

Compiler-Assisted Optimization of Privacy-Preserving Neural Network Architectures

Lara Commins

Department of Computer Science and Engineering, University at Buffalo, Buffalo, NY, USA.
lara1974@buffalo.edu

Abstract

Privacy-preserving neural network inference has moved from a narrow cryptographic curiosity to a systems problem that spans compilers, runtime schedulers, cryptographic protocols, hardware accelerators, and governance regimes. The central challenge is not simply whether a neural network can be evaluated without exposing inputs or weights, but whether such evaluation can be performed with acceptable latency, cost, accuracy, robustness, and accountability. Compiler-assisted optimization offers a promising response by treating privacy constraints as first-class properties of the computation graph. Instead of applying generic neural network optimizations and then fitting them into a cryptographic protocol, compiler-assisted methods can restructure architectures, lower operator representations, reduce communication rounds, and co-schedule secure computation primitives according to deployment context. This paper examines that emerging design space from a system-level perspective. It discusses how optimizing compilers can bridge secure multi-party computation, homomorphic encryption, and trusted execution environments with neural architecture search, quantization, activation reduction, operator fusion, and memory planning. The analysis emphasizes structural trade-offs rather than narrow benchmark gains. Particular attention is given to the ways privacy-preserving compilation affects fairness, robustness, sustainability, auditability, and policy compliance. The paper argues that compiler-assisted optimization should be understood as an infrastructural and governance instrument, not merely a performance technique. Its long-term value depends on whether it can produce transparent, verifiable, and deployable systems that preserve privacy without shifting unacceptable costs onto operators, users, or the public.

Keywords

privacy-preserving machine learning; compiler optimization; secure multi-party computation; homomorphic encryption; neural architecture; system governance; deployment sustainability; fairness.

1. Introduction

Privacy-preserving neural network inference has become a critical research area because machine learning systems increasingly process sensitive data in domains such as healthcare, finance, public administration, and personal computing. Early work in secure inference demonstrated that neural network predictions could be computed over encrypted or secret-shared data using cryptographic protocols, but the resulting systems often imposed severe latency and communication costs. Frameworks such as ABY3, GAZELLE, MiniONN, Chameleon, and DeepSecure established foundational techniques for secure evaluation, including mixed-protocol design, oblivious transformations, and specialized arithmetic for matrix multiplication and nonlinear activation functions [1], [2], [3], [4], [5]. These contributions made privacy-preserving inference feasible in principle, yet they also revealed

that cryptographic feasibility alone is insufficient for real deployment. The operational burden of secure computation is closely tied to neural architecture, data layout, network topology, and the compiler decisions that translate high-level models into protocol-specific instructions.

Compiler-assisted optimization has emerged as a response to this gap. A privacy-preserving compiler can analyze a neural network graph and determine where linear operations should be executed under homomorphic encryption, where secret sharing may be more efficient, where garbled circuits are unavoidable, and where trusted hardware might be appropriate if the threat model permits it. Rather than treating cryptographic protocols as opaque backends, such compilers expose protocol costs to higher-level transformations. This allows optimizations to be selected not only for arithmetic efficiency but also for communication round complexity, memory pressure, key management overhead, and deployment constraints. The result is a more integrated view of privacy-preserving architecture, in which model design and protocol scheduling are co-optimized rather than sequentially engineered.

This paper argues that compiler-assisted optimization is best understood as a socio-technical infrastructure. Its technical choices influence who can deploy privacy-preserving inference, under what cost structure, with what level of assurance, and subject to what forms of audit. Compiler decisions about activation functions, quantization, batching, and operator fusion are not neutral. They affect accuracy for underrepresented groups, robustness against adversarial manipulation, energy consumption, and the practical ability of regulators to inspect system behavior. The discussion therefore proceeds from system architecture to governance implications, considering how compiler design can support or undermine fairness, sustainability, and public accountability.

2. Background and System Context

Privacy-preserving neural network inference draws on several cryptographic families. Secure multi-party computation allows multiple parties to jointly evaluate a function without revealing their private inputs to one another. Homomorphic encryption allows computation directly over encrypted data, typically at the cost of large ciphertexts and limited nonlinear operations. Trusted execution environments provide hardware-isolated computation, but they depend on vendor trust, attestation, and side-channel resistance. Each approach has distinct performance characteristics and threat models. CryptoNets demonstrated early homomorphic neural network inference with high throughput and accuracy under encryption, while later systems such as CHET and EVA introduced compiler-level abstractions for fully homomorphic neural network evaluation [6], [7], [8]. These systems showed that compiler optimization is essential for managing the gap between high-level neural network semantics and low-level cryptographic operations.

Secure multi-party computation frameworks have similarly evolved toward compiler-like abstractions. CrypTen and related libraries provide tensor-level programming interfaces that hide some protocol complexity, while PySyft and federated learning systems emphasize distributed data governance [10], [12], [13]. Practical secure aggregation enables privacy-preserving model updates in federated settings [14]. However, these systems often remain difficult to optimize end to end because neural network operators, communication patterns, and cryptographic primitives interact in complex ways. A convolution may be efficient in one protocol but wasteful in another. A rectified linear unit may be cheap in plaintext but expensive under encryption because it requires comparison or approximation. A batch normalization layer may be folded into adjacent linear operations in one backend but require separate secure computation in another.

Compiler-assisted optimization addresses this complexity by introducing intermediate representations that preserve both neural network structure and privacy cost information. Such representations can annotate tensors with secrecy levels, permitted leakage, protocol compatibility, and communication cost. They can also encode deployment constraints such as bandwidth limits, trusted hardware availability, regulatory jurisdiction, and latency service-level objectives. In this sense, the compiler becomes a mediator between model designers, cryptographic engineers, system operators, and policy requirements. The background of privacy-preserving inference is therefore not only a history of cryptographic advances but also a history of abstraction layers that attempt to make secure computation tractable for mainstream machine learning engineering.

3. Compiler-Assisted Optimization Stack

A privacy-preserving compiler typically begins with a neural network graph and lowers it through several stages. The front end may accept models from common frameworks and translate them into a graph representation. The middle end applies architecture-level transformations such as constant folding, operator fusion, quantization, pruning, and activation substitution. The back end maps operations to protocol-specific primitives, schedules communication rounds, and allocates memory. In homomorphic encryption compilers such as CHET and EVA, the middle end must manage ciphertext packing, rotation, and noise growth, while the back end must select parameter sets and evaluate circuits efficiently [7], [8]. In secure multi-party computation compilers, the back end may choose between arithmetic sharing, Boolean sharing, garbled circuits, and oblivious transfer depending on operator type and network conditions.

The most consequential compiler decisions often concern nonlinear operations. Neural networks rely on activation functions, normalization, pooling, and attention mechanisms that are not naturally compatible with additive or multiplicative homomorphic operations. Secure multi-party computation can evaluate these functions, but the cost may dominate the entire inference. Compiler-assisted approaches therefore seek to reduce the number of expensive nonlinear operations through architectural transformation. Some methods train networks with fewer rectified linear units, replace them with polynomial approximations, or shift nonlinearity into lower-cost protocol phases [11]. This is not merely a model compression problem. It is a co-design problem in which training objectives, compiler constraints, and protocol costs must be aligned. A network that is accurate in plaintext may be unsuitable for secure inference if its activation pattern creates excessive communication or requires many comparison gates.

Operator fusion and memory planning are equally important. Fusing consecutive linear layers can reduce round trips between parties, while fusing normalization into convolution can eliminate separate secure operations. Memory planning matters because secure computation often operates under tight bandwidth and storage constraints. Intermediate ciphertexts or secret shares can consume substantial memory, especially in large batch settings. A compiler that understands protocol-level lifetimes can reuse buffers, reorder computation, and avoid materializing unnecessary intermediates. Such optimizations may appear mundane, but they determine whether privacy-preserving inference can run on edge devices, within regulated enterprise environments, or under real-time service constraints.

Compiler-assisted optimization also raises questions of verifiability. If a compiler rewrites a model to reduce activation cost, how can an auditor know that the rewritten model still satisfies fairness, safety, or accuracy requirements? If the compiler selects a protocol based on

runtime conditions, how can a regulator reconstruct the privacy guarantees that were actually enforced? These questions suggest that privacy-preserving compilers should emit not only executable artifacts but also machine-readable evidence about transformations, assumptions, and residual leakage. Without such evidence, compiler optimization may improve performance while weakening accountability. The optimization stack therefore has a governance dimension from the outset.

4. Architectural Trade-Offs in Privacy-Preserving Neural Networks

Privacy-preserving neural architecture design involves unavoidable trade-offs among accuracy, latency, communication, memory, security, and deployability. Homomorphic encryption can support high-throughput inference in some settings, but deep networks with many nonlinear layers may require expensive bootstrapping or approximation. Secure multi-party computation can evaluate arbitrary functions, but communication rounds grow with network depth and nonlinearity. Trusted execution can be fast, but it introduces hardware trust assumptions and side-channel risks. No single architecture dominates across all contexts. Compiler-assisted optimization helps navigate these trade-offs by exposing them explicitly and allowing deployment-specific choices. For example, a latency-sensitive mobile application may prefer a shallow quantized model with a small number of secure comparison operations, while a batch-oriented cloud service may prioritize throughput and tolerate larger ciphertexts [2], [4].

Activation functions illustrate the tension between model expressiveness and cryptographic cost. Rectified linear units are popular because they train well and avoid vanishing gradients, but their comparison-based nature is expensive in secure computation. Alternative activations such as polynomials, rational functions, or smooth approximations may be cheaper under homomorphic encryption, yet they can change training dynamics and reduce accuracy on rare classes. Compiler-assisted methods can search for activation substitutions that meet both accuracy and protocol constraints, but such searches must consider fairness. A transformation that preserves average accuracy may disproportionately harm subgroups whose data distributions depend on sharp nonlinear boundaries. This is a system-level risk, not merely a modeling detail.

Another trade-off concerns batching and parallelism. Secure inference often benefits from batching because fixed cryptographic overhead can be amortized across many examples. However, batching increases memory pressure and may conflict with privacy requirements that discourage linking multiple user requests. Federated and distributed settings introduce additional constraints because data cannot be centrally batched. Secure aggregation can protect model updates, but inference over distributed inputs may require different protocol schedules [14]. Compiler-assisted optimization can adapt batch size, packing strategy, and communication topology to the deployment context, but it must also respect governance rules about data minimization and purpose limitation. A compiler that silently increases batching to improve throughput may inadvertently create a linkage risk.

Security assumptions are equally important. A compiler may optimize for a semi-honest adversary, a malicious adversary, or a covert adversary. These choices affect protocol selection and performance. Optimizations that are safe under semi-honest assumptions may be vulnerable when participants deviate from the protocol. Similarly, techniques that rely on trusted hardware may not be acceptable in jurisdictions with strict procurement or sovereignty requirements. The architecture of a privacy-preserving neural network is therefore inseparable from its threat model and institutional context. Compiler-assisted optimization can assist by

making threat models explicit and by refusing transformations that violate declared assumptions. This is a form of safety enforcement at the compilation layer [15].

5. Governance, Deployment, and Policy Implications

The governance of privacy-preserving machine learning extends beyond cryptographic guarantees. Regulations such as the General Data Protection Regulation and emerging artificial intelligence legislation emphasize lawfulness, fairness, transparency, and accountability. Privacy-preserving inference can support data protection by minimizing exposure of personal data, but it does not automatically satisfy legal requirements. A system may process encrypted data yet still make discriminatory decisions, lack meaningful explanation, or evade oversight because its internal operations are opaque. Compiler-assisted optimization can contribute to governance by producing auditable transformation records, documenting protocol choices, and enabling reproducible builds of privacy-preserving models [16], [17], [18], [19]. However, these benefits are not automatic. If compilers are proprietary or poorly documented, they may obscure rather than clarify accountability.

Deployment also raises questions of infrastructure and capacity. Privacy-preserving inference often requires specialized hardware, high-bandwidth networks, and cryptographic key management. Organizations with fewer resources may struggle to adopt such systems, leading to a privacy divide in which only well-funded institutions can offer strong protections. Compiler-assisted optimization could reduce barriers by generating efficient code for commodity hardware and by automating protocol selection. Yet it could also concentrate power in the hands of platform providers that control the most advanced compilers and accelerators. Policy interventions may be needed to promote interoperability, open standards, and public-interest infrastructure. Otherwise, privacy-preserving machine learning may become a premium feature rather than a baseline expectation.

Regulatory auditing of privacy-preserving systems is challenging because the relevant evidence is distributed across model architecture, compiler transformations, cryptographic parameters, and runtime logs. A regulator cannot simply inspect inputs and outputs. It may need to verify that a compiler enforced a declared privacy budget, that protocol parameters matched legal requirements, and that optimization did not introduce prohibited inference. This suggests a role for certified compilation pipelines that produce verifiable claims about privacy, accuracy, and fairness. Such pipelines could be analogous to safety-certified compilers in aviation or medical devices, where transformations are constrained by standards and accompanied by evidence. The policy implication is that compiler design should be treated as part of the compliance surface, not as an implementation detail hidden behind an application programming interface.

6. Robustness, Fairness, and Sustainability

Robustness and fairness are often treated as separate from privacy, but compiler-assisted optimization connects them. Adversarial examples can exploit quantization, approximation, and activation substitution introduced to reduce cryptographic cost. A model that is robust in plaintext may become fragile after compiler-driven transformation. Similarly, fairness metrics can shift when nonlinearities are approximated or when layers are fused in ways that change numerical precision. These effects are especially concerning in high-stakes domains such as healthcare and criminal justice, where privacy protections are necessary but not sufficient. Compiler-assisted optimization should therefore include robustness and fairness constraints in its search space, not as afterthoughts. A privacy-preserving compiler could reject

transformations that exceed subgroup error thresholds or that increase vulnerability to known attacks [15], [16], [17].

Sustainability is another often overlooked dimension. Privacy-preserving inference is computationally expensive. Homomorphic encryption and secure multi-party computation can require orders of magnitude more energy than plaintext inference, depending on the protocol and network configuration. If privacy-preserving systems are deployed at scale, their environmental footprint becomes a system-level concern. Compiler-assisted optimization can reduce energy use by eliminating redundant operations, selecting efficient protocols, and scheduling computation to match available resources. However, energy efficiency must be measured across the entire lifecycle, including model training, key generation, communication, and hardware manufacture. A narrow focus on inference latency may hide substantial upstream or downstream costs [20]. Sustainable privacy engineering requires transparency about these trade-offs and incentives for efficient design.

Fairness and sustainability also intersect with deployment equity. Resource-constrained organizations may be unable to run energy-intensive privacy-preserving inference, which can exclude them from providing strong privacy protections. Compiler-assisted optimization that lowers resource requirements can help democratize access, but only if the resulting tools are available under appropriate licenses and supported by open standards. Otherwise, the benefits accrue to large platforms while smaller actors face higher costs or weaker privacy. This is a governance issue as much as a technical one. The design of compilers, their licensing, and their integration with public infrastructure will shape who can participate in privacy-preserving machine learning.

7. Case Illustrations and Cross-Domain Comparisons

Healthcare offers a clear case for privacy-preserving neural network inference. Hospitals may wish to use a centralized diagnostic model without transmitting patient data. Secure multi-party computation and homomorphic encryption can theoretically enable such use, but clinical workflows demand low latency, high reliability, and auditability. Compiler-assisted optimization can adapt model architectures to available cryptographic backends and produce evidence for compliance. However, medical data are heterogeneous and often imbalanced, so approximations that help performance may harm rare disease detection. A compiler that only optimizes average accuracy is insufficient. It must support constraints on sensitivity, subgroup performance, and clinician interpretability [3], [5].

Financial services provide another instructive domain. Fraud detection and credit scoring involve sensitive financial records and regulatory obligations. Privacy-preserving inference can help institutions collaborate without sharing raw customer data, but adversarial actors may attempt to infer information through repeated queries. Compiler-assisted optimization can reduce leakage by limiting precision, batching queries, or enforcing differential privacy at the interface. Yet these choices interact with business requirements and legal duties. A model that is too noisy may fail to detect fraud, while one that is too precise may expose information. The compiler becomes a site where privacy, accuracy, and compliance are negotiated.

Edge and mobile deployments introduce different constraints. Devices have limited memory, battery, and network bandwidth. Trusted execution environments may be available, but their security properties vary. Federated learning can keep raw data local, but secure inference over local data may still require communication with a server. Compiler-assisted optimization can generate code for heterogeneous edge hardware, select lightweight protocols, and schedule

computation to avoid battery drain. Cross-domain comparison shows that there is no universal privacy-preserving architecture. The appropriate design depends on threat model, data sensitivity, regulatory context, hardware, and user expectations. Compiler-assisted optimization is valuable precisely because it can specialize to these contexts while maintaining a common abstraction layer [10], [12], [13].

8. Future Directions

Future research should pursue several directions. First, privacy-preserving compilers need richer cost models that go beyond latency and throughput to include communication energy, memory pressure, fairness risk, and auditability. Second, compiler transformations should be formally verified or at least accompanied by certificates that can be checked by third parties. Third, architectures should be co-designed with compilers from the beginning, rather than treating compilation as a post-training step. This may involve differentiable proxies for protocol cost during training. Fourth, governance frameworks should recognize compilation as a compliance-relevant activity and require documentation of transformations, assumptions, and residual risks [1], [7], [11], [19].

Another direction concerns interoperability. If every privacy-preserving framework uses its own intermediate representation and protocol assumptions, deployment will remain fragmented. Open standards for privacy-preserving computation graphs could allow models to migrate across backends and enable independent auditing. Such standards must balance expressiveness with verifiability. They must also address versioning, since cryptographic parameters and hardware capabilities change over time. A model compiled today may not be secure or efficient on tomorrow's infrastructure. Compiler toolchains should therefore support reproducible builds and long-term maintenance.

Finally, research should attend to the political economy of privacy-preserving infrastructure. Who controls the compilers, the accelerators, and the key management services? Who bears the cost of energy and latency? Who can audit the system? These questions are not external to technical design. They shape which optimizations are developed, which benchmarks are prioritized, and which users are served. Compiler-assisted optimization can advance privacy, fairness, and sustainability, but only if it is guided by public-interest values as well as performance metrics [14], [15], [20].

9. Conclusion

Compiler-assisted optimization of privacy-preserving neural network architectures represents a convergence of cryptography, machine learning systems, and governance. It moves beyond the question of whether secure inference is possible to ask how it can be made deployable, equitable, robust, and accountable. By treating privacy constraints as first-class properties of the computation graph, compilers can enable architecture-level transformations, protocol selection, and deployment-specific specialization that would be difficult to achieve manually. Yet these capabilities also introduce new risks. Optimizations can alter fairness, weaken robustness, increase energy consumption, or obscure accountability if they are not transparent and verifiable. The design of privacy-preserving compilers is therefore not merely a technical exercise. It is an act of infrastructure building that shapes the distribution of privacy, cost, and power in machine learning systems. Future work should integrate formal verification, fairness constraints, sustainability metrics, and policy compliance into the compilation pipeline from the outset. Only then can privacy-preserving neural networks become trustworthy public infrastructure rather than niche cryptographic demonstrations.

References

1. Mohassel, P., & Rindal, P. (2018). ABY3: A mixed protocol framework for machine learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (pp. 35–52). Association for Computing Machinery.
2. Juvekar, C., Vaikuntanathan, V., & Chandrakasan, A. (2018). GAZELLE: A low latency framework for secure neural network inference. In *Proceedings of the 27th USENIX Security Symposium* (pp. 1651–1669). USENIX Association.
3. Liu, J., Juuti, M., Lu, Y., & Asokan, N. (2017). Oblivious neural network predictions via MiniONN transformations. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (pp. 619–631). Association for Computing Machinery.
4. Riazi, M. S., Weinert, C., Tkachenko, O., Songhori, E. M., Schneider, T., & Koushanfar, F. (2018). Chameleon: A hybrid secure computation framework for machine learning applications. In *Proceedings of the 2018 Asia Conference on Computer and Communications Security* (pp. 707–721). Association for Computing Machinery.
5. Rouhani, B. D., Riazi, M. S., & Koushanfar, F. (2018). DeepSecure: Scalable provably-secure deep learning. In *Proceedings of the 55th Annual Design Automation Conference* (Article 2). Association for Computing Machinery.
6. Gilad-Bachrach, R., Dowlin, N., Laine, K., Lauter, K., Naehrig, M., & Wernsing, J. (2016). CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy. In *Proceedings of the 33rd International Conference on Machine Learning* (pp. 201–210). PMLR.
7. Dathathri, R., Saarikivi, O., Chen, H., Laine, K., Lauter, K., Maleki, S., & Musuvathi, M. (2019). CHET: An optimizing compiler for fully-homomorphic neural-network inferencing. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation* (pp. 142–156). Association for Computing Machinery.
8. Dathathri, R., Kostova, B., Saarikivi, O., Dai, W., Laine, K., & Musuvathi, M. (2020). EVA: An encrypted vector arithmetic language and compiler for efficient homomorphic computation. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation* (pp. 546–561). Association for Computing Machinery.
9. Lee, E., Lee, J.-W., Lee, J., Kim, Y., Kim, Y., No, J.-S., & Choi, W. (2020). Low-complexity deep convolutional neural networks on fully homomorphic encryption using multiplexed parallel convolutions. *arXiv preprint arXiv:2008.12171*.
10. Knott, B., Venkataraman, S., Hannun, A., Sengupta, S., Ibrahim, M., & van der Maaten, L. (2021). CrypTen: Secure multi-party computation meets machine learning. In *Advances in Neural Information Processing Systems* 34. Curran Associates.
11. Peng, H., Zhou, S., Luo, Y., Xu, N., Duan, S., Ran, R., ... & Ding, C. (2023). Rrnet: Towards relu-reduced neural network for two-party computation based private inference. *arXiv preprint arXiv:2302.02292*.

12. Ziller, A., Trask, A., Lopardo, A., Szymkow, B., Wagner, B., Bluemke, E., ... & Kaissis, G. (2021). PySyft: A library for easy federated learning. In *Federated Learning Systems* (pp. 111–139). Springer.
13. Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., ... & Zhao, Z. (2021). Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14(1–2), 1–210.
14. Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., ... & Seth, K. (2017). Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1175–1191). Association for Computing Machinery.
15. Papernot, N., McDaniel, P., Sinha, A., & Wellman, M. P. (2018). SoK: Security and privacy in machine learning. In *2018 IEEE European Symposium on Security and Privacy* (pp. 399–414). IEEE.
16. Barocas, S., Hardt, M., & Narayanan, A. (2019). Fairness and machine learning: Limitations and opportunities. fairmlbook.org.
17. Mittelstadt, B. (2019). Principles alone cannot guarantee ethical AI. *Nature Machine Intelligence*, 1(11), 501–507.
18. Coyle, D., & Weller, A. (2020). "Explaining" machine learning reveals policy challenges. *Science*, 368(6498), 1433–1434.
19. Veale, M., & Borgesius, F. Z. (2021). Demystifying the draft EU artificial intelligence act. *Computer Law Review International*, 22(4), 97–112.
20. Strubell, E., Ganesh, A., & McCallum, A. (2019). Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 3645–3650). Association for Computational Linguistics.