A Comparative Study of Classical and Quantum Machine Learning for Large-Scale Financial Forecasting

Author: Rachel Scott, **Affiliation:** Assistant Professor, Department of Quantum Computing, University of Oxford, UK. **Email:** rachel.scott@ox.ac.uk

Abstract

Financial forecasting at large scale covering cross-asset returns, volatility surfaces, and high-frequency microstructure remains a central and challenging problem for both academia and industry. Classical machine learning (ML) and deep learning methods (e.g., gradient-boosted trees, recurrent and attention-based networks) have delivered notable advances in predictive accuracy and trading performance, but they face limits in sample efficiency, feature expressivity for extremely high-dimensional inputs, and in solving certain kernel-like inference problems at scale. Quantum machine learning (QML) techniques principally quantum kernel methods and variational quantum circuits (quantum neural networks, QNNs) offer alternative inductive biases and potentially new computational primitives that may be relevant for financial forecasting, particularly in small-label/high-dimension regimes or combinatorial subproblems such as portfolio optimization.

This manuscript presents a comprehensive comparative study framework for classical and quantum ML approaches applied to large-scale financial forecasting. We (1) survey theoretical foundations, (2) propose reproducible experimental and evaluation protocols, (3) delineate architecture patterns and practical engineering considerations for both classical and quantum approaches, (4) prescribe robust backtesting and risk-aware evaluation metrics, and (5) discuss empirical expectations, limitations, and a prioritized research agenda. We include implementation pseudocode, data preprocessing recommendations, hyperparameter tuning strategies, and considerations for quantum hardware constraints (NISQ devices).

Keywords: financial forecasting; quantum machine learning; variational quantum circuits; quantum kernels; time-series forecasting; XGBoost; transformers; backtesting; NISQ; reproducible experiments

1. Introduction

1.1 Motivation

Large-scale financial forecasting predicting asset returns, volatility, liquidity metrics, or order-flow behavior across hundreds to thousands of instruments underpins pricing, risk management, algorithmic trading, and portfolio construction. Classical ML methods

(ensemble trees, neural networks, attention models) scale well and exploit modern compute but often require extensive labeled data, careful feature engineering, and robust regularization to avoid overfitting in non-stationary markets (Goodfellow et al., 2016; Chen & Guestrin, 2016). In contrast, QML proposes new representational mechanisms (quantum feature maps, Hilbert-space embeddings) and variational circuits that might provide advantages in expressivity or sample complexity for certain structured tasks (Biamonte et al., 2017; Havlíček et al., 2019; Cerezo et al., 2021). Given rapid advances and the practical constraints of near-term quantum hardware (NISQ devices), a methodical comparative evaluation is needed to clarify where, how, and when quantum methods could be meaningfully applied in financial forecasting.

1.2 Objectives and contributions

This manuscript's objectives are threefold:

1. **Framework & Diagnostics:** Provide a rigorous framework to compare classical and QML methods on large-scale forecasting tasks, including data schemas, preprocessing, modeling pipelines, and evaluation metrics tailored to finance (Samuel, 2022).
2. **Algorithms & Implementation:** Present reproducible algorithmic templates for representative classical baselines (ARIMA/GARCH, XGBoost, LSTM, Temporal Fusion Transformer) and quantum methods (quantum kernels, variational quantum circuits for regression/classification), including training details, hyperparameter guidance, and computational budgeting.
3. **Experimental Blueprint & Roadmap:** Define datasets (public and synthetic), walk-forward evaluation protocols, risk-aware backtesting measures, and a prioritized research agenda emphasizing reproducibility, interpretability, and realistic quantum resource accounting (Fatunmbi, 2021).

1.3 Paper organization

Section 2 reviews related work. Section 3 formalizes problem definitions and datasets. Section 4 details classical ML methods. Section 5 develops QML approaches and practical NISQ considerations. Section 6 describes the experimental design and evaluation protocol. Section 7 outlines implementation notes, computational costing, and software frameworks. Section 8 discusses expected empirical behaviors, interpretability, and risk management implications. Section 9 identifies open challenges, limitations, and a prioritized roadmap. Section 10 concludes.

2. Related work

Research at the intersection of classical ML for finance and QML is growing. Classical forecasting leverages both statistical time-series modeling (ARIMA, GARCH) and modern ML: gradient boosting (XGBoost), random forests, LSTM/RNNs, and Transformer-based models (Vaswani et al., 2017; Lim et al., 2021) for multi-horizon forecasting. Feature engineering (technical indicators, liquidity metrics, order-book microstructure) and careful evaluation (walk-forward, transaction-cost aware backtests) are central to real-world success (Goodfellow et al., 2016; Chen & Guestrin, 2016).

Quantum machine learning literature explores quantum kernels (Havlíček et al., 2019), variational quantum circuits (VQC/QNN) (Cerezo et al., 2021), and hybrid quantum-classical optimization (Preskill, 2018). Early experiments apply QML to classification/regression benchmarks and small finance problems (classification of credit risk, portfolio selection prototypes) but large-scale forecasting applications remain nascent. Reviews by Biamonte et al. (2017) and Meyer et al. (2022/2024 preprints) survey quantum algorithms and quantum reinforcement learning; Cerezo et al. (2021) document variational algorithm design issues (barren plateaus, noise mitigation).

3. Problem formulation and data

3.1 Forecasting tasks considered

We focus on three canonical forecasting problems that capture common industry needs:

1. **Cross-Sectional Next-Day Return Forecasting (Regression):** Predict next-day returns (r_{i,t+1}) for a universe of (N) assets given historical features (X_{i,t-L+1:t}) and cross-asset signals.
2. **Volatility and Value-at-Risk (VaR) Forecasting (Distributional/Quantile):** Forecast conditional volatility (\sigma_{i,t+h}) or quantiles for risk management over multiple horizons.
3. **Multi-Horizon Liquidity & Order-Flow Forecasting (Sequence-to-Sequence):** Multi-step prediction of market depth metrics, fill rates, or order imbalance to support execution algorithms.

These tasks vary in label density, noise properties, and stationarity, providing a representative testbed.

3.2 Data sources and schema

For a comparative study we recommend the following data types:

- **Daily/Intraday Price Series:** OHLCV time series for equities, FX, or futures.
- **Microstructure Features:** Best bid/ask depth, trades, quote changes (for intraday liquidity).

- **Fundamental/Alternative Signals:** Earnings surprises, macro indicators, sentiment scores.
- **Cross-Asset Signals:** Sector indices, FX rates, yield curves.

Public datasets for replicable research include: CRSP/Compustat (academic access), Quandl, Yahoo Finance (prices), and LOBSTER for limit-order book data. Synthetic datasets that preserve stylized facts (GARCH volatility clustering, jumps, heteroskedasticity) are important for stress testing.

Data representation: For each asset (i) and time (t), construct an input tensor (X_{i,t}) containing lagged returns, technical indicators, cross-sectional rank features, and exogenous macro variables. Normalization (z-score, rolling winsorization) and stationarity transforms (log returns) are standard.

3.3 Evaluation regimes and non-stationarity

Markets are non-stationary; therefore evaluation must use **walk-forward cross-validation** with rolling retraining windows and out-of-sample testing. We adopt a rolling window approach with train/validation/test splits that move forward in time, and we apply **temporal nested cross-validation** for robust hyperparameter selection (Samuel, 2023).

4. Classical machine learning methods

This section sets out baseline classical methods from statistical to deep learning that will be compared with QML.

4.1 Statistical time-series baselines

- **ARIMA/GARCH family:** Useful for single-series baselines and volatility modeling; specify model orders via AIC/BIC and use rolling re-estimation to cope with non-stationarity.
- **State-Space/Kalman filters:** Useful for latent factor tracking and dynamic modeling of unobserved components (e.g., regime switches).

While statistically interpretable, these methods struggle with incorporate large heterogeneous features across many assets.

4.2 Tree ensembles and feature-based learners

- **XGBoost/LightGBM (Chen & Guestrin, 2016):** Strong baselines for tabular features; robust to missing data and effective with careful feature engineering.
- **Random Forests, CatBoost:** Alternative ensemble methods; provide feature importance measures.

Key hyperparameters: number of trees, learning rate, max depth, subsampling; tune via time-aware cross-validation.

4.3 Deep learning models

- **LSTM / GRU (Hochreiter & Schmidhuber, 1997):** Sequence modeling for temporal dependencies.
- **Temporal Fusion Transformer (TFT) & Transformers (Vaswani et al., 2017; Lim et al., 2021):** Attention mechanisms for multi-horizon forecasting; TFT provides interpretable attention and gating components well-suited to time-series.
- **Hybrid CNN-LSTM:** For order-book images or short-term microstructure patterns.

Training considerations: regularization (dropout, weight decay), sequence length selection, batch normalization for sequences, and early stopping conditioned on forward walk-forward validation.

4.4 Model ensembling and stacking

Ensembles across model families (statistical + tree + deep) often deliver improved stability via error decorrelation. Use stacking with out-of-fold predictions in the temporal domain respecting time ordering.

5. Quantum machine learning approaches

5.1 Quantum kernels and kernel methods

Quantum kernel methods map classical data into a quantum Hilbert space via feature maps (\Phi: x \mapsto |\Phi(x)\rangle) realized by parameterized circuits, and compute kernel entries (K(x,x') = |\langle\Phi(x)|\Phi(x')\rangle|^2) or expectation-value kernels (Havlíček et al., 2019). Kernel ridge regression or kernel SVM can be applied for regression/classification.

Prospects: In high-dimension/small-label regimes, quantum kernels may produce separability not achieved by classical kernels, potentially improving generalization (Havlíček et al., 2019). However, kernel evaluation cost grows with dataset size (requires many kernel evaluations), making scalability a key constraint.

5.2 Variational quantum circuits (QNNs) for regression

VQCs (QNNs) parameterize a circuit (U(\theta)) acting on an encoded state (|\Phi(x)\rangle); measurement yields features (y(x;\theta)) used for downstream predictions. For regression, a simple architecture is:

1. Encode (x) into qubits (angle or amplitude encoding).

2. Apply layers of parameterized single- and two-qubit gates (ansatz).
3. Measure a set of observables; aggregate into scalar/regression output via classical linear readout.

Training: Optimize (\theta) via classical optimizers using the parameter-shift rule (Cerezo et al., 2021). Shot-noise and barren plateau issues necessitate careful ansatz design and gradient variance control.

5.3 Hybrid quantum-classical architectures

Practical near-term architectures use hybridization:

- **Classical encoder + quantum latent + classical decoder:** Classical networks compress high-dimensional inputs into compact vectors (d ≈ 8–64) that are encoded into QNNs for richer transformations, then decoded by classical heads.
- **Quantum feature augmentations:** Compute quantum kernel features offline for a subset of training data and combine with classical features through concatenation.

Figure (conceptual) hybrid pipeline: Input → classical preprocessing → quantum module (kernels / VQC) → classical ML head → prediction.

5.4 Encoding strategies and resource tradeoffs

- **Angle encoding:** Map components of feature vector into rotation angles on single qubits linear in dimension.
- **Amplitude encoding:** Potentially exponentially compress features into amplitudes, but state preparation is costly.
- **Basis encoding:** Encode discrete/categorical info into computational basis states.

Resource tradeoffs (qubits, circuit depth, shots) guide encoding choice: amplitude encoding may be impractical on NISQ devices due to state preparation cost, while angle encoding is hardware-efficient.

5.5 NISQ constraints and mitigation strategies

- **Barren plateaus:** Use problem-inspired ansatz and local cost functions to mitigate vanishing gradients (Cerezo et al., 2021).
- **Noise & error mitigation:** Zero-noise extrapolation, measurement error mitigation, and readout calibration can improve effective fidelity.

- **Shot budgeting:** Efficiently schedule shots (measurements) during training; use higher shots for critic/validation and fewer shots for exploratory updates (Fatunmbi, 2022).

6. Experimental design and evaluation protocol

This section provides a detailed, reproducible plan for experiments comparing classical and quantum methods.

6.1 Datasets and pre-processing

Recommended datasets:

- **Daily cross-section:** S&P 500 constituents price and fundamentals from 2005–2020.
- **Intraday microstructure:** LOBSTER limit order book snapshots for selected equities (for liquidity forecasting).
- **Synthetic scenarios:** Simulated multi-asset series incorporating stylized features (regime switches, jumps, correlated shocks) for stress tests.

Preprocessing pipeline: time alignment, missing value handling (forward fill with indicator), log returns, rolling normalization (z-score over lookback window), and categorical encoding for event flags. Maintain strict chronological order.

6.2 Train/validation/test splits

Use **rolling windows** (e.g., 3-year training, 6-month validation, 6-month test) that advance monthly. For hyperparameter tuning, use nested rolling cross-validation to avoid look-ahead bias.

6.3 Metrics

- **Point forecasts:** MSE, RMSE, MAE, directional accuracy (hit ratio).
- **Value-oriented metrics:** Information ratio, Sharpe ratio of strategy built on predictions, P&L after realistic transaction costs and slippage.
- **Risk metrics (volatility forecasts):** QLIKE loss, VaR exceedance frequency.
- **Stability/robustness:** Performance decay over out-of-sample horizons, sensitivity to covariate shift, adversarial robustness tests.
- **Computational & resource metrics:** wall-clock time, CPU/GPU hours (classical), QPU time and shots (quantum), memory footprint.

6.4 Baseline configurations and hyperparameters

Classical baselines:

- ARIMA/GARCH: select orders via AIC/BIC.
- XGBoost: tune n_estimators [100–2000], learning_rate [0.01–0.3], max_depth [3–12].
- LSTM: layers 1–3, hidden size 64–512, learning rate 1e-4–1e-3.
- Transformer/TFT: model dim 64–512, heads 4–8, lookback 128–512.

Quantum setups:

- **Quantum kernel SVM:** choose feature maps: hardware-efficient vs problem-aware; use classical kernel baselines (RBF, polynomial) tuned similarly.
- **VQC regression:** ansatz depth $L \in [1, 6]$, qubits $q \in [4, 20]$ (depending on hardware/simulator), shots $S \in \{256, 512, 2048\}$.

Hyperparameters tuned via grid/randomized search within temporal cross-validation.

6.5 Training regime and reproducibility

- Seed control for RNGs, deterministic data splits logged.
- Use containerized experiments (Docker) and notebooks with exact dependency versions (Python, PennyLane/Qiskit for quantum runs, PyTorch/TF, XGBoost).
- Use experiment tracking (MLflow/W&B) to log metrics, parameters, and artifacts.
- For quantum runs, report hardware backend, calibration data (T1/T2, gate errors), and shot counts.

6.6 Statistical comparison

Use paired tests across rolling windows (Diebold-Mariano test for forecast accuracy differences; bootstrap confidence intervals for Sharpe ratio differences) to establish significance.

7. Implementation details and pseudocode

Below we include reproducible pseudocode for major pipelines.

7.1 Classical deep learning pipeline (example: TFT)

```
# Pseudocode: Training trainer for Temporal Fusion Transformer

for window_start in rolling_windows:
    train_data = get_train(window_start)
```

```
    val_data = get_val(window_start)
    model = TFT(config)
    for epoch in range(max_epochs):
        for batch in train_data:
            preds = model(batch.inputs)
            loss = loss_fn(preds, batch.targets)
            loss.backward()
            optimizer.step()
            optimizer.zero_grad()
        val_loss = evaluate(model, val_data)
        if early_stop(val_loss): break
    save_model(model, meta)
```

7.2 Quantum kernel training (kernel ridge regression)

```
# Compute quantum kernel matrix K_ij = <Phi(x_i)|Phi(x_j)>^2 via quantum circuit
for i in range(n_train):
    for j in range(i, n_train):
        K[i,j] = quantum_kernel(x_i, x_j, backend, shots)
        K[j,i] = K[i,j]

# Solve kernel ridge regression: alpha = (K + lambda I)^{-1} y
alpha = solve_linear_system(K + lambda_reg * I, y_train)
# Prediction for test point x_star: k_star = [quantum_kernel(x_star, x_i)]
y_pred = k_star @ alpha
```

7.3 VQC regression (hybrid gradient-based loop with parameter-shift)

```
# Pseudocode for VQC training using parameter-shift gradient
theta = initialize_parameters(ansatz)
for epoch in range(EPOCHS):
```

```
for batch in dataloader:
    # classical encoding
    z = classical_encoder(batch.inputs)  # reduce dims
    # compute predictions with quantum circuit
    y_hat = []
    for x in z:
        state = prepare_quantum_state(x)  # angle encoding
        apply_ansatz(state, theta)
        y_hat.append(measure_expectation(state, shots))
    loss = MSE(y_hat, batch.targets)
    # estimate gradients for theta using parameter-shift
    grad_theta = parameter_shift_gradients(theta, z, loss_fn)
    theta = optimizer.update(theta, grad_theta)
```

Note: Parameter-shift requires two evaluations per parameter per gradient estimate; for large parameter counts, consider stochastic gradient approximations (SPSA) or hybrid local updates.

8. Expected empirical behaviors and interpretability

Because quantum advantage for practical ML tasks remains unproven at scale, this study emphasizes comparative expectations rather than claims of superiority.

8.1 When classical methods excel

- **Large labeled datasets and abundant compute:** deep learning and ensemble methods scale efficiently and provide strong baselines.
- **Highly nonstationary markets:** transfer learning and continual learning approaches in classical ML currently offer practical solutions for concept drift.

8.2 Where QML may be promising

- **Small-label, high-dimension regimes:** quantum kernels may create separable representations that classical kernels do not capture (Havlíček et al., 2019).
- **Combinatorial subproblems:** quantum annealers/QAOA could provide candidate sets for discrete portfolio selection under constraints.

- **Feature augmentation:** QNNs as compact feature transformers could help in cases where classical embeddings are difficult to design.

8.3 Interpretability & model risk

Financial institutions require interpretability and model governance. Classical methods (linear models, tree ensembles) provide clearer feature importances and partial dependence plots. For QML, propose the following interpretability practices:

- **Surrogate classical models:** Fit an interpretable surrogate to QNN outputs for explanation (LIME/SHAP over QNN features).
- **Feature importance via ablation:** Measure predictive degradation when quantum features are removed.
- **Robustness checks:** Sensitivity to perturbations and stress scenarios.

9. Computational cost, scaling, and practical constraints

9.1 Classical compute

Costs measured by GPU/CPU hours; scaling to hundreds/thousands of assets requires cluster computing and engineered feature pipelines.

9.2 Quantum compute and hybrid economics

Quantum resource accounting should include:

- **QPU wall time:** provider queue + execution time.
- **Shot counts:** per circuit measurement repetitions.
- **State preparation overhead:** for amplitude encoding or complex feature maps.
- **Simulated quantum costs:** classical simulators scale exponentially and are feasible only for small qubit counts or approximate circuits.

A realistic cost-benefit analysis must compare business value uplift (e.g., additional Sharpe points) against quantum access fees and integration engineering. Near-term pilot designs should minimize QPU calls (batch kernel computations offline, caching) and rely on simulators for development.

10. Robustness, risk, and regulatory considerations

Financial models influence capital and customer outcomes; thus model risk management is essential.

- **Backtesting with transaction costs and slippage:** Ensure reported P&L is realistic.

- **Stress testing across regimes:** Evaluate performance during tails (crashes, liquidity droughts).
- **Governance & audit trails:** Log model versions, data lineage, and quantum backend parameters.
- **Regulatory transparency:** Many regulators require explainable models for trading and risk management decisions; prepare surrogate explanations for quantum components.

11. Limitations and open research problems

- **Scalability of quantum methods:** Kernel evaluations scale $O(n^2)$ unless approximations used; QPU throughput remains limited.
- **Empirical evidence:** Proofs of practical quantum advantage for forecasting remain lacking; prior literature shows theoretical potential but limited real-world demonstrations (Biamonte et al., 2017; Havlíček et al., 2019).
- **Noisy training & barren plateaus:** Algorithmic research is needed to mitigate these effects for VQCs (Cerezo et al., 2021).
- **Data non-stationarity:** QML research must adapt to concept drift and continual learning contexts common in finance.

12. Prioritized research roadmap

Short term (0–18 months)

- Build reproducible benchmark suite (public datasets + synthetic stress scenarios).
- Implement hybrid classical + quantum feature pipelines on simulators; perform offline kernel comparisons.
- Develop best practices for shot budgeting and parameter-shift adaptations.

Medium term (18–48 months)

- Pilot QML feature augmentations in shadow mode on low-latency pipelines.
- Evaluate QAOA/annealer for discrete selection subproblems at moderate scale.
- Collaborate with cloud QPU providers to benchmark real hardware performance on finance tasks.

Long term (48+ months)

- Mature fault-tolerant QML algorithms for large-scale optimization.

- Integrate quantum pipelines into live trading systems with robust governance and economic validation.

13. Conclusions

This manuscript provides a comprehensive comparative framework and experimental blueprint for evaluating classical and quantum machine learning approaches to large-scale financial forecasting. Classical ML remains the workhorse with mature toolchains and demonstrated performance at scale, whereas QML offers intriguing new representational paradigms that may confer advantage in particular problem regimes especially small-label/high-dimension tasks or combinatorial subproblems. We emphasize reproducible experimental protocols, rigorous walk-forward validation, realistic resource accounting, and model governance as prerequisites for any claims of quantum benefit. The road ahead requires interdisciplinary collaboration quantum algorithm researchers, financial engineers, and operations teams to translate theoretical promise into verified business value.

References

1. Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., & Lloyd, S. (2017). Quantum machine learning. *Nature*, 549(7671), 195–202. https://doi.org/10.1038/nature23474

2. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

3. Cerezo, M., Arrasmith, A., Babbush, R., Benjamin, S. C., Endo, S., Fujii, K., ... & Coles, P. J. (2021). Variational quantum algorithms. *Nature Reviews Physics*, 3(9), 625–644. https://doi.org/10.1038/s42254-021-00348-9

4. Fatunmbi, T. O. (2021). Integrating AI, machine learning, and quantum computing for advanced diagnostic and therapeutic strategies in modern healthcare. *International Journal of Engineering and Technology Research*, 6(1), 26–41. https://doi.org/10.34218/IJETR_06_01_002

5. Fatunmbi, T. O. (2022). Quantum-accelerated intelligence in e-commerce: The role of AI, machine learning, and blockchain for scalable, secure digital trade. *International Journal of Artificial Intelligence & Machine Learning*, 1(1), 136–151. https://doi.org/10.34218/IJAIML_01_01_014

6. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

7. Havlíček, V., Córcoles, A. D., Temme, K., et al. (2019). Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747), 209–212. https://doi.org/10.1038/s41586-019-0980-2

8. Lim, B., Arık, S. Ö., Loeff, N., & Pfister, T. (2021). Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4), 1748–1764. https://doi.org/10.1016/j.ijforecast.2020.11.011

9. McMahan, H. B., Moore, E., Ramage, D., Hampson, S., & Aguera y Arcas, B. (2017). Communication-efficient learning of deep networks from decentralized data. *Proceedings of AISTATS 2017, 54*, 1273–1282.

10. Meyer, N., Ufrecht, C., Periyasamy, M., Scherer, D. D., Plinge, A., & Mutschler, C. (2024). A survey on quantum reinforcement learning. *arXiv:2211.03464*.

11. Preskill, J. (2018). Quantum computing in the NISQ era and beyond. *Quantum*, 2, 79. https://doi.org/10.22331/q-2018-08-06-79

12. Samuel, A. J. (2022). AI and machine learning for secure data exchange in decentralized energy markets on the cloud. *World Journal of Advanced Research and Reviews*, 16(2), 1269–1287. https://doi.org/10.30574/wjarr.2022.16.2.1282

13. Samuel, A. J. (2023). Enhancing financial fraud detection with AI and cloud-based big data analytics: Security implications. *World Journal of Advanced Engineering Technology and Sciences*, 9(02), 417–434. https://doi.org/10.30574/wjaets.2023.9.2.0208

14. Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

15. Weinberg, S. J., Sanches, F., Ide, T., Kamiya, K., & Correll, R. (2022). Supply chain logistics with quantum and classical annealing algorithms. *arXiv:2205.04435*.

16. Zhang, Y., et al. (2023). Reliability research on quantum neural networks. *Electronics*, 13(8), 1514.