Efficient Confidential Credit Risk Modeling Using Homomorphic Encryption and Approximation-Aware Deep Learning for Secure Financial Data Analytics

Kacper Daniec
Department of Computer Science, Colorado State University, Fort Collins, CO, USA.
daniec682@colostate.edu

Laureano Curci
School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR, USA.
laureano.work@oregonstate.edu

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

Abstract

Credit risk modeling depends on sensitive financial and behavioral data, which creates a tension between statistical precision and confidentiality when models are trained or evaluated across organizational boundaries. Homomorphic encryption supports computation on encrypted inputs, but secure neural inference for tabular credit data is constrained by multiplicative depth, noise growth, activation approximation error, and the need for calibrated probability-of-default estimates. This article develops an approximation-aware confidential credit risk modeling framework that combines CKKS approximate homomorphic encryption with low-depth polynomial neural networks, calibration-aware training, and depth-conscious architecture selection. The proposed formulation treats encrypted inference as a constrained approximation problem in which activation polynomials, plaintext growth bounds, modulus-chain sizing, and decision thresholds are selected jointly rather than sequentially. Credit-specific requirements, including probability calibration, rank preservation, monotonic risk factors, auditability, and cost-sensitive default decisions, are integrated into the model design and evaluation logic. The analysis clarifies how approximation error propagates through affine layers, why minimax activation fitting is preferable to unconstrained regression of nonlinearities, and how ciphertext packing and parameter folding affect latency and noise consumption. It also identifies failure modes that are specific to encrypted credit analytics, including interval violation, overconfident calibrated scores, noise-budget exhaustion, and mismatches between encrypted-score computation and downstream decision rules. The resulting methodological framework provides a defensible basis for secure financial data analytics without treating cryptographic efficiency and credit-model validity as separate objectives.

Keywords

homomorphic encryption; credit risk modeling; approximation-aware deep learning; CKKS; calibration; secure inference

1. Introduction

Credit risk assessment is an operational decision problem rather than a purely predictive task. Lenders estimate probabilities of default, rank borrowers for review, set exposure limits, and apply cost-sensitive thresholds under regulatory and internal model-risk constraints. The underlying records include income, leverage, repayment histories, transaction behavior, and other attributes whose disclosure can create legal, competitive, and personal harms. Secure computation is therefore relevant not only for cloud inference but also for scenarios in which a data holder seeks a score from a model owner without transmitting raw features, or in which several institutions wish to evaluate a shared model without pooling client-level records.

Fully homomorphic encryption supports computation on encrypted data without decryption [1]. The CKKS scheme is especially relevant for financial analytics because it encodes real-valued vectors and supports approximate addition and multiplication [2]. Encrypted neural inference has been demonstrated for image-oriented and general workloads [3], while encrypted classification and secure two-party learning established that cryptographic inference can be embedded in practical prediction pipelines [4], [5], [6], [7]. Polynomial-evaluation strategies for encrypted deep networks and optimized activation design have reduced the cost of nonlinear computation [8], and broader surveys document the trade-offs among cryptographic assumptions, interaction patterns, and model complexity [9]. Open-source libraries have also made parameter selection, packing, and noise management more reproducible [10]. These advances establish feasibility, but they do not by themselves resolve the model-risk questions that arise when the object being computed is a credit score rather than an arbitrary prediction.

Credit scoring has a long methodological history grounded in statistical discrimination, interpretability, and operational stability [11], [12]. Machine-learning models can improve discrimination and capture nonlinear interactions in consumer and small-business credit data [13], [14], [15]. Their advantages, however, depend on careful preprocessing, calibration, validation, and treatment of class imbalance. A secure inference framework that ignores these requirements can preserve secrecy while degrading the decision quality that motivated encryption in the first place. This is the central gap addressed here: existing encrypted inference methods primarily minimize cryptographic cost or approximate standard neural architectures, whereas credit risk modeling requires an approximation-aware design that treats probability calibration, rank ordering, cost-sensitive thresholds, and noise-aware depth selection as first-class constraints.

The contribution is a coherent methodological framework for confidential credit risk modeling using CKKS and approximation-aware deep learning. The framework specifies a trust boundary, a low-depth polynomial network topology for tabular credit features, minimax activation approximation over bounded intervals, a joint training objective that penalizes risk loss, approximation error, and calibration error, a modulus-chain sizing relation for depth planning, and a decision rule that separates encrypted score computation from threshold application by the authorized key holder. The analysis develops the conditions under which approximation error affects default probabilities and rankings, explains why shallow polynomial networks are preferable to ReLU-based architectures in this setting, and identifies the operational failure modes that must be tested before deployment. The emphasis is on defensible design logic and technical trade-offs rather than on unreported empirical performance.

2. Problem Formulation and Security Model

2.1 Credit risk inference under encryption

Let each borrower or facility be represented by a feature vector x_i in R^m, with a binary default indicator y_i in {0,1} for an observed performance window. A conventional logistic score is a useful reference because it clarifies the quantity that secure inference must preserve. For a linear score z, the probability of default is obtained through the logistic function.

$$p_i = \sigma(w^T x_i + b),\ \sigma(z) = 1/(1 + e\hat{}(-z))$$

Here x_i is the feature vector, w is the coefficient vector, b is the intercept, z is the log-odds score, and σ(·) maps the score to a probability. The logistic model is not proposed as the final architecture; it supplies a baseline decision semantics. A neural credit model replaces the single linear score with layered affine transformations and nonlinear activations, producing a default probability after a final sigmoid or calibrated score transformation. The secure computation problem is to evaluate that composite function while inputs remain encrypted and to return a score only to an authorized party.

Credit features are heterogeneous. Continuous variables such as income, utilization, and debt-service coverage require bounded normalization before encryption. Categorical variables require a fixed encoding determined before encryption so that the data owner and evaluator agree on dimensionality. Missingness indicators must be encoded explicitly because imputation inside ciphertext is possible only if the imputation rule is expressible as a fixed arithmetic operation. These preprocessing choices are not peripheral: they determine the plaintext magnitude bounds used in polynomial approximation and the noise-growth analysis that follows.

2.2 Trust boundary and cryptographic assumptions

The reference architecture involves three roles. The data owner holds private features and may also hold the secret decryption key. The model owner supplies model parameters and the inference program. The evaluator performs encrypted computation, typically in a cloud environment. The evaluator is assumed to be honest-but-curious: it follows the protocol but may inspect ciphertexts, metadata, and intermediate artifacts. The model parameters may be public to the evaluator when model confidentiality is not required; if the model itself is proprietary, encrypted weights can be used at additional depth and ciphertext-expansion cost. The secret key is not available to the evaluator. The final score is decrypted by the key holder, who applies the lending policy threshold locally.

Security rests on the semantic security of the underlying encryption scheme under chosen-plaintext attack and on careful key management. CKKS is approximate: decryption yields a value close to the intended plaintext rather than an exact field element. The analysis therefore separates cryptographic secrecy from numerical fidelity. Encryption can hide x_i from the evaluator while still allowing a score that differs from plaintext evaluation because of polynomial approximation, rescaling, and accumulated noise. A credible design must bound that difference and test whether it is decision-relevant. The framework does not claim that encryption hides all side information, such as the number of inference requests or coarse timing patterns; batching, fixed-shape padding, and deployment-level traffic controls remain necessary when such leakage is material.

2.3 Credit-specific validity requirements

The object returned by secure inference must support four properties. First, it must be a probability-like score whose calibration is adequate for portfolio decisions; discrimination alone is insufficient when thresholds are tied to expected loss. Second, it must preserve rank ordering for borrowers who are close to a cutoff, because small score inversions can change review workloads. Third, monotonic constraints on selected risk factors should be respected where policy or regulation requires them, for example when higher debt-to-income ratios should not reduce predicted risk after conditioning on other variables. Fourth, the model must be auditable. Encryption does not remove the need for documentation of features, training populations, approximation intervals, calibration methods, and threshold logic.

These requirements modify architecture selection. A deep ReLU network may achieve strong discrimination on tabular data, but ReLU is not polynomial and must be replaced or approximated for CKKS evaluation. A high-degree polynomial activation can approximate ReLU closely on a bounded interval, yet each additional multiplication consumes multiplicative depth and increases noise. The design problem is therefore not simply to approximate a standard network; it is to choose a network whose plaintext behavior and encrypted behavior coincide closely enough for credit decisions while remaining within the depth budget.

3. Efficient Homomorphic Credit Risk Architecture

3.1 CKKS arithmetic and error semantics

CKKS encrypts a vector of real or complex values into a ciphertext that supports approximate addition, multiplication by plaintext constants, ciphertext-ciphertext multiplication, rotations, and rescaling. For two ciphertexts ct_1 and ct_2 that encrypt plaintext vectors m_1 and m_2, decryption after homomorphic multiplication returns an approximation to the product rather than an exact product.

$$Dec(ct_1 \odot ct_2) \approx m_1 m_2 + e_ops$$

In this expression, ct_1 and ct_2 are ciphertexts, m_1 and m_2 are the corresponding plaintext vectors, $\odot$ denotes homomorphic multiplication, Dec denotes decryption with the secret key, and e_ops represents accumulated approximation and rescaling error. The relation is schematic but important: every nonlinear operation contributes numerical error in addition to its depth cost. Additions and plaintext multiplications are comparatively cheap, whereas ciphertext-ciphertext multiplication and rotation dominate both latency and noise growth. The architecture should therefore push as much work as possible into affine transformations with public weights and concentrate nonlinearity in a small number of low-degree polynomial evaluations.

3.2 Network topology for tabular credit features

A shallow multilayer perceptron with two to four hidden layers is a reasonable starting topology for structured credit data. Each layer applies an affine map followed by a polynomial activation. Batch normalization is folded into the preceding affine transformation at inference time. Dropout is disabled. Max-pooling and other data-dependent selection operations are avoided because they require comparisons that are expensive or unavailable under CKKS. Categorical embeddings, if used, are represented as fixed one-hot or low-rank linear projections with public parameters. Feature crosses can be introduced through deterministic products before encryption or through additional affine interactions in the first layer.

Ciphertext packing should align with the batching structure of the lending workload. One common layout packs multiple borrowers across slots and evaluates the same model on a batch. A second layout packs features of a single borrower and uses rotations to compute dot products. Batch-oriented packing amortizes rotations and improves throughput when many applications arrive together; feature-oriented packing reduces the complexity of handling variable batch sizes but increases rotation count for wide feature vectors. The choice depends on latency targets and whether the evaluator can pad requests to a fixed batch size. For credit decisions, fixed-shape padding is often acceptable because the cost of extra slots is lower than the risk of variable-shape leakage.

3.3 Approximation-aware activation design

Automatic ReLU replacement methods can reduce private inference cost by selecting polynomial substitutes according to network-level latency constraints [16]. That logic is directly relevant to credit models, where the nonlinearity budget is small and the input range can be bounded through preprocessing. Earlier encrypted network designs also showed that low-degree polynomials can replace standard activations when the approximation interval is controlled [17]. The proposed framework goes further by tying the activation polynomial to credit-specific calibration and decision thresholds rather than treating approximation error as a generic reconstruction problem.

For a bounded scalar score z in [−B,B], a polynomial approximation to the logistic function can be written in Chebyshev form.

$$\sigma(z) \approx p_d(z) = \sum_{k=0}^{d} c_k T_k(\alpha z)$$

Here d is the polynomial degree, T_k is the Chebyshev polynomial of degree k, c_k are expansion coefficients, α is a scaling factor that maps the interval [−B,B] into the domain of the Chebyshev basis, and p_d(z) is the resulting approximation. The interval bound B must be chosen from the training distribution and from architectural constraints: too small a bound clips plausible scores, while too large a bound wastes degree on regions that never occur. A conservative approach is to set B from training and validation quantiles, then add a safety margin for distribution shift.

Coefficient selection should minimize the worst-case error over the interval rather than fit only observed score values.

The expression defines a minimax polynomial approximation problem over the bounded interval. The objective penalizes the largest absolute deviation between the logistic function and the polynomial p_d. Minimax fitting is appropriate because credit thresholds can fall anywhere in the score range, and a small average error with a large local deviation near a cutoff is less acceptable than a slightly larger but uniformly bounded error. In practice, numerical Remez-type methods or Chebyshev interpolation with interval checks can be used to obtain coefficients. The resulting polynomial must be tested for monotonicity on the operational interval if score ordering is to be preserved at the final layer.

The degree d is a design variable. Low-degree polynomials are cheaper under CKKS but may underfit curvature near the extremes. Higher-degree polynomials improve pointwise approximation but increase multiplicative depth and noise. A useful discipline is to assign a per-layer degree budget, use degree three or five for hidden activations, and reserve a slightly higher degree for the final probability transformation if the architecture does not return a log-

odds score. If the final score can be decrypted as a log-odds value and transformed by the key holder, the secure network can terminate in an affine layer, avoiding a costly encrypted sigmoid altogether.

4. Approximation-Aware Training and Secure Inference

4.1 Joint training objective

Training a plaintext model with ReLU and then replacing ReLU after training is suboptimal because the learned weights adapt to the original nonlinearity. Approximation-aware training instead uses the polynomial activation during training, applies interval regularization to keep pre-activations within the validated range, and includes a calibration term so that the final scores remain usable near decision thresholds. The composite objective can be expressed as a weighted sum of risk loss, approximation loss, and calibration loss.

$$L_total = L_risk + \lambda_approx\, L_approx + \lambda_cal\, L_cal$$

Here L_risk is the primary credit risk loss, such as binary cross-entropy or a cost-sensitive surrogate; L_approx measures activation or output deviation from a reference nonlinearity over the validated interval; L_cal measures calibration error or a differentiable surrogate such as a Brier-type penalty; and λ_approx and λ_cal control the trade-off between discrimination, numerical fidelity, and probability calibration. The reference nonlinearity can be the logistic function at the output and a smooth bounded activation in hidden layers. The point is not to force the encrypted model to mimic an arbitrary ReLU network, but to train a model whose polynomial form is the model of record.

Class imbalance requires additional care. Credit default events are often rare, and a model optimized only for unweighted cross-entropy can produce probabilities that are well ranked but poorly calibrated. Cost-sensitive weighting changes the effective threshold and must be separated from the calibration layer. If the key holder applies a cost-sensitive threshold after decryption, the encrypted model should output a calibrated probability or log-odds score, not a pre-thresholded decision. Thresholds are policy parameters, while the encrypted computation is a statistical scoring function.

4.2 Secure evaluation graph

The encrypted inference graph composes affine maps and polynomial activations. For L layers, the plaintext-equivalent evaluation can be written as a composition of layer-wise operations.

In this expression, A_l is the weight matrix at layer l, b_l is the bias vector, p_d^(l) is the polynomial activation used at layer l, and ŷ_i is the encrypted score before decryption. The superscript on p_d identifies the layer, while the subscript d denotes the polynomial degree, which may vary by layer. The affine parameters can be plaintext if the model is not confidential. If model confidentiality is required, A_l and b_l are encrypted and the evaluator performs ciphertext-ciphertext multiplications, which increases depth and latency. For many credit analytics deployments, the model owner is willing to reveal the model architecture and parameters to the evaluator under contract because the sensitive asset is the borrower data; this assumption should be stated explicitly in the deployment documentation.

The evaluation graph should be compiled with a fixed order of operations. Rotations for dot products and slot-wise operations should be hoisted and merged where possible. Polynomial evaluation can use Paterson-Stockmeyer or baby-step giant-step scheduling to reduce the number of sequential multiplications, but the optimizer must respect the modulus

level at each step. A layer with a high-degree polynomial may consume more depth than an affine layer with many additions, so depth planning should be based on the longest multiplicative path rather than on parameter count.

4.3 Noise budget and depth planning

CKKS noise grows with each multiplication and rescaling. The modulus chain must provide enough levels for the required depth while preserving the security parameter at the final level. A conservative planning relation ties the final modulus to the multiplicative depth, plaintext growth, target error, and security margin.

$$\log q_L \geq D \log_2 B + \log_2\left(\varepsilon^{-1}\right) + \kappa$$

Here q_L is the modulus remaining at the final level, D is the maximum multiplicative depth, B is an upper bound on the plaintext magnitude growth per level, ε is the target numerical error, and κ is a padding term for security and implementation margin. The inequality is a sizing heuristic rather than a cryptographic proof; it expresses the need to allocate modulus bits to plaintext growth, error control, and security. If the polynomial degree or the number of nonlinear layers increases, D or B increases, and the modulus chain must be extended or the architecture reduced. Bootstrapping can refresh the noise budget, but it is costly for interactive credit workflows and should be treated as a last resort unless the batch size and latency budget justify it.

The practical implication is that approximation-aware deep learning for credit risk should favor shallow networks, plaintext affine parameters, bounded preprocessing, and polynomial activations that are no more complex than necessary to preserve calibration near thresholds. Techniques that reduce the depth of standard private inference, such as automatic replacement of expensive nonlinearities, are useful only if the replacement preserves the credit model's calibration and monotonicity properties [16]. A depth reduction that shifts scores across a cutoff is not an efficiency gain in this domain.

5. Analytical Assessment and Technical Trade-offs

5.1 Approximation error, probability calibration, and ranking

If a final-layer polynomial satisfies $|\sigma(z) - p_d(z)| \leq \varepsilon$ on the validated score interval, then the encrypted default probability differs from the plaintext reference by at most ε at the output, assuming the preceding layers are evaluated without additional error. The full encrypted network introduces additional error from hidden-layer approximation, rescaling, and noise. A simple triangle inequality suggests that the total output error is bounded by the sum of layer-wise approximation errors propagated through Lipschitz constants of the affine maps and activation polynomials. This bound is loose, but it identifies the correct design levers: keep the spectral norms of affine maps moderate, keep activation derivatives bounded on the interval, and prevent pre-activations from leaving the approximation domain.

Calibration is more sensitive than discrimination to small systematic distortions. A monotone distortion of scores can preserve ranking while shifting probabilities away from observed default rates. Because lending thresholds are defined on probabilities or expected loss, a well-ranked but miscalibrated encrypted score can still change approval volumes. Post-hoc calibration methods developed for supervised learning provide a useful reference for plaintext calibration [21]. In the secure framework, post-hoc calibration can be performed after decryption by the key holder if the calibrator is not confidential. If calibration must remain encrypted, it should be represented as a low-degree monotone polynomial or an affine

transformation in log-odds space, and its approximation error should be included in the validation plan.

Rank preservation requires additional care. A polynomial approximation to a monotone sigmoid is not automatically monotone over the entire real line, but it can be monotone on a finite interval if its derivative is positive there. For single-score models, monotonicity on the validated interval preserves ordering after the final transformation. For multilayer networks, rank preservation depends on the composition of affine maps and activations; it cannot be assumed from the final layer alone. Credit models that require monotonic constraints on selected features should enforce them during training through constrained optimization or post-training projection, then verify the constraints on encrypted outputs through a sample-based equivalence audit.

5.2 Latency, packing, and amortization

The dominant costs in encrypted inference are ciphertext-ciphertext multiplications, rotations, and rescaling. Affine transformations with plaintext weights require plaintext-ciphertext multiplications and additions, which are cheaper but not free. Polynomial activations require several ciphertext-ciphertext multiplications or ciphertext-plaintext multiplications depending on the evaluation schedule. Packing multiple borrowers into slots amortizes rotations and improves throughput, but it also increases the required plaintext modulus and may complicate per-borrower thresholding. A batch of n borrowers can be processed together, with scores decrypted as a vector and thresholds applied elementwise by the key holder.

Latency and depth are related but distinct. A network with low depth may still have high latency if it uses many rotations or a large batch. A network with high depth may be unacceptable because it exhausts the modulus chain even if its arithmetic operation count is modest. The architecture should therefore be optimized under two constraints: a maximum multiplicative depth consistent with the modulus chain and a maximum rotation count consistent with the latency target. Approximation-aware training can reduce both by encouraging weights that keep activations within a small interval, allowing lower-degree polynomials, and by avoiding unnecessary feature interactions that require many rotations.

5.3 Failure modes and boundary conditions

Four failure modes are especially relevant. Interval violation occurs when encrypted pre-activations leave the range used for polynomial fitting; the polynomial may then diverge rapidly, producing scores that are numerically plausible but decision-invalid. The mitigation is interval regularization during training and a runtime check based on plaintext validation statistics, not on inspecting encrypted values. Noise-budget exhaustion occurs when the modulus chain is too short for the chosen depth; decryption then returns a score with large error or fails outright. The mitigation is conservative depth planning and a validation phase that measures ciphertext level consumption for representative workloads without treating those measurements as evidence of model performance.

Calibration shift is a third failure mode. Even if the encrypted score approximates the plaintext score closely, the deployment population may differ from the training population, and the polynomial approximation may be less accurate in the shifted region. Monitoring should therefore compare decrypted score distributions and approval rates against plaintext reference evaluations on periodic samples approved for auditing. The fourth failure mode is a mismatch between encrypted scoring and downstream decision logic. If the encrypted model

outputs a probability but the lending policy expects a log-odds score, or if the threshold is applied before calibration, the secure computation can be cryptographically correct and operationally wrong. Decision semantics must be fixed before encryption and documented as part of the model specification.

6. Implementation Considerations and Validation Logic

6.1 Deployment workflow

A defensible implementation separates model development, encryption, evaluation, and decision application. During development, the model owner trains the polynomial network in plaintext, fits activation polynomials over documented intervals, calibrates the output, and freezes the feature preprocessing. During encryption, the data owner applies the fixed preprocessing, encodes features into CKKS plaintext slots, encrypts them with a public key, and transmits ciphertexts with fixed shapes. The evaluator executes the compiled inference graph and returns encrypted scores. The key holder decrypts, applies calibration if it was not part of the encrypted graph, and then applies the cost-sensitive threshold.

The cryptographic parameters should be selected for the maximum batch size, multiplicative depth, and security level approved for the deployment. Libraries for encrypted tensor operations can support the required packing and polynomial evaluation, but the parameter configuration must be reviewed independently because library defaults are not a substitute for deployment-specific analysis [19]. Practical homomorphic encryption involves trade-offs among ciphertext size, computation time, and security that have long been recognized in applied cryptography [20]. These trade-offs should be documented alongside the model card, not hidden in infrastructure configuration.

6.2 Equivalence and validation protocol

Validation should compare plaintext and encrypted inference on the same held-out records. The relevant metrics are not limited to aggregate discrimination. They include mean absolute score difference, maximum score difference within the operational interval, rank correlation, approval-rate disagreement at policy thresholds, and calibration discrepancy. A model can pass an aggregate discrimination check while failing a cutoff-sensitive disagreement test. The validation protocol should therefore stratify by score decile and by known vulnerable or underrepresented segments, subject to privacy and legal constraints on subgroup reporting.

The protocol should also test approximation robustness under stress. Synthetic perturbations within the documented feature ranges can reveal whether polynomial activations remain within a stable region. Out-of-range perturbations should trigger rejection or fallback rather than silent extrapolation. Because CKKS is approximate, the test plan should report numerical tolerances explicitly and should distinguish cryptographic decryption failure from model-level score deviation. The same plan should verify that encrypted inference does not alter the monotonicity constraints imposed on selected features. These checks are analytical and procedural; they do not require claiming empirical performance beyond what the validation records establish.

6.3 Model risk, auditability, and secure analytics

Homomorphic encryption changes the data-access boundary but not the need for model-risk governance. Documentation should record the feature set, training population, hyperparameters, polynomial degrees, approximation intervals, calibration method, threshold

costs, and CKKS parameter choices. The encrypted inference graph should be versioned so that a score can be reproduced from a specified model, preprocessing version, and parameter set. A survey of homomorphic encryption schemes and implementations provides useful context for selecting primitives and understanding their operational assumptions [22]. In credit analytics, the most defensible use of encryption is often narrow and well specified: a lender or bureau evaluates a frozen model on encrypted features and returns a score to the authorized decision system, rather than attempting to move the entire model-development lifecycle into an encrypted domain.

Secure financial analytics also requires a clear distinction between confidentiality and explainability. Encryption protects inputs and intermediate values from the evaluator, but the model owner and key holder may still be able to produce reason codes after decryption. If reason codes must be generated without revealing the full feature vector, local explanations can be computed from encrypted contributions or from a separate secure protocol. The choice should be driven by the institution's obligations and by the sensitivity of the explanation itself. A secure score without a documented explanation pathway is unlikely to satisfy model-risk review in a lending institution.

7. Conclusion

Confidential credit risk modeling with homomorphic encryption is not merely a cryptographic engineering problem. The score produced under encryption must remain calibrated, rank-preserving where required, monotone on constrained risk factors, and consistent with the lending policy that consumes it. This article has developed an approximation-aware framework that treats those properties as constraints on architecture and training rather than as post-hoc checks. CKKS provides the arithmetic substrate for real-valued encrypted inference, while shallow polynomial networks, interval-bounded activation fitting, plaintext affine parameters, and depth-conscious packing reduce the cost of nonlinear computation. Minimax activation approximation is preferable to unconstrained fitting because credit decisions are threshold-sensitive and worst-case deviations near cutoffs matter more than average reconstruction error.

The analysis also clarifies the limits of the approach. Approximation error propagates through layers, noise growth constrains depth, and calibration can shift even when aggregate ranking is preserved. These are substantive boundary conditions, not implementation details. A credible deployment therefore requires plaintext-to-encrypted equivalence testing, cutoff-sensitive disagreement analysis, calibration monitoring, parameter documentation, and a clear split between encrypted scoring and local threshold application. Automatic activation replacement and other private-inference optimizations can improve efficiency, but only when they preserve the credit model's probabilistic and decision semantics. The resulting framework offers a technically coherent route toward secure financial data analytics in which confidentiality and model validity are designed together.

References

1. Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. In Proceedings of the 41st Annual ACM Symposium on Theory of Computing (pp. 169–178). ACM.

2. Cheon, J. H., Kim, A., Kim, M., & Song, Y. (2017). Homomorphic encryption for arithmetic of approximate numbers. In Advances in Cryptology – ASIACRYPT 2017 (pp. 409–437). Springer.

3. 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.

4. Bost, R., Popa, R. A., Tu, S., & Goldwasser, S. (2015). Machine learning classification over encrypted data. In Proceedings of the 2015 Network and Distributed System Security Symposium. Internet Society.

5. Mohassel, P., & Zhang, Y. (2017). SecureML: A system for scalable privacy-preserving machine learning. In 2017 IEEE Symposium on Security and Privacy (pp. 19–38). IEEE.

6. Juvekar, C., Vaikuntanathan, V., & Chandrakasan, A. (2018). GAZELLE: A low latency framework for secure neural network inference. In 27th USENIX Security Symposium (pp. 1651–1669). USENIX Association.

7. Mishra, P., Lehmkuhl, R., Srinivasan, A., Zheng, W., & Popa, R. A. (2020). Delphi: A cryptographic inference service for neural networks. In 29th USENIX Security Symposium (pp. 2505–2522). USENIX Association.

8. Lou, Q., & Jiang, L. (2019). SHE: A fast and accurate deep neural network for encrypted data. In Advances in Neural Information Processing Systems 32. Curran Associates.

9. Boulemtafes, A., Derhab, A., & Challal, Y. (2020). A review of privacy-preserving techniques for deep learning. Neurocomputing, 384, 21–45.

10. Al Badawi, A., Bates, J., Bergamaschi, F., Cousins, D. B., Ead, S., Eads, C., ... & Polyakov, Y. (2022). OpenFHE: Open-source fully homomorphic encryption library. In Proceedings of the 10th Workshop on Encrypted Computing and Applied Homomorphic Cryptography. ACM.

11. Hand, D. J., & Henley, W. E. (1997). Statistical classification methods in consumer credit scoring: A review. Journal of the Royal Statistical Society: Series A (Statistics in Society), 160(3), 523–541.

12. Thomas, L. C., Edelman, D. B., & Crook, J. N. (2002). Credit scoring and its applications. SIAM.

13. Lessmann, S., Baesens, B., Seow, H.-V., & Thomas, L. C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. European Journal of Operational Research, 247(1), 124–136.

14. Khandani, A. E., Kim, A. J., & Lo, A. W. (2010). Consumer credit-risk models via machine-learning algorithms. Journal of Banking & Finance, 34(11), 2767–2787.

15. Barboza, F., Kimura, H., & Altman, E. (2017). Machine learning models and bankruptcy prediction. Expert Systems with Applications, 83, 405–417.

16. Peng, H., Huang, S., Zhou, T., Luo, Y., Wang, C., Wang, Z., ... & Ding, C. (2023, October). Autorep: Automatic relu replacement for fast private network inference. In 2023 IEEE/CVF International Conference on Computer Vision (ICCV) (pp. 5155-5165). IEEE.

17. Hesamifard, E., Takabi, H., & Ghasemi, M. (2017). CryptoDL: Deep neural networks over encrypted data. arXiv preprint arXiv:1711.05189.

18. Kim, M., Song, Y., Wang, S., Xia, Y., & Jiang, X. (2018). Secure logistic regression based on homomorphic encryption: Design and evaluation. JMIR Medical Informatics, 6(2), e19.

19. Benaissa, A., Retiat, B., Cebere, B., & Belfodil, A. E. (2021). TenSEAL: A library for encrypted tensor operations using homomorphic encryption. arXiv preprint arXiv:2104.03152.

20. Naehrig, M., Lauter, K., & Vaikuntanathan, V. (2011). Can homomorphic encryption be practical? In Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop (pp. 113–124). ACM.

21. Niculescu-Mizil, A., & Caruana, R. (2005). Predicting good probabilities with supervised learning. In Proceedings of the 22nd International Conference on Machine Learning (pp. 625–632). ACM.

22. Acar, A., Aksu, H., Uluagac, A. S., & Conti, M. (2018). A survey on homomorphic encryption schemes: Theory and implementation. ACM Computing Surveys, 51(4), Article 79.