carlini et al. [11] showed that the embedding-projection layer of a model with hidden dimension h can be recovered with O(h) carefully constructed logit-bias queries, exploiting the low-rank structure of the softmax output—a striking demonstration that API design choices (e.g., exposing top-k log-probabilities) directly determine extractability.

4.2. Adversarial Attacks as a Reverse-Engineering Primitive

Adversarial example generation is, at its core, local reverse engineering of the decision boundary. The projected gradient descent (PGD) attack [15] iterates

$$x_{t+1} = \Pi_{x+S}\left(x_t + \alpha \cdot \text{sign}\left(\nabla_x L(f_\theta(x_t), y)\right)\right) \quad (2)$$

where Π projects back onto an $\ell\infty$ ball of radius ε. In the white-box setting the gradient is exact; in the black-box setting it is estimated via finite differences or substitute models [4]. Carlini and Wagner [16] formulated stronger optimization-based attacks that remain a standard robustness benchmark. The transferability of adversarial examples [4] is itself evidence of structural similarity between independently trained models, making adversarial probing a diagnostic for how much of θ an attacker has implicitly recovered.

4.3. Gradient Inversion and Data Leakage

In federated and distributed training, clients share gradients $g = \nabla\theta \, L(f\theta(x), y)$ rather than raw data. Zhu et al. [7] showed this is insufficient for privacy: by optimizing a synthetic pair $(\hat{x}, \hat{y})$ to match the observed gradient,

$$(\hat{x}, \hat{y}) = \arg\min_{\hat{x},\hat{y}} \|\nabla_\theta L(f_\theta(\hat{x}), \hat{y}) - g\|^2 \quad (3)$$

the original training sample can be reconstructed, often pixel-accurately for small batches. Zhao et al. [8] showed the label can be recovered analytically from the sign of the final-layer gradient, simplifying the optimization. Nasr et al. [26] generalized white-box inference to active attackers who perturb shared parameters to amplify leakage. These results motivate secure aggregation and differential privacy [36] as countermeasures, though both impose utility costs.

4.4. Membership Inference and Model Inversion

Membership inference asks whether a specific record was in D. Shokri et al. [5] train shadow models to mimic the victim's behavior on known members/non-members, then train an attack classifier on the resulting confidence distributions. The signal is overfitting: members receive systematically higher confidence and lower loss. Model inversion [6] instead reconstructs a representative input for a class by gradient-ascending the class score. Both are amplified in over-parameterized regimes and are partially mitigated by differential privacy [36] and confidence masking [35], though Carlini et al. [10] showed that large language models memorize and regurgitate verbatim training sequences, a worst-case fusion of memorization and extraction.

4.5. Architecture Reconstruction and Side Channels

When the logical interface hides architecture, physical side channels reveal it. Cache Telepathy [31] uses shared-cache timing to infer matrix-multiplication dimensions, narrowing the search space of possible architectures by orders of magnitude. Batina et al. [30] recover layer count, sizes, and activation functions from electromagnetic emanations of a microcontroller running inference. DeepSniffer [32] learns to map memory-access and kernel-execution traces to layer topologies. These attacks establish that *architectural*

secrecy is not a durable defense once an adversary has physical or co-located access [29].

4.6. Weight Reconstruction

Full parametric recovery is the hardest target. For ReLU networks, the piecewise-linear structure permits exact recovery in principle: each linear region exposes a linear map, and boundaries between regions reveal neuron weights up to sign and scaling symmetries [9]. Jagielski et al. [9] achieved functionally equivalent extraction of two-layer networks and high-fidelity approximations of deeper ones. The 2024 LLM result [11] is the practical apex to date: rather than the full network, it surgically recovers the last layer—the most commercially diagnostic component, since its width equals the hidden dimension—demonstrating that partial weight reconstruction can be both cheap and damaging.

V. REVERSE ENGINEERING LARGE LANGUAGE MODELS

5.1. Transformers as the Target Architecture

The transformer [12] composes multi-head self-attention and position-wise feed-forward blocks. Reverse engineering a transformer therefore means recovering (i) configuration (layers, heads, hidden/MLP widths, vocabulary), (ii) weights, and (iii) the algorithms implemented by attention and MLP circuits. The first is partially exposed through tokenizer artifacts and the logit-projection extraction of [11]; the third is the province of mechanistic interpretability (Section 6).

5.2. Prompt Injection and Indirect Injection

Prompt injection treats the model's instruction-following as an attack surface: adversarial text overrides the developer's system prompt. Greshake et al. [19] introduced *indirect* prompt injection, in which malicious instructions are planted in third-party content (web pages, emails, documents) that an LLM-integrated application later ingests, compromising the application without the attacker ever touching the prompt directly. This is now codified as a top enterprise risk [39]. From a reverse-engineering standpoint, successful injection reveals the boundaries of the model's instruction hierarchy and, through systematic probing, leaks the hidden system prompt itself—an exfiltration of proprietary application logic.

5.3. Jailbreak Analysis

Jailbreaks circumvent safety alignment. Wei et al. [17] characterized two failure modes—*competing objectives* (helpfulness vs. harmlessness) and *mismatched generalization* (safety training failing to cover the input distribution). Zou et al. [18] then demonstrated *universal, transferable* adversarial suffixes discovered by gradient-based search on open models that transfer to closed ones—directly analogous to the substitute-model transfer of [4], but in token space. Jailbreak research is reverse engineering of the alignment policy: each successful break maps a region where the learned safety boundary diverges from the intended one.

VI. INTERPRETABILITY AND EXPLAINABLE AI: REVERSE ENGINEERING FOR UNDERSTANDING

Interpretability is reverse engineering pursued for science rather than theft. It spans two cultures. Post-hoc XAI explains individual predictions without claiming to recover internal mechanisms: saliency maps [20], Grad-CAM [21], LIME [13], SHAP [14], and integrated gradients [28] attribute a prediction to input features. These methods are attractive for enterprise compliance but are known to be fragile and sometimes misleading, since a faithful-looking attribution need not reflect the true computation.

Mechanistic interpretability instead aims to recover the algorithm itself. Feature visualization [22] reconstructs the inputs that maximally excite a neuron; the transformer-circuits program [23] reverse-engineers attention heads into composable algorithmic units (e.g., induction heads); ROME [24] localizes and edits factual associations to specific MLP layers, proving causal rather than merely correlational understanding; and dictionary-learning / sparse-autoencoder methods [25] decompose superposed, polysemantic activations into monosemantic features. This is the most direct white-box reverse engineering in the field—its target is identical to that of a weight-extraction adversary, but its purpose is auditing and safety [40].

The convergence is conceptually important: a model that can be mechanistically interpreted can, in principle, be more easily stolen, while a model hardened against extraction is correspondingly harder

to audit. This tension between *transparency* and *defensibility* is a recurring theme of Sections 7 and 9.

VII. COMPARATIVE ANALYSIS

Table I situates the principal reverse-engineering techniques against the unifying taxonomy of Section 2. Table II pairs each attack family with its leading defenses and the residual gap. Table III contrasts open-weight and proprietary deployment surfaces.

TABLE I. Taxonomy of AI Reverse-Engineering Techniques

Technique	Access	Target	Query/Signal	Key works
Model extraction	Black-box	Functionality / params	Labels, logits	[3],[9],[11],[33]
Adversarial / evasion	Black/White	Decision boundary	Gradients, transfer	[1],[4],[15],[16]
Gradient inversion	White-box	Training data	Shared gradients	[7],[8],[26]
Membership inference	Black-box	Data membership	Confidence/loss	[5],[26],[35]
Model inversion	Black/White	Class prototypes	Confidence scores	[6],[10]
Side-channel recon.	Physical	Architecture	EM / cache / power	[29],[30],[31],[32]
Prompt injection	Black-box	System prompt / logic	Crafted text	[19],[39]
Jailbreak analysis	Black/Gray	Alignment policy	Token-space search	[17],[18]
Mech. interpretability	White-box	Learned algorithms	Activations/ weights	[22],[23],[24],[25]

TABLE II. Attacks, Defenses, and Residual Gaps

Attack	Primary defenses	Mechanism	Residual gap
Extraction	Query monitoring; output truncation; watermarking [37],[38]	Limit signal; prove ownership	Stealthy low-rate stealing [11]
Adversarial	Adversarial training [15]; certified defenses	Robust optimization	Cost; adaptive attacks
Gradient inversion	Secure aggregation; DP-SGD [36]	Hide / perturb gradients	Utility–privacy trade-off
Membership inference	DP [36]; MemGuard [35]; regularization	Reduce overfit signal	LLM memorization [10]
Prompt injection	Input isolation; instruction hierarchy; filtering [39]	Separate data/instructions	Indirect injection [19]
Jailbreak	RLHF; adversarial fine-tuning; classifiers	Reinforce safety boundary	Transferable suffixes [18]

TABLE III. Open-Weight vs. Proprietary Model Exposure

Dimension	Open-weight	Proprietary (API-gated)
Weight access	Full → trivial white-box analysis	None directly; partial via [11]
Extraction needs	Unnecessary (already public)	Required; query-bounded
Auditability	High (interpretability feasible)	Low (black-box only)
Jailbreak research	Enables transferable attacks [18]	Beneficiary of that transfer
Defense posture	Watermark/licensing [37],[38]	Rate-limit, monitor, truncate

The comparison surfaces a paradox: open-weight models are maximally transparent and therefore maximally auditable, but also offer attackers a free white-box laboratory whose discoveries transfer to closed systems [18]. Proprietary models invert the trade-off—resistant to direct theft but opaque to external safety review, and still partially extractable [11]. No deployment posture is simultaneously private and auditable, which is the central governance dilemma of the field [40].

VIII. SECURITY AND ETHICAL CONCERNS

8.1. Intellectual Property and Provenance

Extraction directly threatens the capital embedded in a model. Defensive provenance techniques include backdoor-based watermarking [38], which embeds a verifiable trigger–response pair, and dataset inference [37], which proves ownership statistically from a suspected stolen model's knowledge of private data.

Both are reactive—they enable litigation after theft rather than preventing it—underscoring that prevention and attribution are distinct goals.

8.2. Privacy and Regulatory Exposure

Membership inference [5], gradient inversion [7], and verbatim memorization [10] each constitute a data-protection incident under regimes such as the GDPR, because they can re-identify or reconstruct personal data from a model. Differential privacy [36] offers a principled bound but degrades utility, and is rarely deployed at LLM scale. Enterprises that fine-tune foundation models on customer data inherit this liability directly.

8.3. Dual-Use and Responsible Disclosure

Every technique here is dual-use. Jailbreak discovery [17], [18] improves both attackers and red teams; interpretability [24], [25] aids both auditing and targeted manipulation. The community has largely converged on coordinated disclosure and on framing offensive research as a precondition for defense—mirrored in governance instruments such as the NIST AI Risk Management Framework [40] and the OWASP LLM Top 10 [39]. Ethical practice requires that demonstrations use consented or synthetic data and that extraction research target the authors' own models or public benchmarks.

IX. APPLICATIONS

Reverse engineering of AI is not solely adversarial; it underwrites several legitimate and increasingly mandated activities:

- Security auditing and red-teaming:

Systematically probing deployed models for extractable secrets, injectable prompts [19], and jailbreak paths [18] before adversaries do.

- Regulatory compliance and XAI:

Attribution methods [13], [14], [28] and mechanistic analysis [24] supply the explanations required by emerging AI regulation [40].

- Model provenance and forensics:

Watermark verification [38] and dataset inference [37] to detect stolen or unlicensed models.

- Safety verification:

Interpretability to certify that a model does not implement deceptive or unsafe internal circuits [23], [25].

- Benchmarking and scientific understanding:

Recovering learned algorithms to advance the science of deep learning rather than to exploit it [22], [23].

X. CHALLENGES AND LIMITATIONS

Several obstacles bound the field. Scale: exact weight recovery is tractable for shallow ReLU networks [9] but combinatorially hard for billion-parameter transformers; only partial recovery has been demonstrated [11]. Symmetry and identifiability: permutation and scaling symmetries mean many parameter sets realize the same function, so "the" weights are recoverable only up to equivalence. Faithfulness of explanations: post-hoc XAI can mislead, and there is no agreed metric for explanation faithfulness. Defense evaluation: many defenses fall to adaptive attacks [16], so static benchmarks overstate security. Reproducibility and access: the most capable proprietary models are unavailable for white-box study, forcing reliance on open proxies whose transfer validity is assumed rather than proven [18]. Finally, the autonomous-agent setting (Section 11) is largely *uncharted*—a gap we flag explicitly with a placeholder citation [41] rather than overstate the literature.

XI. FUTURE RESEARCH DIRECTIONS

11.1.Reverse Engineering Autonomous AI Agents

LLM-based agents that plan, call tools, and act over multiple steps present a new and under-studied surface. An agent's observable trajectory—its tool calls, intermediate reasoning, and memory writes—leaks its internal policy and prompt scaffolding far more than a single-shot model. Indirect prompt injection [19] becomes a control-hijacking vector when the agent has real-world side effects. We anticipate a research program on *trajectory-based policy extraction* and *agent fingerprinting*, but note that, as of this writing, the peer-reviewed literature is thin; we therefore mark this direction with a placeholder reference [41] rather than cite work we cannot verify.

11.2.Scalable Mechanistic Interpretability

Sparse-autoencoder feature decomposition [25] and circuit analysis [23] must scale from toy and mid-size models to frontier systems, ideally automated. Success would simultaneously advance safety auditing and, as a side effect, lower the cost of structural extraction—a tension future governance must address.

11.3.Provable Defenses and Privacy

Differential privacy [36] and secure aggregation need efficiency gains to be practical at LLM scale, and watermarking [38] must survive distillation and fine-tuning to remain a viable provenance signal against extraction [11].

11.4.Reverse Engineering Toward AGI-Scale Systems

As systems approach broad autonomy, reverse engineering shifts from an IP and privacy concern to a *safety necessity*: verifying that a powerful system's internal objectives match its specified ones may require interpretability tools far beyond today's [23], [25], [40]. The field's offensive and defensive halves converge here—the ability to read a model's mind is both the ultimate attack and the ultimate safeguard. A comparative empirical study of extraction risk across the open/closed frontier is needed; lacking a verified citation, we mark it as a placeholder [42].

XII. A TIMELINE OF AI REVERSE ENGINEERING

TABLE IV. Selected Milestones (2013–2024)

Year	Milestone	Ref.
2013–14	Evasion attacks; adversarial examples discovered	[27],[2],[1]
2015	Model inversion from confidence scores	[6]
2016	Model stealing via prediction APIs; DP-SGD	[3],[36]
2017	Membership inference; substitute black-box attacks; transformers; SHAP/Grad-CAM	[5],[4],[12],[14],[21]
2018–19	EM/cache side-channel architecture recovery; Knockoff/Copycat; white-box inference	[30],[31],[33],[34],[26]
2019–20	Deep leakage from gradients; high-fidelity extraction; DeepSniffer	[7],[8],[9],[32]
2021–22	Training-data extraction from LLMs; transformer circuits; ROME	[10],[23],[24]
2023	Jailbreak taxonomy; universal suffixes; indirect prompt injection; monosemanticity; OWASP LLM Top 10	[17],[18],[19],[25],[39]
2024	Extraction of production LLM projection layer through public APIs	[11]

XIII. CONCLUSION

Reverse engineering has matured from a curiosity about adversarial examples into a comprehensive discipline spanning functionality theft, parameter recovery, data reconstruction, architecture inference, and —most recently—the probing of large language models and the algorithms inside them. The unifying view advanced here, organizing the field by access and target, exposes deep connections: substitute-model transfer [4] and transferable jailbreaks [18] are the same idea in different spaces; weight extraction [11] and mechanistic interpretability [25] pursue the same artifact for opposite ends. The decisive open problems—scaling interpretability, defending without sacrificing auditability, and understanding autonomous agents—are precisely those where offensive and defensive interests converge. We have been deliberately conservative with citations, marking unverified claims as placeholders [41], [42], and we encourage readers to substitute verified, current sources before publication. The trajectory is clear: as AI systems grow more capable and more autonomous, the ability to reverse engineer them—responsibly—will be indispensable both to those who would exploit them and to those who must keep them safe.

REFERENCES

[1] Goodfellow, I. J., J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015.

[2] Szegedy, C., et al., "Intriguing properties of neural networks," in Proc. Int. Conf. Learn. Represent. (ICLR), 2014.
[3] Tramèr, F., F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in Proc. 25th USENIX Security Symp., 2016, pp. 601–618.
[4] Papernot, N., P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in Proc. ACM Asia Conf. Comput. Commun. Security (ASIACCS), 2017, pp. 506–519, doi: 10.1145/3052973.3053009.
[5] Shokri, R., M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in Proc. IEEE Symp. Security and Privacy (S&P), 2017, pp. 3–18, doi: 10.1109/SP.2017.41.
[6] Fredrikson, M., S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in Proc. ACM SIGSAC Conf. Comput. Commun. Security (CCS), 2015, pp. 1322–1333, doi: 10.1145/2810103.2813677.
[7] Zhu, L., Z. Liu, and S. Han, "Deep leakage from gradients," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 32, 2019.
[8] Zhao, B., K. R. Mopuri, and H. Bilen, "iDLG: Improved deep leakage from gradients," arXiv:2001.02610, 2020.
[9] Jagielski, M., N. Carlini, D. Berthelot, A. Kurakin, and N. Papernot, "High accuracy and high-fidelity extraction of neural networks," in Proc. 29th USENIX Security Symp., 2020, pp. 1345–1362.
[10] Carlini, N., et al., "Extracting training data from large language models," in Proc. 30th USENIX Security Symp., 2021, pp. 2633–2650.
[11] Carlini, N., et al., "Stealing part of a production language model," in Proc. 41st Int. Conf. Mach. Learn. (ICML), 2024.
[12] Vaswani, A., et al., "Attention is all you need," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 30, 2017.
[13] Ribeiro, M. T., S. Singh, and C. Guestrin, "'Why should I trust you?': Explaining the predictions of any classifier," in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD), 2016, pp. 1135–1144, doi: 10.1145/2939672.2939778.
[14] Lundberg, S. M., and S.-I. Lee, "A unified approach to interpreting model predictions," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 30, 2017.
[15] Madry, A., A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in Proc. Int. Conf. Learn. Represent. (ICLR), 2018.
[16] Carlini, N., and D. Wagner, "Towards evaluating the robustness of neural networks," in Proc. IEEE Symp. Security and Privacy (S&P), 2017, pp. 39–57, doi: 10.1109/SP.2017.49.
[17] Wei, A., N. Haghtalab, and J. Steinhardt, "Jailbroken: How does LLM safety training fail?" in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 36, 2023.
[18] Zou, A., Z. Wang, N. Carlini, M. Nasr, J. Z. Kolter, and M. Fredrikson, "Universal and transferable adversarial attacks on aligned language models," arXiv:2307.15043, 2023.
[19] Greshake, K., S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, "Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection," in Proc. 16th ACM Workshop Artif. Intell. Security (AISec), 2023, doi: 10.1145/3605764.3623985.
[20] Simonyan, K., A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," in Proc. Int. Conf. Learn. Represent. (ICLR) Workshop, 2014.
[21] Selvaraju, R. R., M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), 2017, pp. 618–626, doi: 10.1109/ICCV.2017.74.
[22] Olah, C., A. Mordvintsev, and L. Schubert, "Feature visualization," Distill, 2017, doi: 10.23915/distill.00007.
[23] Elhage, N., et al., "A mathematical framework for transformer circuits," Transformer Circuits Thread, Anthropic, 2021.
[24] Meng, K., D. Bau, A. Andonian, and Y. Belinkov, "Locating and editing factual associations in GPT," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 35, 2022.

[25] Bricken, T., et al., "Towards monosemanticity: Decomposing language models with dictionary learning," Transformer Circuits Thread, Anthropic, 2023.

[26] Nasr, M., R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in Proc. IEEE Symp. Security and Privacy (S&P), 2019, pp. 739–753, doi: 10.1109/SP.2019.00065.

[27] Biggio, B., et al., "Evasion attacks against machine learning at test time," in Proc. ECML PKDD, 2013, pp. 387–402, doi: 10.1007/978-3-642-40994-3_25.

[28] Sundararajan, M., A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in Proc. 34th Int. Conf. Mach. Learn. (ICML), 2017, pp. 3319–3328.

[29] Hong, S., et al., "Security analysis of deep neural networks operating in the presence of cache side-channel attacks," arXiv:1810.03487, 2018.

[30] Batina, L., S. Bhasin, D. Jap, and S. Picek, "CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel," in Proc. 28th USENIX Security Symp., 2019, pp. 515–532.

[31] Yan, M., C. W. Fletcher, and J. Torrellas, "Cache telepathy: Leveraging shared resource attacks to learn DNN architectures," in Proc. 29th USENIX Security Symp., 2020, pp. 2003–2020.

[32] Hu, X., et al., "DeepSniffer: A DNN model extraction framework based on learning architectural hints," in Proc. 25th Int. Conf. Architectural Support Program. Lang. Oper. Syst. (ASPLOS), 2020, pp. 385–399, doi: 10.1145/3373376.3378460.

[33] Orekondy, T., B. Schiele, and M. Fritz, "Knockoff nets: Stealing functionality of black-box models," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2019, pp. 4954–4963, doi: 10.1109/CVPR.2019.00509.

[34] Correia-Silva, J. R., R. F. Berriel, C. Badue, A. F. de Souza, and T. Oliveira-Santos, "Copycat CNN: Stealing knowledge by persuading confession with random non-labeled data," in Proc. Int. Joint Conf. Neural Netw. (IJCNN), 2018, doi: 10.1109/IJCNN.2018.8489592.

[35] Jia, J., A. Salem, M. Backes, Y. Zhang, and N. Z. Gong, "MemGuard: Defending against black-box membership inference attacks via adversarial examples," in Proc. ACM SIGSAC Conf. Comput. Commun. Security (CCS), 2019, pp. 259–274, doi: 10.1145/3319535.3363201.

[36] Abadi, M., et al., "Deep learning with differential privacy," in Proc. ACM SIGSAC Conf. Comput. Commun. Security (CCS), 2016, pp. 308–318, doi: 10.1145/2976749.2978318.

[37] Maini, P., M. Yaghini, and N. Papernot, "Dataset inference: Ownership resolution in machine learning," in Proc. Int. Conf. Learn. Represent. (ICLR), 2021.

[38] Adi, Y., C. Baum, M. Cisse, B. Pinkas, and J. Keshet, "Turning your weakness into a strength: Watermarking deep neural networks by backdooring," in Proc. 27th USENIX Security Symp., 2018, pp. 1615–1631.

[39] OWASP Foundation, "OWASP Top 10 for Large Language Model Applications," 2023–2025. [Online]. Available: https://owasp.org/www-project-top-10-for-large-language-model-applications/

[40] NIST, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," NIST AI 100-1, Jan. 2023, doi: 10.6028/NIST.AI.100-1.

Appendix A. Suggested Figures, Architecture, and Project Ideas

A.1 Suggested Diagrams / Figures

- Fig. 1 — Taxonomy map: a 2-D grid with the access axis (black/gray/white/physical) on one side and the target axis (functionality/parameters/data/architecture) on the other, with each technique plotted as a point.
- Fig. 2 — Model-extraction pipeline: victim API → query strategy → surrogate training → fidelity/accuracy evaluation, with the distillation loss of Eq. (1) annotated.
- Fig. 3 — Gradient-inversion loop: shared gradient → optimization of synthetic ($\hat{x}$, $\hat{y}$) → reconstructed sample, per Eq. (3).
- Fig. 4 — Transformer reverse-engineering stack: configuration recovery → logit-projection extraction [11] → circuit/feature interpretation [23],[25].
- Fig. 5 — Attack–defense Sankey diagram linking each attack family (Table II) to its defenses and

residual gaps.
- Fig. 6 — Timeline ribbon visualizing Table IV.

A.2 Suggested Implementation Architecture (Auditing Toolkit)

A practical, defensively-oriented reverse-engineering toolkit could be structured in four layers:

1) Probe layer: a query orchestrator (rate-limited, logged) supporting black-box label/logit collection, gradient access for owned models, and side-channel capture hooks.
2) Attack-module layer: pluggable implementations of extraction, membership inference, gradient inversion, and prompt-injection/jailbreak fuzzers, each emitting a standardized risk report.
3) Interpretability layer: saliency/SHAP for quick attributions and sparse-autoencoder + circuit tooling for deep mechanistic audits on white-box targets.
4) Governance layer: watermark verification, dataset-inference provenance checks, and a compliance dashboard mapping findings to NIST AI RMF [40] and OWASP LLM [39] controls.

A.3 Derived Project Ideas

- Query-efficient extraction benchmark: measure how API design (top-k logprobs, logit bias) changes extractability, replicating and extending [11] on open models.
- Indirect-prompt-injection scanner for LLM agents: instrument a tool-using agent and detect when ingested content hijacks its policy [19].
- Watermark robustness study: test whether backdoor watermarks [38] survive distillation-based extraction and fine-tuning.
- Automated sparse-autoencoder feature labeling: scale monosemantic feature discovery [25] with LLM-assisted naming and causal validation via activation patching [24].
- Privacy-utility frontier for fine-tuned enterprise LLMs: quantify membership-inference risk [5] vs. DP-SGD [36] cost on customer-data fine-tunes.