

VeriRegime: Deep Learning Model Optimization for Zero-Knowledge Verifiable Trading Signal Generation

Lin Boyi 123090327
The Chinese University of Hong Kong, Shenzhen
DDA4220: Deep Learning

November 8, 2025 (Revised)

Abstract

The rise of autonomous AI trading agents in decentralized finance (DeFi) introduces a critical trust problem: users cannot verify whether trading signals genuinely originate from AI models or are manually manipulated. Zero-knowledge machine learning (zkML) enables cryptographic verification of model inference, but existing neural architectures incur prohibitive proof generation costs. This project investigates **deep learning optimization techniques** to design zkML-compatible models for trading signal classification. We focus on three core contributions: (1) **zkML-aware knowledge distillation** that transfers LSTM teacher knowledge to compact MLP students optimized for proof generation; (2) **polynomial activation networks** that replace ReLU with low-degree approximations to reduce arithmetic constraints; (3) **sensitivity-guided adaptive quantization** that allocates per-layer bit-widths based on Hessian analysis. Through systematic ablation studies, we map the Pareto frontier between model accuracy and proof efficiency, establishing design principles for verifiable financial AI systems.

1 Introduction

1.1 Motivation: The Trust Problem in AI Trading

Autonomous AI trading agents are proliferating in cryptocurrency markets, with platforms like ai16z and Virtual Protocol managing millions in assets. However, a fundamental trust gap exists: **users cannot verify that trading decisions genuinely come from AI models rather than human intervention**. This opacity undermines the core value proposition of algorithmic trading—systematic, emotionless decision-making.

Zero-knowledge proofs (ZKPs), particularly succinct non-interactive arguments of knowledge (zkSNARKs), offer a cryptographic solution. By generating a proof alongside each inference, a model can mathematically demonstrate: “*This output was produced by running model f with parameters θ on input x* ”—without revealing θ or x .

1.2 The Challenge: Neural Networks are Proof-Expensive

Standard deep learning architectures (LSTMs, Transformers) are **incompatible with efficient proof generation**:

- **Non-linear activations** (ReLU, sigmoid) require expensive selection gates in arithmetic circuits

- **High-precision weights** (FP32) explode constraint counts when compiled to finite-field arithmetic
- **Large parameter counts** directly translate to proof size and generation time

Existing zkML research focuses on *system-level* optimizations (proof systems, compilers), but largely ignores *model-level* design. This project addresses the question: **How should we re-design neural architectures and training procedures to make financial time-series models zkML-friendly while preserving predictive accuracy?**

1.3 Research Scope and Contributions

We frame this as a **deep learning optimization problem** with zkML as a constraint, rather than a blockchain project with ML components. Our contributions are:

1. **zkML-Aware Distillation Framework:** A teacher-student pipeline where LSTM teachers capture temporal dependencies, and MLP students learn to replicate their behavior with zkML-compatible operations
2. **Polynomial Activation Networks:** Systematic study of low-degree polynomial approximations to ReLU, trained with custom regularization to balance approximation error and proof efficiency
3. **Sensitivity-Guided Quantization:** Adaptive per-layer bit-width allocation using Hessian-based sensitivity analysis, avoiding the accuracy degradation of uniform quantization
4. **Empirical Analysis:** Comprehensive ablation studies mapping the accuracy-proof cost trade-off, validated through end-to-end zkML deployment

2 Literature Review

2.1 Knowledge Distillation for Model Compression

Knowledge distillation [1] transfers dark knowledge from large teacher models to compact students. Recent work explores **task-specific distillation**: DistilBERT for NLP [2], MobileNet for vision [3]. However, **distillation for zkML compatibility remains unexplored**—our work extends this paradigm by optimizing student architectures for proof generation, not just parameter count.

2.2 Quantization-Aware Training

Quantization reduces numerical precision to lower computation and memory costs. Post-training quantization (PTQ) is simple but lossy; quantization-aware training (QAT) [4] simulates quantization during training to recover accuracy. **Mixed-precision quantization** [5] adaptively assigns bit-widths, but existing methods optimize for hardware efficiency (e.g., INT8 on GPUs), not zkML constraint counts. We adapt sensitivity analysis [6] to the finite-field arithmetic context.

2.3 Activation Function Design

ReLU’s non-differentiability at zero and unbounded output create challenges for both optimization and circuit compilation. Polynomial activations (e.g., x^2 , Swish approximations [7]) offer differentiability and bounded constraints. **Our contribution:** systematically evaluate polynomial families

for financial time-series under zkML constraints, proposing training-time regularization to minimize approximation error.

2.4 Zero-Knowledge Machine Learning (zkML)

Early frameworks like zkCNN [8] demonstrated feasibility. Recent systems (EZKL, Modulus Labs) provide production tooling. However, **all existing work treats neural architectures as fixed inputs**—they optimize proof systems, not models. We invert this: treat zkML compilers as fixed infrastructure and optimize models for them.

3 Research Questions

1. **Distillation Design:** How can we design distillation objectives that preserve temporal reasoning from LSTM teachers while constraining students to zkML-friendly operations (matrix multiplications, polynomial activations)?
2. **Activation Trade-offs:** What is the Pareto frontier between polynomial approximation error and arithmetic constraint reduction for common activations (ReLU, Sigmoid, Tanh)? Which polynomial families (power series, Chebyshev, rational functions) are optimal?
3. **Adaptive Quantization:** Can Hessian-based sensitivity analysis effectively guide per-layer bit-width allocation in zkML models? How does this compare to uniform quantization and neural architecture search?
4. **Generalization:** Do zkML-optimized models maintain accuracy across different market regimes (bull, bear, high/low volatility)? What inductive biases are preserved/lost in distillation?
5. **End-to-End Feasibility:** What are the practical limits (proof time, proof size, verification cost) of deploying optimized models via EZKL/Halo2 to Ethereum testnets?

4 Methodology

4.1 Problem Formulation

Task: Given a sequence of market observations $\mathbf{x}_{t-w:t} = \{x_{t-w}, \dots, x_t\}$ where $x_i \in \mathbb{R}^d$ contains price/volume features, predict a trading signal $y \in \{\text{BUY}, \text{HOLD}, \text{SELL}\}$.

Features ($d = 8$):

- EMA(5), EMA(10), EMA(20) — Exponential moving averages
- RSI, MACD — Technical indicators
- Volume MA(5), Volume MA(10)
- Funding rate (for perpetual futures)

Labels: Based on forward 1-hour returns:

$$y_t = \begin{cases} \text{BUY} & \text{if } r_{t+1h} > +2\% \\ \text{SELL} & \text{if } r_{t+1h} < -2\% \\ \text{HOLD} & \text{otherwise} \end{cases}$$

Dataset: Bitcoin (BTC/USDT) 1-minute candles from Binance API, 2023-2024 (500K samples). Train/Val/Test split: 70/15/15.

4.2 Baseline: LSTM Teacher Model

Architecture:

- Input: $\mathbf{x}_{t-60:t}$ (60-minute sliding window)
- 2-layer LSTM, hidden size 128
- Fully connected layer: $128 \rightarrow 3$ (softmax)
- Loss: Cross-entropy + label smoothing ($\epsilon = 0.1$)

Training: Adam optimizer, learning rate 10^{-3} with cosine decay, batch size 256, early stopping on validation F1-score.

Purpose: Establish accuracy upper bound. LSTM’s recurrence makes it zkML-incompatible but effective at capturing temporal dependencies.

4.3 zkML-Compatible MLP Student

Architecture:

- Input: \mathbf{x}_t (flattened: $60 \times 8 = 480$ features)
- 3 hidden layers: $480 \rightarrow 128 \rightarrow 64 \rightarrow 32 \rightarrow 3$
- Activation: **Polynomial** (degree d , varies by experiment)
- No batch normalization (incompatible with zkML)
- Output: Argmax (no softmax—avoid division in circuit)

Key Constraint: All operations must compile to efficient arithmetic circuits.

4.4 zkML-Aware Knowledge Distillation

Objective: Train MLP student to mimic LSTM teacher’s behavior.

Loss Function:

$$\mathcal{L} = \underbrace{\alpha \cdot \mathcal{L}_{\text{CE}}(y_{\text{true}}, \hat{y}_{\text{student}})}_{\text{Hard label loss}} + \underbrace{\beta \cdot \mathcal{L}_{\text{KD}}(z_{\text{teacher}}, z_{\text{student}})}_{\text{Soft label loss}} + \underbrace{\gamma \cdot \mathcal{L}_{\text{reg}}}_{\text{zkML regularization}}$$

Components:

- \mathcal{L}_{CE} : Standard cross-entropy with true labels
- \mathcal{L}_{KD} : KL-divergence between teacher/student logits at temperature $T = 3$
- \mathcal{L}_{reg} : Novel **constraint-aware regularization**:

$$\mathcal{L}_{\text{reg}} = \lambda_1 \|\mathbf{W}\|_1 + \lambda_2 \sum_l \text{ReLU}(|w_l| - \tau_{\text{quant}})$$

where the first term encourages sparsity (fewer non-zero weights = smaller proof), and the second penalizes weights exceeding quantization thresholds.

Hyperparameters: $\alpha = 0.5$, $\beta = 0.5$, $\gamma = 0.1$, tuned via grid search.

4.5 Polynomial Activation Design

Motivation: ReLU requires selection gates in circuits:

$$\text{ReLU}(x) = \max(0, x) \implies \text{selector} \cdot x$$

Each selector adds constraints. Polynomials are native to arithmetic circuits.

Candidates:

1. **Quadratic:** $\sigma(x) = x^2$ (simplest, always positive)
2. **Cubic:** $\sigma(x) = x + 0.1x^3$ (preserves sign, bounded derivative)
3. **ReLU Approximation:** $\sigma(x) = \frac{x + \sqrt{x^2 + \epsilon}}{2}$ Taylor-expanded to degree 3
4. **Swish Approximation:** Polynomial fit to $\sigma(x) = x \cdot \text{sigmoid}(x)$

Training Protocol:

- Initialize with pre-trained ReLU model
- Gradually replace activations layer-by-layer (curriculum learning)
- Fine-tune with reduced learning rate (10^{-4})

Evaluation: Measure (1) accuracy drop vs. ReLU, (2) arithmetic constraint count via EZKL compilation.

4.6 Adaptive Quantization via Sensitivity Analysis

Problem: Uniform quantization (e.g., all layers to 8-bit) degrades accuracy. Different layers have different sensitivity to precision reduction.

Method: Hessian-Aware Quantization (HAQ) [6]

Step 1 — Sensitivity Measurement: For each layer l , compute trace of Hessian w.r.t. layer weights:

$$S_l = \text{Tr}(\nabla_{\mathbf{W}_l}^2 \mathcal{L})$$

High S_l indicates high sensitivity—requires more bits.

Step 2 — Bit-width Allocation: Given constraint budget C (total bits across all layers), solve:

$$\min_{\{b_l\}} \sum_l S_l \cdot Q(b_l) \quad \text{s.t.} \quad \sum_l b_l \leq C$$

where $Q(b_l)$ is quantization error for b_l bits (estimated via calibration set).

Step 3 — Quantization-Aware Fine-tuning:

- Simulate quantization with allocated bit-widths during forward pass
- Use straight-through estimators for gradients
- Train for 10 epochs with learning rate 10^{-5}

Baselines: Compare against uniform 4-bit, 8-bit, 16-bit quantization.

4.7 zkML Compilation and Verification

Toolchain: EZKL (v10+) with Halo2 proof system

Pipeline:

1. Export trained PyTorch model to ONNX format
2. Compile ONNX \rightarrow arithmetic circuit via `ezkl compile`
3. Generate proving/verifying keys
4. For each inference:
 - Compute output + generate proof (prover time measured)
 - Verify proof (verifier time + proof size measured)
5. Deploy Solidity verifier contract to Ethereum Sepolia testnet
6. Measure on-chain verification gas cost

Metrics:

- **Accuracy:** F1-score, macro-averaged across 3 classes
- **Proof Size:** Bytes (smaller = more efficient on-chain storage)
- **Prover Time:** Seconds on M1 MacBook Pro (practical deployment limit)
- **Verifier Time:** Milliseconds (off-chain throughput)
- **Gas Cost:** Wei (on-chain verification economic feasibility)

5 Experimental Design

5.1 Experiment 1: Teacher-Student Distillation Ablation

Goal: Validate that distillation preserves accuracy compared to training MLP from scratch.

Variants:

- MLP-Scratch: Train MLP directly on hard labels
- MLP-Distill-Hard: $\alpha = 1, \beta = 0$ (only hard labels from teacher)
- MLP-Distill-Soft: $\alpha = 0, \beta = 1$ (only soft labels from teacher)
- MLP-Distill-Combined: $\alpha = 0.5, \beta = 0.5$ (proposed)
- MLP-Distill-zkReg: Combined + $\gamma = 0.1$ (with zkML regularization)

Metrics: Accuracy, parameter count, sparsity percentage.

5.2 Experiment 2: Polynomial Activation Comparison

Goal: Identify best polynomial family for accuracy-efficiency trade-off.

Setup:

- Fix architecture (3-layer MLP)
- Fix quantization (8-bit uniform)
- Vary activation: ReLU (baseline), Quadratic, Cubic, ReLU-approx, Swish-approx

Analysis:

- Plot Pareto frontier: Accuracy vs. Constraint Count
- Visualize activation functions and their derivatives
- Analyze failure modes (dead neurons, gradient vanishing)

5.3 Experiment 3: Quantization Strategy Comparison

Goal: Demonstrate adaptive quantization outperforms uniform.

Variants:

- FP32 (baseline, no quantization)
- Uniform-16bit
- Uniform-8bit
- Uniform-4bit
- Adaptive (Hessian-guided, same total bit budget as Uniform-8bit)

Metrics:

- Accuracy degradation vs. FP32
- Per-layer bit-width allocation (visualize as heatmap)
- Proof size reduction

5.4 Experiment 4: End-to-End zkML Deployment

Goal: Validate practical feasibility on Ethereum testnet.

Setup:

- Select best model from Experiments 1-3
- Generate 100 proofs on test set
- Deploy verifier to Sepolia testnet
- Submit 10 proofs on-chain

Measurements:

- Average prover time (acceptable threshold: < 30 seconds)
- Proof size distribution
- On-chain verification gas (acceptable threshold: < 500K gas)

6 Expected Results and Analysis

6.1 Hypothesis 1: Distillation Recovers $> 90\%$ of Teacher Accuracy

Rationale: Prior work shows distillation preserves $\sim 95\%$ accuracy in computer vision. Financial time-series may be harder due to noise, but LSTM’s inductive bias should transfer via soft labels.

Success Metric: F1-score gap $< 5\%$ between LSTM teacher and MLP-Distill-zkReg.

6.2 Hypothesis 2: Cubic Polynomials Offer Best Trade-off

Rationale: Quadratic lacks negative outputs (problematic for classification). Degree > 3 increases constraints without commensurate accuracy gain.

Success Metric: Cubic activation achieves $< 2\%$ accuracy drop vs. ReLU while reducing constraints by $> 30\%$.

6.3 Hypothesis 3: Adaptive Quantization Beats Uniform

Rationale: Early layers extract low-level features (tolerate low precision), while later layers refine predictions (need higher precision).

Success Metric: Adaptive-8bit matches Uniform-16bit accuracy with 50% fewer total bits.

6.4 Pareto Frontier Visualization

We will generate plots mapping:

- X-axis: Proof generation time (seconds)
- Y-axis: Model F1-score
- Color: Proof size (bytes)
- Markers: Different configurations (activation \times quantization)

This identifies the “efficient frontier” of zkML models suitable for production deployment.

7 Timeline

Week	Task	Deliverable
7 (Nov 8-14)	Data collection, EDA, LSTM training	Teacher model checkpoint
8 (Nov 15-21)	MLP student training, Exp 1 (distillation)	Distillation ablation results
9 (Nov 22-28)	Exp 2 (activations), Exp 3 (quantization)	Polynomial + quantization analysis
10 (Dec 1-7)	EZKL compilation, proof benchmarking	Off-chain proof metrics
11 (Dec 8-14)	Exp 4 (on-chain deployment)	Testnet deployment demo
12 (Dec 15-21)	Results analysis, visualization	Pareto frontier plots
13 (Dec 22-28)	Report writing	Draft final report
14 (Dec 29-Jan 4)	Presentation preparation	Final slides + demo video

Table 1: Project Timeline (Fall 2025)

Risk Mitigation:

- If EZKL compilation fails: Focus on off-chain proof benchmarking only (still valid DL research)
- If distillation underperforms: Fall back to training MLP directly with zkML regularization
- If time runs short: Drop Experiment 4 (on-chain deployment) and focus on core DL experiments

8 Significance and Impact

8.1 Deep Learning Contributions

This project advances neural architecture design for constrained computation:

- **Distillation for Verification:** Extends knowledge distillation to the novel objective of proof efficiency
- **Activation Theory:** Systematic empirical analysis of polynomial activations in financial domain
- **Adaptive Quantization:** Demonstrates Hessian-based methods generalize beyond hardware optimization to cryptographic constraints

8.2 Practical Impact

Trustworthy AI Trading: Enables auditable AI agents where users can verify every trading decision came from the declared model, not human manipulation.

DeFi Governance: DAO treasuries could use verifiable AI signals for automated rebalancing with cryptographic audit trails.

Regulatory Compliance: Financial institutions exploring algorithmic trading can demonstrate model provenance to regulators.

8.3 Future Work

- Extend to multi-asset portfolio optimization with verifiable rebalancing
- Explore recursive SNARKs for verifying model *training*, not just inference
- Investigate federated learning with zkML to enable collaborative model training without data sharing

9 Conclusion

VeriRegime bridges zero-knowledge cryptography and deep learning optimization to address a real trust gap in autonomous AI trading. By treating zkML compatibility as a first-class constraint in neural architecture design, we establish a principled framework for building verifiable financial AI. Our focus on **distillation strategies, polynomial activations, and adaptive quantization** represents novel deep learning research with immediate practical applications in decentralized finance.

References

- [1] Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.
- [2] Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.
- [3] Howard, A. G., Zhu, M., Chen, B., et al. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.
- [4] Jacob, B., Kligys, S., Chen, B., et al. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. *CVPR 2018*.
- [5] Wang, K., Liu, Z., Lin, Y., Lin, J., & Han, S. (2019). HAQ: Hardware-aware automated quantization with mixed precision. *CVPR 2019*.
- [6] Dong, Z., Yao, Z., Gholami, A., Mahoney, M. W., & Keutzer, K. (2019). HAWQ: Hessian aware quantization of neural networks with mixed-precision. *ICCV 2019*.
- [7] Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941*.
- [8] Lai, R. W., Tai, R. K., Wong, H. W., & Chow, S. S. (2021). zkCNN: Zero knowledge proofs for convolutional neural network predictions and accuracy. *ACM CCS 2021*.
- [9] EZKL Team. (2024). EZKL: Easy zero-knowledge inference. <https://ezkl.xyz>.