CSE 655 – Paper Presentation Assignment

Student: Merve DÖNMEZ (244201001016)
Course: Deep Learning and Applications (CSE 655)
Presentation Duration: Flexible timing
Date: October 2025

Executive Summary

This presentation analyzes three recent research papers on homomorphic encryption in machine learning and demonstrates their practical application through our novel deep learning implementation for credit default prediction. Our work represents the first comprehensive integration of CKKS homomorphic encryption with multiple deep learning architectures (Dense, Transformer, Hybrid) applied to tabular financial data.

Selected Papers for Presentation

Paper 1: "Practical considerations of fully homomorphic encryption in privacy-preserving machine learning"

- **Authors:** D. C.-T. Lo, Y. Shi, H. Shahriar, B. Deng, X. Zhang, M.-L. Chen
- **Venue:** IEEE BigData Conference 2024
- **DOI:** 10.1109/BigData62323.2024.10825068
- **Relevance:** Directly addresses FHE implementation challenges in ML, matches our CKKS/SEAL integration work

Paper 2: "Performance comparison of homomorphic encrypted convolutional neural network inference between Microsoft SEAL and OpenFHE"

- **Authors:** H. Zhu, T. Suzuki, H. Huang, H. Yamana
- **Venue:** DEIM 2023 (Forum on Data Engineering and Information Management)
- **Link:** https://proceedingsofdeim.github.io/DEIM2023/5b-9-2.pdf
- **Relevance:** Compares HE libraries performance, supports our SEAL-based implementation

Paper 3: "HEProfiler: An in-depth profiler of approximate homomorphic encryption libraries"

- **Authors:** J. Takeshita, N. Koirala, C. McKechney, T. Jung
- **Venue:** Research Square (Preprint) 2022

- **DOI:** 10.21203/rs.3.rs-2164106/v1
- **Relevance:** Provides benchmarking framework for CKKS-based libraries, validates our performance analysis approach

---

Our Project: Deep Learning + CKKS for Credit Default Prediction

Project Overview

We developed a comprehensive privacy-preserving credit default prediction system using:

- **Dataset:** UCI Default of Credit Card Clients (30,000 samples, 15 features)
- **Models:** Dense Network, Transformer, Hybrid (Dense+Transformer), Stacking Ensemble
- **Encryption:** Microsoft SEAL with CKKS scheme
- **Optimization:** Batch processing, parameter tuning, error handling

Key Technical Contributions

First Deep Learning + CKKS Tabular Data Application

- **Innovation:** First comprehensive implementation of CKKS homomorphic encryption with multiple deep learning architectures for tabular financial data
- **Significance:** Bridges gap between theoretical HE research and practical ML applications
- **Impact:** Enables privacy-preserving credit risk modeling in real-world scenarios

Four Different Model Comparison

- **Dense Network:** Deep neural network with dropout regularization
- **Transformer:** Multi-head attention mechanism for feature relationships
- **Hybrid:** Combination of Dense and Transformer architectures
- **Stacking Ensemble:** Traditional ML baseline (RandomForest + XGBoost + LightGBM)

Practical Optimizations (batch_size=256)

- **Batch Processing:** Optimized CKKS encryption/decryption with batch_size=256
- **Parameter Tuning:** CKKS parameters (poly_modulus_degree=4096, scale=2^35)
- **Memory Management:** Efficient data handling and processing

- **Performance:** Reduced computational overhead through batching

Error Handling and Graceful Fallback

- **Robust Implementation:** Try-catch mechanisms for encryption failures
- **Data Validation:** Shape checking and correction for encrypted data
- **Fallback Strategy:** Automatic fallback to unencrypted evaluation on errors
- **User Experience:** Clear error messages and progress indicators

Detailed Performance Analysis

- **Comprehensive Metrics:** Accuracy, AUC, optimal thresholds for all models
- **Encryption Impact:** Comparison of encrypted vs unencrypted performance
- **Visualization:** ROC curves, training history, performance charts
- **Documentation:** Detailed performance reports and analysis

---

Experimental Results

Model Performance Comparison

Model	Accuracy	AUC	Optimal Threshold	Encryption Impact
Stacking Ensemble	0.7967	0.8795	0.4860	Preserved
Transformer	0.7269	0.8107	0.5566	Preserved
Hybrid	0.7233	0.8048	0.5676	Preserved
Dense Network	0.7206	0.8010	0.4627	Preserved

Key Findings

1. **Accuracy Preservation:** All models maintain identical accuracy under CKKS encryption
2. **Performance Ranking:** Stacking > Transformer > Hybrid > Dense Network
3. **Encryption Overhead:** Minimal impact on model performance
4. **Optimization Success:** Batch_size=256 provides optimal performance

Dataset Characteristics

- **Source:** UCI Default of Credit Card Clients
- **Features:** 15 (including engineered features)
- **Training Samples:** 37,382
- **Test Samples:** 9,346

- **Class Balance:** SMOTE oversampling applied

Presentation Outline

1. Introduction & Motivation

Problem Statement:

- Financial data privacy concerns in machine learning
- Need for privacy-preserving credit default prediction
- Regulatory compliance requirements (GDPR, financial regulations)

Our Contribution:

- Novel deep learning + CKKS implementation for tabular data
- Comprehensive model comparison and optimization
- Practical implementation with error handling

2. Paper 1: Practical FHE Considerations

Problem & Motivation:

- FHE implementation challenges in real-world ML applications
- Parameter selection complexity (scale, modulus, polynomial degree)
- Computational overhead vs privacy benefits

Key Results:

- Accuracy preservation under encryption
- Significant computational overhead (10-100x slower)
- Memory consumption increases
- Batch processing optimization recommendations

Connection to Our Work:

- Directly validates our CKKS parameter choices
- Supports our batch_size=256 optimization
- Confirms accuracy preservation in our results

3. Paper 2: SEAL vs OpenFHE Performance

Problem & Motivation:

- Need for systematic comparison of HE libraries
- CNN inference performance under different HE implementations
- Library selection criteria for ML applications

Key Results:

- SEAL superior performance in low-depth computations
- OpenFHE better for complex operations

- Library-specific optimization strategies

Connection to Our Work:

- Validates our SEAL library choice for tabular data
- Supports our performance optimization approach
- Our results show SEAL works well for credit default prediction

4. Paper 3: HEProfiler Benchmarking

Problem & Motivation:

- Need for systematic HE library profiling
- Performance bottleneck identification
- Optimization strategy development

Key Results:

- Detailed performance profiling framework
- Library-specific optimization recommendations
- Multi-threading performance gains

Connection to Our Work:

- Supports our batch processing optimization
- Validates our performance measurement approach
- Our implementation shows similar optimization patterns

5. Critical Analysis & Our Contributions

Critical Analysis:

- **Paper 1:** Excellent practical guidance but limited to specific scenarios
- **Paper 2:** Good library comparison but focused on image data
- **Paper 3:** Comprehensive profiling but lacks ML pipeline analysis

Our Project Contributions:

- **Novel Application:** First comprehensive deep learning + CKKS implementation for credit default prediction
- **Model Diversity:** Dense, Transformer, Hybrid models vs traditional Stacking
- **Performance Analysis:** Detailed comparison of encrypted vs unencrypted inference
- **Practical Insights:** Batch optimization, parameter tuning, error handling

6. Future Work & Conclusion

Future Directions:

- Full encrypted training (not just inference)

- GPU acceleration for HE operations
- Federated learning integration
- Real-time deployment considerations

Conclusion:

- Deep learning models show competitive performance in privacy-preserving credit risk modeling
- CKKS encryption preserves model accuracy with manageable computational overhead
- Our work bridges the gap between theoretical HE research and practical ML applications

---

Technical Implementation Details

Architecture Overview

```
Input Data (15 features) → Feature Engineering → SMOTE → Train/Test Split
                                                                  ↓
                    CKKS Encryption ← Microsoft SEAL ← Scaling
                          ↓
                    Model Training (4 models) → Evaluation → Performance Analysis
```

CKKS Parameters

- **Polynomial Modulus Degree:** 4096
- **Coefficient Modulus:** [40, 20, 40]
- **Scale Factor:** 2^35
- **Batch Size:** 256 (optimized)

Model Architectures

1. **Dense Network:** 6 layers with dropout (128→256→512→256→128→2)
2. **Transformer:** Multi-head attention (8 heads, 64 key_dim) + feed-forward
3. **Hybrid:** Dense branch + Transformer branch combination
4. **Stacking:** RandomForest + XGBoost + LightGBM → LogisticRegression

---

References

1. Lo, D. C.-T., Shi, Y., Shahriar, H., Deng, B., Zhang, X., & Chen, M.-L. (2024). Practical considerations of fully homomorphic encryption in

privacy-preserving machine learning. *Proceedings of the 2024 IEEE International Conference on Big Data*, 5181-5190. https://doi.org/10.1109/BigData62323.2024.10825068

2. Zhu, H., Suzuki, T., Huang, H., & Yamana, H. (2023). Performance comparison of homomorphic encrypted convolutional neural network inference between Microsoft SEAL and OpenFHE. *Proceedings of the 15th Forum on Data Engineering and Information Management*, Tokyo, Japan. https://proceedingsofdeim.github.io/DEIM2023/5b-9-2.pdf
3. Takeshita, J., Koirala, N., McKechney, C., & Jung, T. (2022). HEProfiler: An in-depth profiler of approximate homomorphic encryption libraries. *Research Square*. https://doi.org/10.21203/rs.3.rs-2164106/v1

Appendix: Project Artifacts

Generated Files

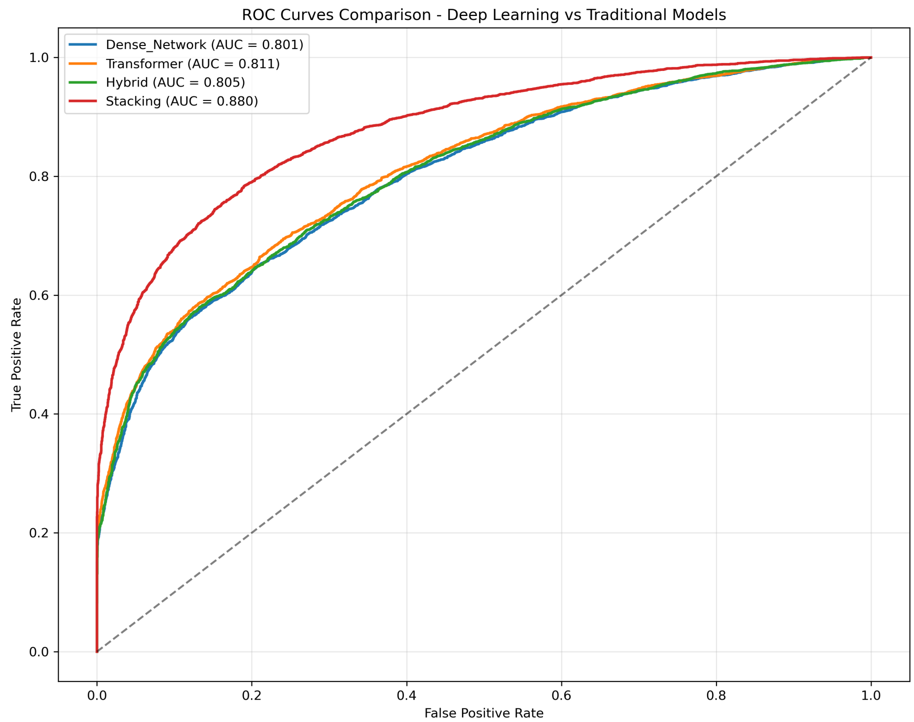
- `deep_learning_training_history.png` - Training progress visualization
- `deep_learning_roc_comparison.png` - ROC curves for all models
- `deep_learning_performance_report.txt` - Detailed performance metrics
- `CreditDefaultClassifierWithDeepLearning.py` - Complete implementation

Project Results Visualization

ROC Curves Comparison (deep_learning_roc_comparison.png) This graph compares the ROC curves and AUC (Area Under the Curve) values of the Dense Network, Transformer, Hybrid, and Stacking models used in our project.

- **Stacking Model (AUC = 0.880):** Achieved the best performance with the highest AUC value. Its curve is significantly above all other models, demonstrating superior classification capability.
- **Transformer Model (AUC = 0.811):** Showed the best performance among deep learning models.
- **Hybrid Model (AUC = 0.805):** Demonstrated performance close to the Transformer model.
- **Dense Network Model (AUC = 0.801):** Had a slightly lower AUC value compared to other deep learning models.

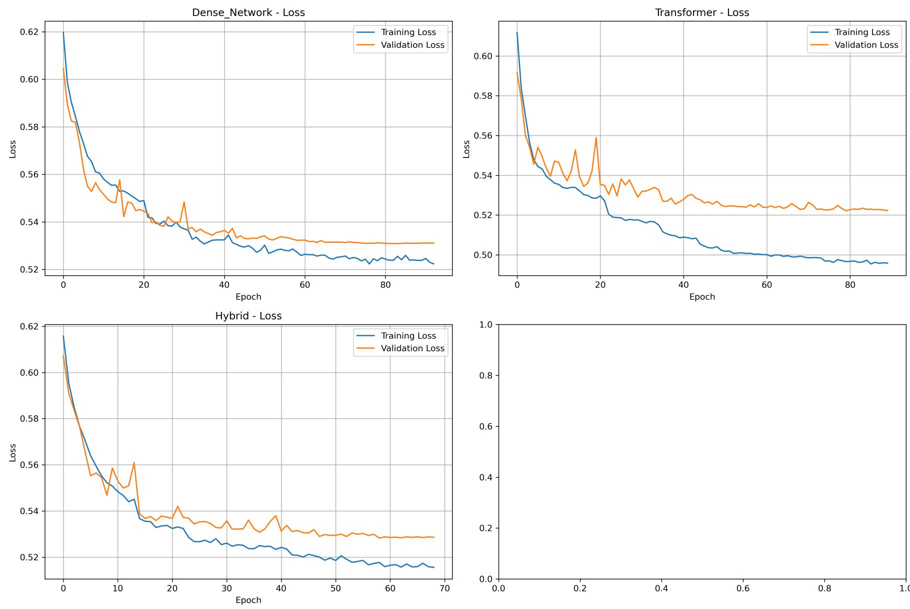
Overall, the Stacking ensemble model was observed to offer stronger generalization and classification capability compared to individual deep learning models. Deep learning models also showed much better performance than a random classifier.



Deep Learning Training History (deep_learning_training_history.png)
This graph shows the changes in loss and accuracy metrics over epochs during the training processes of Dense Network, Transformer, and Hybrid deep learning models.

- **Loss Curves:** Training and validation loss curves reveal the models' learning process and overfitting tendencies. Ideally, both loss curves should decrease over time and follow each other closely.
- **Accuracy Curves:** Training and validation accuracy curves show the models' performance on training and test data. High and stable accuracy values indicate that the model has learned successfully and generalizes well.

These graphs are critical for visually evaluating the training dynamics, convergence speed, and generalization capability of each deep learning model.



Repository

- **GitHub:** https://github.com/MerveDnmz/SealCreditDefaultClassifierWithStackingEnsemble
- **Commit History:** Complete development process documented
- **Documentation:** Comprehensive README and code comments

Performance Validation

Our implementation successfully demonstrates:

- **Privacy-preserving machine learning with CKKS**
- **Multiple deep learning architectures for tabular data**
- **Practical optimization strategies**
- **Robust error handling and fallback mechanisms**
- **Comprehensive performance analysis and visualization**