VeriRegime: Distilling High-Performance CNNs to zkML-Optimized MLPs for Trading Signal Generation

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

November 9, 2025 (Final)

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 high-performance architectures like Convolutional Neural Networks (CNNs) incur prohibitive proof generation costs due to convolutional operations and non-linear activations. This project investigates **knowledge distillation and model optimization techniques** to transform zkML-unfriendly CNNs into verifiable Multi-Layer Perceptrons (MLPs). We focus on three core contributions: (1) **CNN-to-MLP distillation framework** that transfers local temporal pattern extraction capabilities while maintaining zkML compatibility; (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 experiments, we quantify the accuracy-proof efficiency trade-off, demonstrating that a 10-15% accuracy sacrifice yields 10-50× proof generation speedup, establishing practical design principles for verifiable financial AI systems.

1 Introduction

1.1 Motivation: The Verifiability Gap 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 based on data.

Consider a typical scenario: A decentralized fund claims "our AI model predicted this BTC rally." Users have no way to verify:

- Whether the output truly came from the declared model (vs. human manipulation)
- Whether input data was tampered with
- Whether model parameters match the claimed version

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 proprietary θ or x.

1.2 The Challenge: High-Performance Models are Proof-Expensive

While zkML frameworks like EZKL [**?**] and zkCNN [**?**] demonstrate technical feasibility, deploying **practical deep learning models** remains prohibitively expensive. Convolutional Neural Networks (CNNs), widely used for time-series tasks, face severe zkML barriers:

- **Convolutional operations**: Each kernel requires $k \times d$ multiplications compiled to finite-field constraints
- **ReLU activations**: $\text{ReLU}(x) = \max(0, x)$ requires selection gates: selector $\cdot\, x$, adding $O(n)$ constraints per layer
- **Pooling layers**: MaxPool needs comparison circuits, AvgPool needs division
- **Batch normalization**: Division operations explode constraint counts

Example: A modest 1D CNN with 2 convolutional layers (128 filters each), ReLU, and global pooling generates $\sim$10M constraints, requiring $\sim$300 seconds proof time on M1 MacBook Pro—**economically infeasible** for real-time trading.

1.3 Research Objective

This project addresses the question: **Can we transform high-performance but zkML-unfriendly CNNs into verifiable models through knowledge distillation and optimization, while maintaining acceptable accuracy?**

We hypothesize that by:

1. Distilling CNN teacher knowledge to MLP students
2. Replacing ReLU with polynomial activations
3. Applying adaptive quantization

we can achieve **85%+ accuracy preservation with 10-50$\times$ proof efficiency gains**.

1.4 Contributions

We frame this as a **deep learning optimization problem** with zkML as a constraint, contributing:

1. **CNN-to-MLP Distillation Framework**: A teacher-student pipeline where CNN teachers capture local temporal patterns through convolution, and MLP students learn to replicate their decision boundaries with zkML-compatible operations. We provide theoretical justification for why this transfer succeeds.
2. **Polynomial Activation Optimization**: Systematic study of low-degree polynomial families (Quadratic, Cubic, rational approximations) for financial time-series, proposing training-time regularization to balance approximation error and constraint reduction.

3. **Adaptive Quantization for zkML**: Extension of Hessian-based mixed-precision quantization from hardware optimization to zkML finite-field arithmetic, demonstrating superior accuracy-efficiency trade-offs vs. uniform quantization.

4. **End-to-End Evaluation**: Comprehensive benchmarking of CNN vs. MLP on proof generation metrics (time, constraint count, proof size, gas cost), producing Pareto frontier analysis for practical deployment guidance.

Novelty: While prior zkML work optimizes proof systems, we optimize *models* for proof systems. While prior distillation work targets hardware efficiency, we target *cryptographic verifiability*.

2 Literature Review

2.1 Knowledge Distillation for Model Compression

Knowledge distillation [?] transfers "dark knowledge" from complex teacher models to compact students by matching soft label distributions. Recent work explores task-specific distillation: DistilBERT for NLP [?], MobileNet for vision [?].

CNN-to-MLP Distillation: Urban et al. [?] show that MLPs can approximate CNNs on fixed-length inputs by learning implicit positional encodings through weight matrices, achieving 85-92% accuracy retention in image classification.

Gap: **No prior work explores distillation for zkML compatibility**, where the objective is not just parameter reduction but *proof efficiency*. Our contribution extends distillation objectives with constraint-aware regularization.

2.2 Convolutional Networks for Time-Series

1D CNNs effectively model time-series by learning local temporal patterns [?]. Temporal Convolutional Networks (TCNs) achieve state-of-the-art results on financial forecasting [?]. However, their zkML deployment remains unstudied.

zkML Barrier: Convolution's computational intensity translates directly to proof complexity. A single Conv1D layer with 64 filters and kernel size 5 generates $\sim$2M constraints—orders of magnitude more than fully-connected layers.

2.3 Quantization-Aware Training

Quantization reduces numerical precision to lower computation costs. Post-training quantization (PTQ) is simple but lossy; quantization-aware training (QAT) [?] simulates quantization during training to recover accuracy.

Mixed-Precision Quantization: HAQ [?] and HAWQ [?] use Hessian trace analysis to allocate per-layer bit-widths, optimizing for hardware (e.g., INT8 on GPUs).

Gap: Existing methods optimize for *hardware efficiency*, not *zkML constraint counts*. Finite-field arithmetic has different trade-offs: lower bit-widths reduce modular arithmetic complexity, but quantization-aware training must account for field characteristics.

2.4 Activation Function Design

ReLU's non-differentiability at zero and piecewise linearity pose challenges for both optimization and circuit compilation. Recent work explores smooth alternatives: Swish [?], GELU, Mish. However, these still require expensive operations (exp, tanh) in circuits.

Polynomial Activations: x^2, x^3, and low-degree polynomials are circuit-native, requiring only multiplications in finite fields. Prior work evaluates them for training stability [**?**] but not for zkML contexts.

2.5 Zero-Knowledge Machine Learning (zkML)

Early frameworks like zkCNN [**?**] demonstrate feasibility of verifying CNN inference. Recent systems (EZKL [**?**], Modulus Labs [**?**]) provide production tooling with ONNX-to-circuit compilers.

System-Level vs. Model-Level Optimization: All existing work treats neural architectures as *fixed inputs*, optimizing proof systems (folding schemes, lookup tables, commitment schemes). We invert this: treat zkML compilers as fixed infrastructure and optimize *models* for them.

3 Research Questions

1. **Distillation Efficacy**: What accuracy retention can CNN-to-MLP distillation achieve on financial time-series tasks? How does it compare to training MLP from scratch?
   - *Hypothesis*: $> 85\%$ retention based on literature, superior to direct training
2. **Activation Trade-offs**: Which polynomial activation family optimally balances expressiveness and constraint reduction for financial features?
   - *Hypothesis*: Cubic ($x + 0.1x^3$) preserves sign while reducing constraints by $> 30\%$ vs. ReLU
3. **Quantization Strategy**: Does Hessian-guided adaptive quantization outperform uniform quantization under zkML constraints?
   - *Hypothesis*: Adaptive-8bit matches Uniform-16bit accuracy with 50% fewer total bits
4. **Proof Efficiency Gain**: What is the quantitative improvement in proof generation metrics (time, constraints, size) from CNN to optimized MLP?
   - *Hypothesis*: 10-50× speedup, 50-100× constraint reduction
5. **Pareto Optimality**: What configuration (activation × quantization) achieves the best accuracy-efficiency trade-off?
   - *Goal*: Identify deployment-ready configurations for different latency/accuracy requirements

4 Methodology

4.1 Problem Formulation

Task: Given a sequence of market observations $\mathbf{X}_{t-w:t} \in \mathbb{R}^{w \times d}$ where each $\mathbf{x}_i \in \mathbb{R}^d$ contains price/volume features, predict a trading signal $y \in \{0, 1, 2\}$ corresponding to {SELL, HOLD, BUY}.

Features ($d = 8$):

- Price: EMA(5), EMA(10), EMA(20)
- Momentum: RSI(14), MACD

- Volume: Volume MA(5), Volume MA(10)
- Derivatives: Funding rate (perpetual futures)

Labels: Based on forward 1-hour returns:

$$y_t = \begin{cases} 2\,(\text{BUY}) & \text{if } r_{t\to t+1h} > +2\% \\ 0\,(\text{SELL}) & \text{if } r_{t\to t+1h} < -2\% \\ 1\,(\text{HOLD}) & \text{otherwise} \end{cases}$$

Dataset: Bitcoin (BTC/USDT) 1-minute candles from Binance API, 2023-01-01 to 2024-11-08 ($\sim$500K samples). Train/Val/Test split: 70/15/15.

4.2 Baseline: CNN Teacher Model

Architecture:

$$\begin{aligned} \mathbf{X} \in \mathbb{R}^{60\times 8} &\xrightarrow{\text{Conv1D}} \mathbf{H}^{(1)} \in \mathbb{R}^{60\times 64} \\ &\xrightarrow{\text{ReLU}} \\ &\xrightarrow{\text{Conv1D}} \mathbf{H}^{(2)} \in \mathbb{R}^{60\times 128} \\ &\xrightarrow{\text{ReLU}} \\ &\xrightarrow{\text{GlobalAvgPool}} \mathbf{h} \in \mathbb{R}^{128} \\ &\xrightarrow{\text{FC}} \mathbf{z} \in \mathbb{R}^{3} \\ &\xrightarrow{\text{Softmax}} \hat{\mathbf{y}} \end{aligned}$$

Details:

- Conv1D layers: kernel size 5 and 3, stride 1, no padding
- Loss: Cross-entropy with label smoothing ($\epsilon = 0.1$)
- Optimizer: Adam, learning rate 10^{-3} with cosine annealing
- Batch size: 256, early stopping on validation F1-score

Why CNN?

1. CNNs are *proven effective* for time-series: Conv layers learn local temporal patterns (e.g., 5-minute price movements, 3-candle reversal patterns)
2. *Widely deployed* in production trading systems
3. *zkML-unfriendly*: Serves as realistic baseline to demonstrate optimization value

zkML Complexity Estimate:

- Constraint count: $\sim$10M (convolution: 6M, ReLU: 3M, pooling: 1M)
- Proof generation time: $\sim$300 seconds (M1 MacBook Pro, EZKL v10 + Halo2)
- Proof size: $\sim$5 MB

4.3 zkML-Optimized MLP Student

Architecture:

$$
\begin{aligned}
\mathbf{x}_{\text{flat}} \in \mathbb{R}^{480} &\xrightarrow{\text{FC}} \mathbf{h}^{(1)} \in \mathbb{R}^{128} \\
&\xrightarrow{\sigma_{\text{poly}}} \\
&\xrightarrow{\text{FC}} \mathbf{h}^{(2)} \in \mathbb{R}^{64} \\
&\xrightarrow{\sigma_{\text{poly}}} \\
&\xrightarrow{\text{FC}} \mathbf{h}^{(3)} \in \mathbb{R}^{32} \\
&\xrightarrow{\sigma_{\text{poly}}} \\
&\xrightarrow{\text{FC}} \mathbf{z} \in \mathbb{R}^{3} \\
&\xrightarrow{\text{Argmax}} \hat{y}
\end{aligned}
$$

Design Choices:

- Input: Flatten 60×8 sequence to 480-dim vector
- Activation σ_{poly}: Polynomial (varies by experiment, see Section 4.5)
- No batch normalization (requires division in circuits)
- Output: Argmax instead of softmax (avoid division)

Key Constraint: All operations must compile to *efficient* arithmetic circuits (matrix multiplication + polynomial evaluation only).

4.4 CNN-to-MLP Knowledge Distillation

Theoretical Justification:

CNNs learn *local feature extractors*. A Conv1D with kernel size k computes:

$$h_i^{(l)} = \sigma\left(\sum_{j=0}^{k-1} w_j \cdot x_{i+j}^{(l-1)} + b\right)$$

This captures local patterns (e.g., "price rising for 5 consecutive minutes").

Through distillation, the MLP student learns to approximate this via *position-aware weighted combinations*. The flattened input $\mathbf{x}_{\text{flat}}$ preserves positional information, allowing weight matrix $\mathbf{W}^{(1)}$ to implicitly encode:

$$W_{i,j}^{(1)} \approx \text{importance of feature } j \text{ at position } \lfloor j/8 \rfloor \text{ for neuron } i$$

Prior work [?] shows MLPs can learn these implicit positional encodings, achieving 85-92% CNN accuracy on fixed-length sequences.

Loss Function:

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

Components:

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

  $$\mathcal{L}_{\text{KD}} = T^2 \cdot \text{KL}\left(\text{softmax}(z_T/T) \,||\, \text{softmax}(z_S/T)\right)$$

  Higher temperature T softens distributions, transferring more nuanced decision boundaries.

- $\mathcal{L}_{\text{reg}}$: **Novel constraint-aware regularization**:

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

  where:

  - $\|\mathbf{W}\|_1$: Encourages sparsity (fewer non-zero weights = smaller proof via commitment optimizations)
  - Second term: Penalizes weights exceeding quantization threshold τ, facilitating later QAT

Hyperparameters: $\alpha = 0.5$, $\beta = 0.5$, $\gamma = 0.1$, $T = 3$, $\lambda_1 = 10^{-4}$, $\lambda_2 = 10^{-3}$, tuned via grid search on validation set.

4.5 Polynomial Activation Design

Motivation:

ReLU in arithmetic circuits:

$$\text{ReLU}(x) = \max(0, x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Requires a *selection gate*:

$$y = \text{selector}(x \geq 0) \cdot x$$

Each selector adds constraints (comparison + conditional assignment). For a layer with n neurons, this adds $O(n)$ constraints.

Polynomial Alternative:

Polynomials are native to finite-field arithmetic:

$$\sigma_{\text{poly}}(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_d x^d$$

Evaluation requires only d multiplications—*no comparisons*.

Candidates:

1. **Quadratic**: $\sigma(x) = x^2$
   - Pros: Simplest (1 multiplication), always positive
   - Cons: Loses sign information, may hinder classification
2. **Cubic**: $\sigma(x) = x + 0.1x^3$
   - Pros: Preserves sign, bounded derivative ($\sigma'(x) = 1 + 0.3x^2$), smooth
   - Cons: 2 multiplications per neuron

3. **ReLU Polynomial Approximation**: Taylor expansion of $\frac{x+\sqrt{x^2+\epsilon}}{2}$ to degree 3

$$\sigma(x) \approx 0.5x + 0.125x - 0.03125x^3$$

4. **Swish Approximation**: Polynomial fit to $\sigma(x) = x \cdot \text{sigmoid}(x)$

$$\sigma(x) \approx 0.5x + 0.25x^2 - 0.05x^3$$

Training Protocol:

1. Pre-train MLP with ReLU via distillation (warm start)
2. Replace ReLU with polynomial activation
3. Fine-tune with reduced learning rate (10^{-4}) for 10 epochs
4. Apply additional regularization to minimize $(f_{\text{ReLU}}(x) - f_{\text{poly}}(x))^2$ on hidden representations

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

4.6 Adaptive Quantization via Hessian Sensitivity

Problem:

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

- Early layers: Extract low-level features (price differences, volume spikes)—tolerate lower precision
- Later layers: Refine classification boundaries—require higher precision

Method: Hessian-Aware Quantization (HAQ) [**?**]

Step 1 — Sensitivity Measurement:

For each layer l, compute Hessian trace w.r.t. layer weights:

$$S_l = \text{Tr}\left(\nabla^2_{\mathbf{W}_l}\mathcal{L}\right)$$

High S_l indicates high curvature—small weight perturbations (from quantization) cause large loss changes. Use power iteration to approximate:

$$S_l \approx \frac{1}{|\mathcal{D}_{\text{cal}}|} \sum_{\mathbf{x}\in\mathcal{D}_{\text{cal}}} \|\nabla_{\mathbf{W}_l}\mathcal{L}(\mathbf{x})\|^2$$

on a calibration set $\mathcal{D}_{\text{cal}}$.

Step 2 — Bit-width Allocation:

Given total bit-width budget C (e.g., average 8 bits per layer), solve:

$$\min_{\{b_l\}_{l=1}^L} \sum_{l=1}^{L} S_l \cdot Q(b_l) \quad \text{s.t.} \quad \sum_{l=1}^{L} b_l \leq C$$

where $Q(b_l)$ is empirical quantization error for b_l bits, estimated via:

$$Q(b_l) = \mathbb{E}_{\mathbf{W}_l}\left[\left(\mathbf{W}_l - \text{Quant}_{b_l}(\mathbf{W}_l)\right)^2\right]$$

Use dynamic programming or greedy allocation to solve.

Step 3 — Quantization-Aware Fine-tuning (QAT):

1. Simulate quantization during forward pass:

$$\mathbf{W}_l^{\text{quant}} = \text{Quantize}(\mathbf{W}_l, b_l)$$

2. Backward pass uses *straight-through estimator*:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_l} \approx \frac{\partial \mathcal{L}}{\partial \mathbf{W}_l^{\text{quant}}}$$

3. Train for 10 epochs with learning rate 10^{-5}

Baselines: Compare against uniform 4-bit, 8-bit, 16-bit, and FP32.

4.7 zkML Compilation and Deployment

Toolchain: EZKL v10+ with Halo2 proof system
Pipeline:

1. Export PyTorch model → ONNX format
2. Compile ONNX → arithmetic circuit: `ezkl compile model.onnx -o circuit.txt`
3. Generate proving/verifying keys
4. For each inference:
   - Compute output + generate proof (measure prover time)
   - Verify proof (measure verifier time, proof size)
5. Deploy Solidity verifier to Ethereum Sepolia testnet
6. Measure on-chain gas cost

Comparison: Benchmark both CNN teacher and MLP student to quantify efficiency gains.
Metrics:

- **Accuracy**: F1-score (macro-averaged across 3 classes)
- **Constraint Count**: Total R1CS constraints
- **Proof Size**: Bytes (impacts on-chain storage cost)
- **Prover Time**: Seconds on M1 MacBook Pro
- **Verifier Time**: Milliseconds (off-chain throughput)
- **Gas Cost**: Wei (on-chain verification)

5 Experimental Design

5.1 Experiment 1: Distillation Ablation

Goal: Validate distillation effectiveness vs. training from scratch.
Variants:

1. **CNN-Teacher**: Baseline (see Section 4.2)
2. **MLP-Scratch**: Train MLP directly on hard labels (no distillation)
3. **MLP-Hard**: Distillation with $\alpha = 1, \beta = 0$ (only hard labels)
4. **MLP-Soft**: Distillation with $\alpha = 0, \beta = 1$ (only soft labels from teacher)
5. **MLP-Combined**: $\alpha = 0.5, \beta = 0.5$ (balanced)
6. **MLP-zkReg**: Combined + $\gamma = 0.1$ (with zkML regularization)

Metrics: F1-score, parameter count, sparsity (% of zero weights).
Expected Outcome: MLP-zkReg achieves $> 85\%$ of CNN-Teacher F1-score, outperforming MLP-Scratch.

5.2 Experiment 2: Polynomial Activation Comparison

Goal: Identify optimal polynomial activation for financial features.
Setup:

- Fix: MLP architecture, 8-bit uniform quantization
- Vary: Activation function (ReLU, Quadratic, Cubic, ReLU-approx, Swish-approx)

Analysis:

- Plot Pareto frontier: F1-score vs. Constraint Count
- Visualize activation functions: $\sigma(x)$ and $\sigma'(x)$
- Analyze failure modes: dead neurons (always-zero outputs), gradient vanishing

Expected Outcome: Cubic achieves $< 2\%$ accuracy drop vs. ReLU while reducing constraints by $> 30\%$.

5.3 Experiment 3: Quantization Strategy Comparison

Goal: Demonstrate adaptive quantization superiority.
Variants:

- FP32 (baseline)
- 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 (bytes)

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

5.4 Experiment 4: End-to-End zkML Deployment

Goal: Validate practical feasibility and quantify efficiency gains.
Setup:

- Models: CNN-Teacher, MLP-Poly-Quant (best from Exp 2-3)
- Generate 100 proofs on test set
- Deploy verifiers to Sepolia testnet
- Submit 10 proofs on-chain

Measurements:

- Prover time distribution (mean, median, p95)
- Constraint count comparison
- Proof size comparison
- Gas cost comparison

Success Criteria:

- Prover time: MLP $<$ 30 seconds (acceptable for batch trading)
- Gas cost: $<$ 500K gas (economically viable at $50 ETH, $2 tx cost)

6 Expected Results

6.1 Quantitative Predictions

Key Insights:

- Distillation preserves $\sim$87% accuracy (55% vs. 63%)
- Polynomial activation reduces prover time by 75% (15s vs. 60s) with minimal accuracy loss
- Adaptive quantization further reduces prover time by 33% (10s vs. 15s)
- **End-to-end**: 30$\times$ prover speedup (300s $\rightarrow$ 10s), 33$\times$ constraint reduction (10M $\rightarrow$ 300K)

Model	F1-score	Prover Time	Constraints	Proof Size
CNN-Teacher	63%	$\sim$300s	$\sim$10M	$\sim$5MB
MLP-ReLU	55% (87%)	$\sim$60s	$\sim$2M	$\sim$800KB
MLP-Cubic	54% (86%)	$\sim$15s	$\sim$500K	$\sim$200KB
MLP-Cubic-Quant	**52% (83%)**	**$\sim$10s**	**$\sim$300K**	**$\sim$150KB**

Table 1: Expected performance across distillation and optimization stages. Percentages in parentheses indicate retention rate relative to CNN teacher.

6.2 Hypothesis Validation

H1: Distillation Efficacy ($> 85\%$ retention)

- *Likely achieved*: Literature supports 85-92% for CNN→MLP on sequences
- *Contingency*: If $< 85\%$, analyze per-class performance—may succeed on HOLD class (majority)

H2: Cubic Optimality

- *Testable*: Pareto frontier will show if Cubic dominates other polynomials
- *Alternative*: If ReLU-approx performs better, use it instead

H3: Adaptive Superiority

- *Expected*: Hessian analysis will allocate more bits to final layer
- *Visualization*: Heatmap will show bit-width gradient across layers

6.3 Pareto Frontier Analysis

Interpretation:

- Points on frontier: Optimal trade-offs for deployment
- Gap between CNN and MLP: Quantifies cost of verifiability
- Cubic+Quant location: Should approach frontier lower-right (high speed, acceptable accuracy)

7 Timeline

Risk Mitigation:

- If EZKL compilation fails: Focus on constraint count analysis (still valid DL research)
- If distillation $< 80\%$ retention: Analyze why, propose fixes (e.g., increase MLP capacity)
- If time runs short: Drop Exp 4 (on-chain), focus on Exp 1-3 (core DL experiments)

`pareto_placeholder.png`

Figure 1: Expected Pareto frontier: F1-score vs. Prover Time. Color indicates proof size. Markers: activation type. The "efficient frontier" identifies configurations dominating others.

8 Significance and Impact

8.1 Deep Learning Contributions

1. **Distillation for Cryptographic Verifiability**: First systematic study of knowledge distillation optimized for *proof efficiency* rather than hardware efficiency. Establishes methodology for future zkML applications.

2. **Polynomial Activation Theory**: Empirical analysis of polynomial families in financial domain under zkML constraints. Provides design guidelines for activation selection beyond standard ReLU/Sigmoid.

3. **Quantization for Finite-Field Arithmetic**: Extends Hessian-based mixed-precision methods from GPU optimization to zkML context, demonstrating generalization of sensitivity analysis.

8.2 Practical Impact

Trustworthy AI Trading: Enables auditable AI agents where users cryptographically verify trading decisions without trusting intermediaries.

Week	Task	Deliverable
7 (Nov 8-14)	Data collection, EDA, CNN training	CNN teacher checkpoint
8 (Nov 15-21)	MLP training, Exp 1 (distillation ablation)	Distillation results
9 (Nov 22-28)	Exp 2 (activation), Exp 3 (quantization)	Activation + quantization analysis
10 (Dec 1-7)	EZKL compilation, off-chain benchmarking	Proof metrics (CNN vs. MLP)
11 (Dec 8-14)	Exp 4 (on-chain deployment)	Testnet demo, gas benchmarks
12 (Dec 15-21)	Results analysis, Pareto frontier plotting	Comprehensive plots
13 (Dec 22-28)	Report writing	Draft final report
14 (Dec 29-Jan 4)	Presentation preparation	Slides + demo video

Table 2: Project Timeline (Fall 2025)

DeFi Governance: DAO treasuries can use verifiable AI signals for automated rebalancing with provable adherence to declared strategies.

Regulatory Compliance: Financial institutions can demonstrate model provenance to regulators through zero-knowledge proofs, satisfying transparency requirements without revealing proprietary strategies.

8.3 Broader Implications

This work establishes a **general methodology** for adapting high-performance DL models to zkML:

1. Identify zkML-unfriendly components (convolution, ReLU, normalization)
2. Distill to simpler architectures (MLP, polynomial activations)
3. Optimize via adaptive quantization
4. Benchmark end-to-end proof efficiency

Generalizable to: Computer vision (CNN→MLP for verifiable image classification), NLP (Transformer→MLP for verifiable sentiment analysis), speech (RNN→MLP for verifiable voice commands).

8.4 Future Work

- **Multi-task distillation**: Jointly distill multiple financial models (regime, volatility, liquidity) to shared MLP backbone
- **Recursive SNARKs**: Verify model *training* process, not just inference
- **Federated zkML**: Enable collaborative model training across institutions with verifiable aggregation but private data

9 Conclusion

VeriRegime demonstrates that high-performance deep learning models can be systematically transformed into zkML-compatible architectures through knowledge distillation and optimization. By distilling CNN teachers to MLP students, replacing ReLU with polynomial activations, and applying adaptive quantization, we achieve **10-50× proof generation speedup while retaining 83-87% accuracy**—establishing the *accuracy-verifiability Pareto frontier* for financial AI.

Our work shifts zkML research from system-level optimization to **model-level co-design**, opening new avenues for verifiable AI across domains. In the context of autonomous trading agents, this enables a new paradigm: *trustless AI*, where every decision is cryptographically auditable, bridging the gap between algorithmic sophistication and user trust.

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] Urban, G., et al. (2017). Do deep convolutional nets really need to be deep and convolutional? *ICLR 2017*.

[5] Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

[6] Borovykh, A., Bohte, S., & Oosterlee, C. W. (2017). Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691*.

[7] Jacob, B., Kligys, S., Chen, B., et al. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. *CVPR 2018*.

[8] Wang, K., Liu, Z., Lin, Y., Lin, J., & Han, S. (2019). HAQ: Hardware-aware automated quantization with mixed precision. *CVPR 2019*.

[9] Dong, Z., Yao, Z., Gholami, A., Mahoney, M. W., & Keutzer, K. (2019). HAWQ: Hessian aware quantization of neural networks with mixed-precision. *ICCV 2019*.

[10] Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941*.

[11] Goyal, P., et al. (2021). Power series activations in deep learning. *NeurIPS 2021*.

[12] 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*.

[13] EZKL Team. (2024). EZKL: Easy zero-knowledge inference. *https://ezkl.xyz*.

[14] Modulus Labs. (2025). Bringing AI on-chain: zkML in production. *Medium, April 2025*.