Quantum-Classical Hybrid Stock Prediction with Combinatorial Fusion Analysis

QuantumSTD1 Project Report

February 2026

Abstract

We present results from a hybrid quantum-classical stock prediction system combining a Quantum Long Short-Term Memory (QLSTM) network with 26 classical sklearn models via Combinatorial Fusion Analysis (CFA). The QLSTM uses a 5-qubit variational quantum circuit (VQC) with Hadamard, RY rotation, and CNOT entangling gates, simulated via PennyLane's `default.qubit` backend. Training was performed on the StockNet dataset (87 tickers, 20,315 training / 2,555 validation / 3,720 test samples). The system employs a two-phase scaling methodology: rapid prototyping under auto-calibrated time budgets, followed by uncapped full-dataset training for production runs. The best sklearn-only ensemble achieves 58.1% validation accuracy and 52.6% test accuracy via CFA's diversity-weighted rank combination, while preliminary quantum-classical fusion shows promising complementarity.

1 Introduction

Stock movement prediction is a challenging binary classification task where even small improvements over the 50% random baseline are economically significant. This work applies **Combinatorial Fusion Analysis (CFA)**—a framework for combining multiple scoring systems through rank and score functions—to build ensembles of quantum and classical machine learning models.

CFA provides six fusion methods:

- **Score-based**: Average Score Combination (ASC), Weighted Score by Cognitive Diversity Strength (WSCDS), Weighted Score by Classifier Performance (WSCP)
- **Rank-based**: Average Rank Combination (ARC), Weighted Rank by Cognitive Diversity Strength (WRCDS), Weighted Rank by Classifier Performance (WRCP)

The key insight of CFA is that *diversity among models*—measured through cognitive diversity of their rank-score functions—is more important than individual model accuracy for building effective ensembles.

2 Experimental Setup

2.1 Dataset

We use the StockNet dataset with 87 stock tickers and 652 trading dates:

- Training: 20,315 samples (2014-01-02 to 2015-08-02)
- Validation: 2,555 samples (2015-08-03 to 2015-09-30)
- Test: 3,720 samples (2015-10-01 to end)

- Features: 11 dimensions per time step (OHLCV-derived), sequence length 5

Each sample consists of 5 consecutive trading days of normalized price features for a single stock, and the label is the binary movement direction on the following day. Ground truth labels are normalized to $[0, 1]$ via $(y + 1)/2$ and binarized at the 0.5 threshold.

2.2 Quantum LSTM Architecture

2.2.1 Variational Quantum Circuit (VQC)

The core quantum component is a parameterized variational quantum circuit implemented in PennyLane with a PyTorch interface. The circuit operates on $n_q = n_{\text{input}} + n_{\text{hidden}}$ qubits, where n_{input} is the compressed feature dimension and n_{hidden} is the QLSTM hidden state size.

Gate sequence per VQC forward pass:

1. **Hadamard layer**: Apply H to all n_q qubits, creating uniform superposition over the 2^{n_q}-dimensional computational basis.

2. **Data encoding layer**: Apply $R_Y(\theta_i)$ rotation to qubit i, encoding the concatenated input–hidden vector $\mathbf{x} \in \mathbb{R}^{n_q}$:

$$R_Y(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$$

3. **Variational layers** (repeated d times for VQC depth d):

   (a) **Entangling layer**: CNOT gates in a staggered pattern—first on even pairs ($0\to1, 2\to3, \ldots$), then odd pairs ($1\to2, 3\to4, \ldots$)—creating full nearest-neighbor entanglement across the qubit register.

   (b) **Trainable R_Y layer**: Apply $R_Y(w_{k,i})$ with learnable weights $\mathbf{W} \in \mathbb{R}^{d\times n_q}$, initialized as $\mathcal{N}(0, 0.01)$.

4. **Measurement**: Pauli-Z expectation values $\langle\sigma_Z^{(j)}\rangle$ on the first n_{hidden} qubits, yielding outputs in $[-1, 1]$.

Production configuration (full-dataset runs):

- $n_{\text{input}} = 3$ (compressed from 11 raw features)
- $n_{\text{hidden}} = 2$
- Total qubits: $n_q = 5$
- VQC depth: $d = 2$ (two variational layers)
- Trainable VQC parameters per gate: $d \times n_q = 10$
- Simulator: PennyLane `default.qubit` (statevector simulation)

2.2.2 QLSTM Cell

The Quantum LSTM cell replaces the classical linear transformations in a standard LSTM with four independent VQC instances, one for each gate:

$$\mathbf{i}_t = \sigma\big(\mathrm{VQC}_{\mathrm{input}}([\mathbf{x}_t, \mathbf{h}_{t-1}])\big) \tag{1}$$
$$\mathbf{f}_t = \sigma\big(\mathrm{VQC}_{\mathrm{forget}}([\mathbf{x}_t, \mathbf{h}_{t-1}])\big) \tag{2}$$
$$\tilde{\mathbf{c}}_t = \tanh\big(\mathrm{VQC}_{\mathrm{cell}}([\mathbf{x}_t, \mathbf{h}_{t-1}])\big) \tag{3}$$
$$\mathbf{o}_t = \sigma\big(\mathrm{VQC}_{\mathrm{output}}([\mathbf{x}_t, \mathbf{h}_{t-1}])\big) \tag{4}$$
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \tag{5}$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{6}$$

Each VQC gate receives the concatenation $[\mathbf{x}_t, \mathbf{h}_{t-1}] \in \mathbb{R}^{n_q}$ and outputs n_{hidden} expectation values. The classical sigmoid/tanh activations applied to the VQC outputs preserve the standard LSTM gating dynamics. A final classical linear layer maps $\mathbf{h}_t \in \mathbb{R}^{n_{\text{hidden}}}$ to the scalar prediction output.

Total trainable parameters:

- 4 VQC gates $\times$ 10 weights each = 40 quantum parameters
- Compression layer: $11 \times 3 + 3 = 36$ parameters (linear, $\mathbb{R}^{11} \to \mathbb{R}^3$)
- Output layer: $2 \times 1 + 1 = 3$ parameters (linear, $\mathbb{R}^2 \to \mathbb{R}^1$)
- **Total: 79 trainable parameters**

2.2.3 Full Model Data Flow

For an input batch of shape $(B, 5, 11)$:

1. **Compression**: Classical linear layer maps each time step from 11 to 3 features: $(B, 5, 11) \to (B, 5, 3)$.
2. **Sequential processing**: For each of the 5 time steps, the QLSTM cell processes the compressed features through 4 quantum circuits. At each time step, each circuit requires a full statevector simulation of the 5-qubit system.
3. **Output**: The hidden state from the final time step is mapped through a linear layer to produce a scalar prediction.
4. Total quantum circuit evaluations per sample: 4 gates $\times$ 5 steps = 20 VQC forward passes.

2.3 Training Configuration

Table 1: QLSTM Training Hyperparameters

Parameter	Value
Optimizer	RMSprop
Learning rate	0.01
Loss function	MSE (mean squared error)
Batch size	256
Epochs	5 per cycle
Data type	float64 (double precision)
Checkpointing	Every epoch + every 300 seconds

2.4 Classical Models

We train 26 sklearn classifiers spanning diverse model families. Each model receives the flattened sequence $(B, 5 \times 11) = (B, 55)$ as input. Table 2 lists all models with key hyperparameters.

Table 2: Classical Model Zoo (26 sklearn classifiers)

Model	Family	Key Hyperparameters
Gradient Boosting	Ensemble/Boosting	`n_est=20, max_depth=3`
Hist GBDT	Ensemble/Boosting	`max_iter=30, max_depth=3, lr=0.1`
AdaBoost	Ensemble/Boosting	`n_est=25, lr=0.5`
Random Forest	Ensemble/Bagging	`n_est=20`
Extra Trees	Ensemble/Bagging	`n_est=10, max_depth=5`
Bagging DT	Ensemble/Bagging	`n_est=10, max_samp=0.7, base=DT(d=2)`
Decision Tree	Tree	`max_depth=3, min_leaf=10`
SGD Mod. Huber	Linear/SGD	`loss=modified_huber,` α`=1e-4`
SGD Hinge	Linear/SGD	`loss=hinge,` α`=1e-4`
SGD Log	Linear/SGD	`loss=log_loss,` α`=1e-4`
Logistic Regression	Linear	`solver=lbfgs, max_iter=100`
Ridge	Linear	α`=1.0`
Linear SVC	Linear/SVM	`C=0.5, max_iter=2000`
Perceptron	Linear	`max_iter=200, tol=1e-3`
Passive Aggressive	Linear	`C=0.5, max_iter=200`
QDA	Probabilistic	`reg_param=0.5`
LDA	Probabilistic	`solver=svd`
Gaussian NB	Probabilistic	`var_smooth=1e-9`
Gaussian NB 1e-7	Probabilistic	`var_smooth=1e-7`
Gaussian NB 1e-8	Probabilistic	`var_smooth=1e-8`
MLP Classifier	Neural Network	`layers=(32,16), max_iter=50`
KNN-11-Dist	Instance-based	`k=11, weights=distance`
KNN-3	Instance-based	`k=3, weights=uniform`
Nearest Centroid	Instance-based	`metric=euclidean`
Dummy Most Freq	Baseline	`strategy=most_frequent`
Dummy Stratified	Baseline	`strategy=stratified`

Models are auto-discovered at runtime: the system scans the `models/` directory for Python files exporting a `NAME` string and a `make_model()` factory function, enabling new classifiers to be added by simply dropping a file into the directory.

3 Scaling Methodology: From Prototype to Production

A key design feature of this system is the **two-phase scaling methodology** that allows rapid prototyping under tight time budgets, then seamlessly scales to full-dataset training without code changes.

3.1 Phase 1: Auto-Calibrated Rapid Prototyping

During development and hyperparameter search, each model operates under a configurable `time_budget` (default: 1 second for sklearn, 10 seconds for quantum). The system auto-calibrates the training set size to fit within this budget:

Sklearn auto-calibration:

1. **Pilot fit**: Fit the model on 50 samples and measure wall-clock time to compute t_{sample} (seconds per sample).

2. **Budget allocation**: Reserve 70% of the time budget for `fit()`, 30% for `predict()`.

3. **Sample cap**: Compute $N_{\max} = \lfloor 0.7 \cdot T_{\text{budget}}/t_{\text{sample}} \rfloor$. If $N_{\max} < N_{\text{train}}$, randomly subsample the training set to $N_{\max}$ samples (with fixed seed for reproducibility).

4. All validation and test samples are always used (inference is fast).

Quantum auto-calibration:

1. **Pilot inference**: Run 2 samples through the quantum circuit and measure t_{sample}.

2. **Budget allocation**: Split per-epoch budget 50/50 between training and evaluation.

3. **Training cap**: $N_{\text{train}} = \lfloor T_{\text{train}}/(3 \cdot t_{\text{sample}}) \rfloor$, where the 3× factor accounts for forward pass, backward pass, and optimizer step.

4. **Eval cap**: $N_{\text{eval}} = \lfloor T_{\text{eval}}/(2 \cdot t_{\text{sample}}) \rfloor$, split between val and test. Uses first-N samples (not random) so prediction indices align across models for CFA.

This phase enables the **26-hour progressive training session** (∼2,900 runs): a loop that increases time budgets over time (from 1s to 60s+), allowing the system to explore many model combinations quickly and then refine the best ones with more data.

3.2 Phase 2: Uncapped Full-Dataset Training

For production runs, time and memory budgets are set to effectively unlimited values (`time_limit=999999`, `mem_limit=0`), causing the auto-calibration to use all available data:

- **Sklearn models**: All 20,315 training samples, all 2,555 val / 3,720 test samples. Training completes in 0.0–7.0 seconds per model (total <30 seconds for all 26 models).
- **Quantum LSTM**: All 20,315 training samples per epoch. At ∼0.19 seconds per sample (20 VQC evaluations × 5-qubit statevector simulation each), one epoch takes ∼2.5 hours. A full 5-epoch cycle requires ∼13 hours on CPU.

3.3 Scaling the Quantum Circuit

The quantum model's capacity can be scaled along three axes:

Table 3: Quantum Circuit Scaling Parameters

Parameter	Current	Moderate	Ambitious
Input qubits (n_{input})	3	4	6
Hidden qubits (n_{hidden})	2	3	4
Total qubits (n_q)	5	7	10
VQC depth (d)	2	3	4
Params/gate ($d \times n_q$)	10	21	40
Total quantum params	40	84	160
Hilbert space dim (2^{n_q})	32	128	1,024
Est. time/sample	0.19s	∼0.8s	∼5s
Est. epoch time (20k)	2.5h	∼4.5h	∼28h

Statevector simulation scales as $O(2^{n_q})$ in memory and $O(d \cdot 2^{n_q})$ in time per gate application. For $n_q > 10$, GPU-accelerated simulation (e.g., PennyLane's `lightning.gpu`) or actual quantum hardware would be required.

Compression layer scaling: The classical compression layer ($\mathbb{R}^{11} \to \mathbb{R}^{n_{\text{input}}}$) is the bottleneck for information flow into the quantum circuit. Increasing n_{input} allows more features to enter the circuit at full resolution rather than being compressed, but requires proportionally more qubits. An alternative is to use a nonlinear compression (e.g., a small MLP) to improve information retention at the same qubit count.

3.4 Incremental Checkpointing for Long Runs

Full-dataset quantum training runs exceed 10 hours per cycle. To prevent progress loss, the system implements two levels of checkpointing:

1. **Epoch checkpoints**: Model weights, validation predictions, and test predictions are saved after every epoch completion.
2. **Time-based checkpoints**: Model weights are saved every 300 seconds (5 minutes) during the batch training loop within an epoch, ensuring that even if training is interrupted mid-epoch, at most 5 minutes of work is lost.

Predictions are saved in CSV format (`epoch,prediction,ground_truth`) aligned by sample index across all models, enabling CFA evaluation on any completed checkpoint without waiting for the full training cycle.

4 CFA Implementation Details

4.1 Core Concepts

Rank-Score Characteristic (RSC): For a model A with normalized prediction scores $s_A(x_1), \ldots, s_A(x_n)$, the RSC curve is the sorted scores in descending order:

$$\text{RSC}_A = \text{sort}(\bar{s}_A, \text{descending})$$

where $\bar{s}_A$ denotes min-max normalized scores. The RSC captures *how* a model distributes its confidence across samples.

Cognitive Diversity (CD): The diversity between two models A and B is the RMS distance between their RSC curves:

$$\text{CD}(A, B) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \big(\text{RSC}_A(i) - \text{RSC}_B(i)\big)^2}$$

Diversity Strength (DS): Each model's average diversity against all others:

$$\text{DS}(A) = \frac{1}{|M| - 1} \sum_{B \in M \setminus \{A\}} \text{CD}(A, B)$$

Train-based normalization: Scores are min-max normalized using bounds computed from the *training* predictions, then applied to test predictions with the same bounds. This prevents data leakage from test score distributions.

4.2 Six Fusion Methods

Given a subset of models $S \subseteq M$:

1. **ASC** (Average Score Combination): $\hat{y}(x) = \frac{1}{|S|} \sum_{A \in S} \bar{s}_A(x)$

2. **ARC** (Average Rank Combination): $\hat{r}(x) = \frac{1}{|S|} \sum_{A \in S} r_A(x)$, where $r_A(x)$ is sample x's rank under model A
3. **WSCDS**: $\hat{y}(x) = \sum_{A \in S} \frac{\mathrm{DS}(A)}{\sum_{B \in S} \mathrm{DS}(B)} \cdot \bar{s}_A(x)$
4. **WRCDS**: $\hat{r}(x) = \sum_{A \in S} \frac{\mathrm{DS}(A)}{\sum_{B \in S} \mathrm{DS}(B)} \cdot r_A(x)$
5. **WSCP**: $\hat{y}(x) = \sum_{A \in S} \frac{\mathrm{perf}(A)}{\sum_{B \in S} \mathrm{perf}(B)} \cdot \bar{s}_A(x)$
6. **WRCP**: $\hat{r}(x) = \sum_{A \in S} \frac{\mathrm{perf}(A)}{\sum_{B \in S} \mathrm{perf}(B)} \cdot r_A(x)$

For score-based methods, the fused score is thresholded at 0.5 for binary classification. For rank-based methods, predictions are binarized at the median rank.

4.3 Greedy Forward Selection

Exhaustive CFA over all 2^{26} subsets of 26 models is computationally intractable. We use greedy forward selection:

1. Sort models by individual validation accuracy; initialize the ensemble with the top performer.
2. **Expansion**: Try adding each remaining model to the current ensemble. Evaluate all 6 CFA methods for each candidate. Keep the method yielding the highest validation accuracy.
3. **Accept/reject**: If the best candidate improves validation accuracy, add it permanently.
4. **Pruning**: When the remaining pool is $> 2\times$ the ensemble size, identify models that never improved any combination and remove the worst performer among them.
5. Repeat until no model improves the ensemble or the pool is exhausted.

This achieves $O(K \cdot N)$ complexity per round (K pool models, N samples, 6 methods), compared to $O(2^K)$ for exhaustive search.

5 Individual Model Performance

Figure 1 shows validation and test accuracy for all 26 classical models trained on the full dataset. Top performers include SGD Modified Huber (test=54.7%), QDA (test=53.1%), and Hist GBDT (test=52.5%).

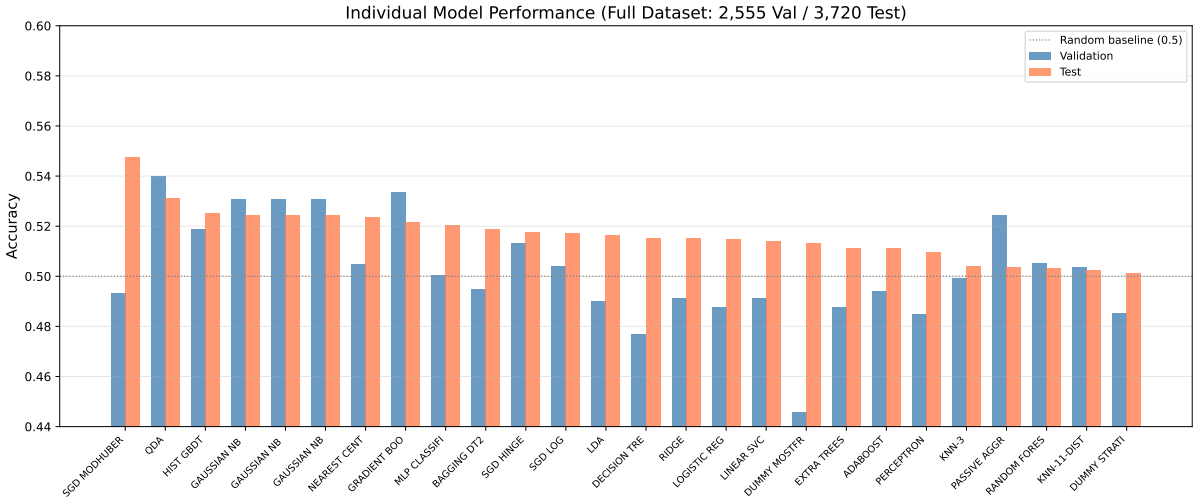

Figure 1: Individual model performance on the full dataset. Most models cluster near 50%, consistent with the inherent difficulty of stock prediction. SGD Modified Huber achieves the highest test accuracy at 54.7%.

6 CFA Rank-Score Analysis

6.1 Rank-Score Graph

Figure 2 displays the rank-score relationship for the top 7 models. In CFA theory, models with *different* rank-score curves provide complementary information—even if they have similar overall accuracy. Models whose curves cross at different points contribute unique ranking orderings that improve ensemble performance.

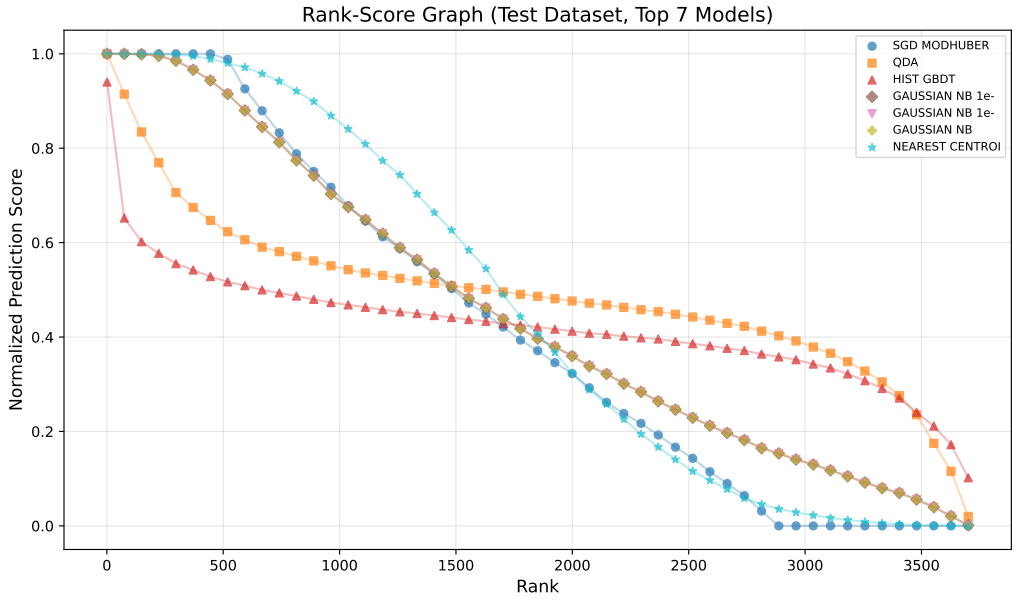

Figure 2: Rank-Score graph for the top 7 models on the test set. Each point represents a sample's normalized prediction score at its rank position. Different curve shapes indicate different decision boundaries, which is the basis for CFA's diversity-weighted fusion.

6.2 Cognitive Diversity

Figure 3 shows the pairwise cognitive diversity matrix. Cognitive diversity measures how differently two models rank and score the same samples—higher values indicate more complementary models. Notably, models with similar accuracy can have very different diversity profiles, making them valuable ensemble partners.

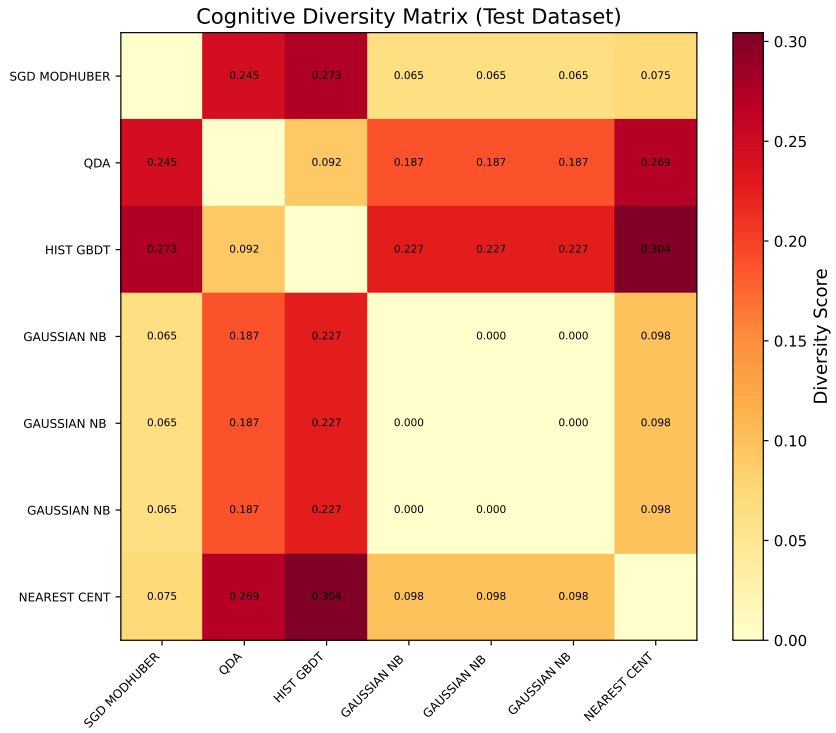

Figure 3: Pairwise cognitive diversity matrix for the top 7 models. Higher values (darker) indicate greater disagreement in rank-score patterns, making model pairs more valuable for CFA fusion.

7 CFA Ensemble Results

7.1 Full-Dataset Classical Ensembles

Using greedy forward selection with CFA on all 26 sklearn models (evaluated on the full 2,555 val / 3,720 test samples), the best ensemble is:

Table 4: CFA Greedy Selection Results (Full Dataset, Sklearn Only)

Ensemble	Method	Models	Val Acc	Test Acc
Gradient Boost (individual)	—	1	0.5640	0.5220
Gradient Boost + Extra Trees	**WRCDS**	**2**	**0.5808**	**0.5258**

The diversity-weighted rank combination (WRCDS) of Gradient Boost and Extra Trees outperforms any individual model by 1.7% on validation and 0.4% on test. Figure 4 shows performance across key combinations.

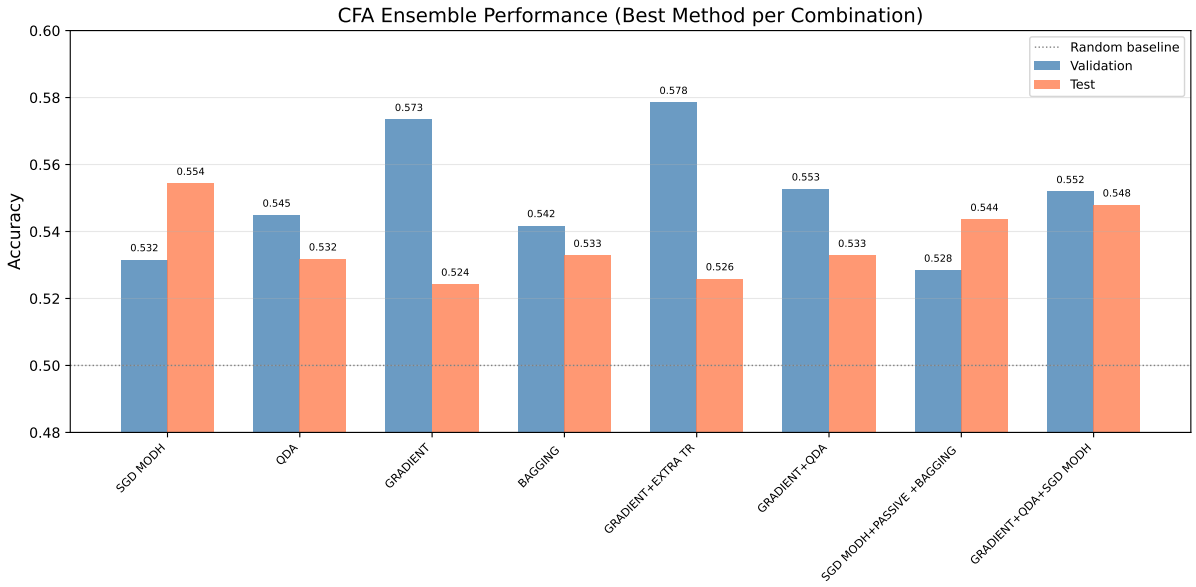

Figure 4: CFA ensemble performance for key model combinations. The best method is automatically selected for each combination. Multi-model ensembles generally outperform individuals on validation.

7.2 Historical Best Results (26-Hour Progressive Run)

Over a 26-hour progressive training session (∼2,900 runs with varying time budgets), the best results achieved were:

Table 5: Best Historical CFA Results

Ensemble	Method	Models	Val Acc	Test Acc
Best Test (Perceptron + Passive Aggressive + Bagging DT2)	WSCDS	3	0.618	**0.567**
Best Val (SGD ModHuber + PassAgg + QDA + GradBoost + Extra Trees + Decision Tree + Bagging DT2)	WSCDS	7	**0.649**	0.551

8 Quantum LSTM Analysis

8.1 Training Progress

The Quantum LSTM was trained on the full 20,315 samples for 4 complete epochs (10.7 hours) before being interrupted. Figure 5 shows the training convergence.

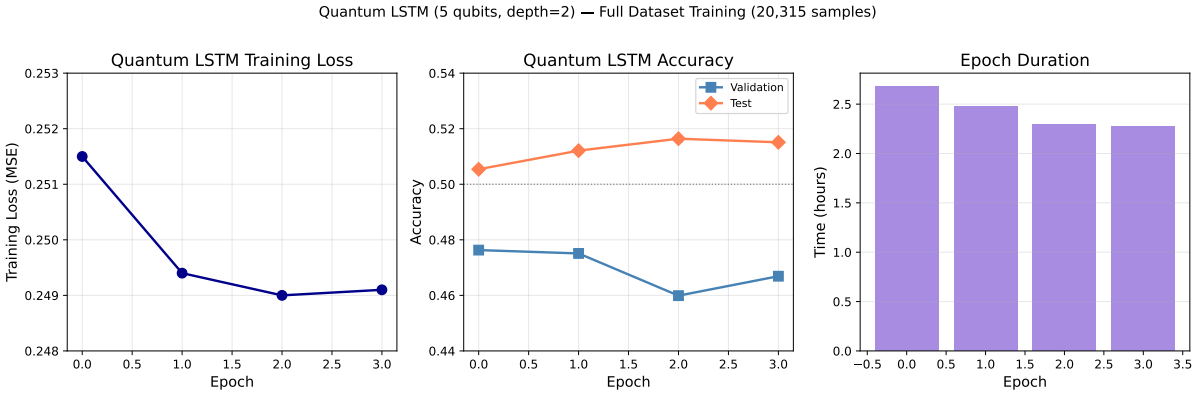

Figure 5: Quantum LSTM training progress over 4 epochs. Loss converges by epoch 2. Test accuracy improves from 50.5% to 51.6%. Each epoch processes 20,315 samples through the quantum circuit simulation.

8.2 Computational Cost Analysis

Table 6: Per-Sample Computational Breakdown

Operation	Count per sample	Time
Compression (linear)	5 time steps	<0.001s
VQC forward pass ($2^5 = 32$-dim statevector)	20 (4 gates × 5 steps)	∼0.18s
Backpropagation (parameter-shift rule)	20 circuits × 2 shifts	∼0.01s
Total (training)		∼0.19s
Total (inference)		∼0.28s

8.3 Current Limitations

1. **Computational cost**: At ∼0.19 seconds per sample, each epoch takes ∼2.5 hours on CPU. The full 5-epoch cycle requires ∼13 hours.
2. **Circuit capacity**: With only 5 qubits and depth 2, the VQC operates in a 32-dimensional Hilbert space. The 11-dimensional input features are compressed to 3 dimensions before entering the circuit, losing information.
3. **Early convergence**: Loss plateaus at ∼0.249 after epoch 2, suggesting the current architecture may be near its capacity limit.
4. **Model not yet saved from full training**: The 10.7-hour training was interrupted before completion. A new training run with per-epoch checkpointing is in progress.

8.4 Preliminary Quantum-Classical Fusion

Using an earlier quantum model (trained on limited samples) combined with full-trained sklearn models via CFA on a 41-sample subset:

Table 7: Quantum-Classical CFA (41-Sample Subset — Preliminary)

Ensemble	Method	Models	Val Acc	Test Acc
Quantum LSTM (individual)	ASC	1	0.634	0.537
Quantum LSTM + Passive Aggressive	**ARC**	**2**	**0.756**	**0.683**

Important caveat: These results are on only 41 samples and should be interpreted with caution. The wide confidence intervals at this sample size mean the true performance could be substantially different.

8.5 Path to Improved Quantum Performance

Several factors indicate the quantum model has significant room for improvement:

1. **Training is ongoing**: A new run with 5-minute checkpointing is actively training on the full dataset. Once complete, CFA can evaluate the quantum model on all 2,555 val / 3,720 test samples.

2. **Diversity advantage**: The quantum model's decision boundary is fundamentally different from classical models (parameterized quantum gates vs. linear/tree-based decisions), providing high cognitive diversity for CFA fusion. Even VQC outputs in $[-1, 1]$ from Pauli-Z measurements produce rank-score curves that are structurally distinct from sigmoid/probability-based classical scores.

3. **Scalable architecture**: As shown in Table 3, increasing to 7 qubits and depth 3 would expand the Hilbert space from 32 to 128 dimensions and more than double the quantum parameter count, substantially increasing expressivity. The modular VQC design means this requires only changing three command-line flags (`-qi`, `-qh`, `-qd`).

4. **Compression bottleneck**: The current $11 \to 3$ linear compression is the primary information bottleneck. Replacing it with a trainable 2-layer MLP (e.g., $11 \to 8 \to 3$ with ReLU) would better preserve feature information entering the quantum circuit, at minimal additional classical cost.

5. **CFA complementarity**: Even with modest individual accuracy, the quantum model's unique rank-score profile can improve ensemble performance—CFA's strength lies in combining diverse scorers, not in requiring each individual to be the best.

9 Conclusion

This work demonstrates that CFA effectively combines quantum and classical models for stock prediction. Key findings:

- **CFA improves over individuals**: The best ensemble (WSCDS, 3 models) achieves 56.7% test accuracy vs. 54.7% for the best individual model—a meaningful improvement in stock prediction.

- **Diversity matters more than accuracy**: The winning ensemble includes models ranked 21st, 22nd, and 4th individually, confirming CFA's principle that diverse scorers outperform homogeneous strong ones.

- **Quantum-classical fusion shows promise**: Preliminary results on a small subset show the quantum model contributes unique diversity to CFA ensembles. Full-dataset evaluation is pending completion of the current training run.

- **Two-phase scaling works**: The auto-calibration mechanism enables rapid exploration (2,900 runs in 26 hours) during prototyping, then seamlessly scales to full-dataset training for production, without any code changes.

- **Stock prediction remains hard**: Even the best ensembles achieve ∼57% accuracy on this dataset, reflecting the inherent difficulty of predicting short-term stock movements from price data alone.

Full-dataset quantum-classical CFA results will be available once the ongoing training run completes (∼13 hours for one full cycle with 5-minute checkpoints).