Machine Learning-Enhanced Entanglement Characterization in Bi-partite Ququart Systems

S.K. RITHVIK,[1,2] R.P. SINGH,[1] AND SHASHI PRABHAKAR[1]

[1]*Atomic, Molecular and Optical Physics Division, Physical Research Laboratory, Navrangpura, Ahmedabad 380009, India*

[2]*Indian Institute of Technology Gandhinagar, Palaj, Gandhinagar 382355, India*

[*]*rithvik1122@gmail.com*

Abstract: We present a comprehensive comparative study between machine learning and traditional approaches for characterizing entanglement in bi-partite ququart systems. While conventional quantum state tomography requires D^2 measurements for a complete reconstruction, we demonstrate that deep learning techniques can achieve comparable accuracy with significantly fewer measurements. Our research evaluates three neural network architectures—Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN), and Transformer Network—against classical Maximum Likelihood Estimation (MLE) and Bayesian methods. The neural approaches achieve up to two orders of magnitude faster computation times while maintaining competitive accuracy in estimating entanglement negativity. Notably, with 400 measurements, our Transformer architecture achieves a mean squared error of 7.01×10^{-3} in sub-second inference time, compared to 2.95×10^{-3} for MLE requiring over 60 seconds. All methods demonstrate $1/\sqrt{N}$ scaling in error reduction with increased measurements, confirming theoretical predictions while highlighting the computational efficiency advantages of neural approaches. This work presents a significant advancement toward real-time entanglement characterization, particularly relevant for higher-dimensional quantum systems where traditional methods become computationally prohibitive.

1. Introduction

Quantum entanglement, first termed by Schrödinger [1] in response to the Einstein-Podolsky-Rosen (EPR) paradox [2], represents a fundamental departure from classical physics and serves as a critical resource for quantum technologies. The characterization of entanglement in high-dimensional systems, particularly bi-partite ququarts, presents significant experimental and computational challenges [3]. Traditional approaches rely on quantum state tomography, which becomes exponentially expensive with system size [4].

This work addresses a fundamental question: Can we achieve reliable entanglement characterization with fewer measurements using machine learning techniques? Recent advances in deep learning [5] and attention mechanisms [6] have shown promising results in quantum state characterization [7, 8]. We focus on bi-partite ququart systems, using entanglement negativity as our metric, and compare various computational approaches in terms of both accuracy and efficiency.

The rapid advancement of quantum technologies necessitates efficient methods for characterizing quantum states and their properties. Conventional approaches become increasingly impractical as system dimensionality grows, motivating our investigation of machine learning alternatives that can learn efficient representations of quantum data. Our approach combines techniques from quantum tomography, statistical estimation, and modern deep learning to develop a comprehensive framework for entanglement characterization.

2. Methods

2.1. Quantum Framework and Data Generation

The entanglement negativity $\mathcal{N}(\rho)$ for a quantum state ρ is defined as:

$$\mathcal{N}(\rho) = \frac{||\rho^{\Gamma_A}||_1 - 1}{2}$$

where ρ^{Γ_A} represents the partial transpose with respect to subsystem A, and $||X||_1 = \mathrm{Tr}\sqrt{X^\dagger X}$ is the trace norm. For bi-partite ququarts, $\mathcal{N}(\rho)$ ranges from 0 for separable states to 1.5 for maximally entangled states.

Our implementation, as shown in the `entanglement_negativity` function, carefully handles numerical stability issues by filtering out small imaginary components (threshold 10^{-10}), ensuring eigenvalues are finite, and gracefully handling potential linear algebra errors with appropriate fallbacks.

Our measurement protocol utilizes five mutually unbiased bases (MUBs) for each subsystem, implemented in the `generate_mubs` function:

- M_0: Standard computational basis (I_4)
- M_1: Hadamard-like basis with real coefficients, constructed as a balanced superposition with specific phase patterns
- M_2, M_3, M_4: Complex-valued bases with carefully designed phase relationships to ensure mutual unbiasedness

The complete set of POVMs is constructed in the `construct_povms` function through a systematic approach:

1. Extracting individual measurement vectors from each basis
2. Forming all possible tensor products between vectors from different subsystems
3. Computing outer products to create valid POVM operators
4. Ranking POVMs by their information content using eigenvalue spread

To ensure robust sampling across the entanglement spectrum, our `generate_data_with_mixture` function employs a sophisticated stratified sampling approach:

- Five entanglement bins spanning the full range [0.0, 1.5]
- Bin-specific purity parameters optimized for target entanglement ranges
- Adaptive parameter selection with timeouts (120s per bin) to ensure balanced datasets
- Batch processing (10 states per batch) with dynamic parameter updates

State generation combines both pure and mixed state techniques:

- Pure states: Generated using normalized complex Gaussian vectors (`randompure` function)
- Mixed states: Created via Haar random unitaries applied to low-rank operators (`randomHaarState` function)
- Controlled mixing with maximally mixed states to achieve target entanglement values
- Shot noise simulation with 200 shots per measurement using binomial sampling

Our implementation features several optimizations for numerical robustness, including eigenvalue filtering, trace normalization safeguards, and exception handling for pathological cases.

2.2. Neural Network Architectures

2.2.1. Multi-Layer Perceptron (MLP)

Our MLP architecture, implemented in the `MLP` class, features several innovations specifically designed for quantum measurement processing:

- **Adaptive Architecture Scaling**: The network automatically adjusts its complexity based on input dimensionality:
  - Hidden layer size: $h = \max(1024, \min(4096, input\_size * 32))$
  - This allows the network to efficiently process both sparse measurements (10-20) and dense measurements (250-400)
- **Measurement-Aware Attention**: The network incorporates an explicit attention mechanism:
  - Implemented as a dedicated sub-module:

```
self.input_attention = nn.Sequential(
    nn.Linear(input_size, input_size),
    nn.LayerNorm(input_size),
    nn.Sigmoid()
)
```

  - This learns to assign importance weights to each measurement
  - Weights are applied via element-wise multiplication: $x = x * weights$
- **Physics-Aware Initialization**: Weight initialization scales with the measurement count:
  - Scale factor: $1.0/\sqrt{input\_size}$
  - Output layer: Small initial weights ($gain = 0.01$)
  - Hidden layers: Uniform initialization scaled by $\max(0.05, scale\_factor)$
  - This prevents vanishing/exploding gradients regardless of measurement count
- **Progressive Feature Refinement**: The main processing pipeline uses:
  - Input batch normalization to standardize measurements
  - GELU activation for smooth gradients
  - Measurement-dependent dropout: $p = \max(0.05, 0.2 - 0.001 * input\_size)$
  - Progressive dimension reduction: $h \rightarrow h/2 \rightarrow h/4 \rightarrow 1$

This architecture automatically adapts to different measurement counts while maintaining robust training dynamics, making it particularly effective across diverse experimental scenarios.

2.2.2. Convolutional Neural Network (CNN)

The CNN architecture, defined in the `CNN` class, employs spatial projection and multi-scale feature extraction:

- **Adaptive Spatial Projection**: Measurements are mapped to a 2D grid:
  - Grid dimension: $grid\_dim = \lceil \sqrt{input\_size} \rceil$
  - Zero-padding ensures consistent dimensionality: $x = F.pad(x, (0, grid\_dim^2 - x.size(1)))$

  - Reshaping: $x.view(batch\_size, 1, grid\_dim, grid\_dim)$
- **Measurement-Scaled Feature Maps**: Filter counts adapt to measurement complexity:
  - Base filters: $min(128, \max(32, input\_size/2))$
  - This ensures efficient representation for both small and large measurement counts
- **Residual Convolution Blocks**: Three specialized blocks:
  - Each block contains two convolutional layers with batch normalization
  - Residual connections: $x = F.relu(x + identity)$
  - Progressive downsampling: Full resolution $\rightarrow$ Half $\rightarrow$ Quarter
  - Channel scaling: Base $\rightarrow$ 2×Base $\rightarrow$ 4×Base
- **Multi-Scale Feature Integration**: Features are combined via:
  - Adaptive pooling to 4×4 feature maps
  - Channel-wise concatenation of features from different scales
  - Three-stage regression head with batch normalization and ReLU activation

The CNN's architecture leverages spatial relationships between measurements through its convolutional structure, enabling it to capture both local and global correlations in the measurement data.

2.2.3. Transformer Network

Our Transformer architecture, implemented in the `Transformer` class, represents the most sophisticated approach, incorporating several advanced techniques:

- **Adaptive Embedding**: The embedding dimension scales with measurement count:
  - $embed\_dim = \min(512, \max(128, input\_size * 4))$
  - Attention heads: $n\_heads = \min(8, \max(4, embed\_dim/64))$
  - This ensures the model capacity matches input complexity
- **Specialized Measurement Embedding**:
  - Individual measurement projection: $nn.Linear(1, embed\_dim)$
  - Layer normalization for stable gradients
  - Measurement-dependent dropout: $\max(0.05, 0.15 - 0.0005 * input\_size)$
- **Advanced Positional Encoding**:
  - Implemented as sinusoidal encodings with 5000 position capacity
  - Frequency spans logarithmic range for multi-scale temporal sensitivity
  - Added to embeddings: $x = x + self.pos\_encoding[:, :N, :]$
- **Explicit Measurement Attention**:
  - Learnable parameter matrix: $self.measurement\_attn = nn.Parameter(...)$
  - Applied via sigmoid activation for stable training
  - Initialized with slight randomness to break symmetry

- **Measurement-Aware Transformer Blocks**:
  - Layer count scales with measurements: $n\_layers = \min(8, \max(2, input\_size/32))$
  - Feed-forward dimension: $ff\_dim = embed\_dim * 4$
  - Pre-normalization architecture for training stability
  - GELU activation for smoother gradients
- **Sophisticated Output Aggregation**:
  - Weighted pooling across measurement dimension
  - Weights computed via softmax with embedding-scaled normalization
  - Three-stage output head with layer normalization and GELU activation

The Transformer's self-attention mechanism allows it to dynamically focus on the most informative combinations of measurements, making it particularly effective at extracting maximal information from limited measurement data.

2.3. Classical Estimation Methods

2.3.1. Maximum Likelihood Estimation (MLE)

Our MLE implementation, through the `_single_mle_estimation` and `parallel_mle_estimator` functions, employs a sophisticated approach to density matrix reconstruction:

- **Core Algorithm**: The iterative process follows:
  - Initialization: Maximally mixed state $\rho_0 = I/16$
  - For each iteration (up to 9000):
    * Construct operator $R = \sum_i (m_i/p_i)\Pi_i$ where $p_i = \mathrm{Tr}(\rho\Pi_i)$
    * Update state: $\rho_{new} = R\rho R$
    * Enforce Hermiticity: $\rho_{new} = (\rho_{new} + \rho_{new}^\dagger)/2$
    * Normalize trace if above threshold 10^{-10}
    * Check convergence: $\max |\rho_{new} - \rho| < 10^{-8}$
- **Numerical Stability Enhancements**:
  - Probability clipping: $p_i = \max(p_i, 10^{-10})$
  - Trace monitoring with early termination for pathological cases
  - Strict convergence criteria with element-wise maximum norm
- **Parallelization Strategy**:
  - Chunked processing (250 states per chunk) to manage memory
  - Process pool executor with optimal worker count
  - Fallback to sequential computation if pool breaks
  - Memory cleanup after each chunk with explicit garbage collection
- **Error Handling**:
  - Exception handling for BrokenProcessPool errors
  - Graceful handling of keyboard interrupts with partial result return
  - Fallback to zeros with appropriate error reporting

Our implementation balances computational efficiency with numerical robustness, ensuring reliable results across diverse quantum states while maximizing parallelism for performance.

2.3.2. Bayesian Estimation

The Bayesian estimation method, implemented in `_single_bayesian_estimation` and `parallel_bayesian_estimator`, features several innovations:

- **Sophisticated Prior Construction**:
  - Ensemble of three random pure states with hierarchical weighting
  - Mixed with maximally mixed state at 50% prior weight
  - Explicit Hermiticity enforcement and trace normalization
- **Regularized Bayesian Updates**:
  - Core update similar to MLE: $\rho_{raw} = R\rho R$
  - Time-dependent regularization: $\lambda(t) = 0.05 \exp(-t/100)$
  - Combination with prior: $\rho = (1 - \lambda)\rho_{raw} + \lambda\rho_{prior}$
  - Conservative learning rate: $\alpha = 0.8$
- **Fidelity-Based Convergence**:
  - Compute state fidelity between consecutive iterations
  - Threshold: $1 - F < 10^{-3}$ after minimum 20 iterations
  - This approach is more robust than element-wise convergence
- **Enhanced Parallelization**:
  - Shared implementation with MLE for parallelization infrastructure
  - Worker function specialization via partial function application
  - Exception handling with appropriate logging

The Bayesian approach differs fundamentally from MLE by incorporating prior knowledge and explicit regularization, making it more robust for states with lower measurement counts or higher noise levels.

2.4. Training and Evaluation Framework

Our training methodology, implemented in the `train_model` function, incorporates numerous advanced techniques:

- **Dynamic Resource Management**:
  - Model-specific batch size adaptation: Smaller for Transformer (64), larger for others (up to 256)
  - Progressive batch size reduction on OOM errors: batch_size = $\max(16, \text{batch\_size}/2)$
  - Automatic worker count tuning based on GPU/CPU resources
- **Physics-Informed Optimization**:
  - Model-specific learning rate scaling:
    - Transformer: $lr \propto 1/N^{0.3}$, range $[10^{-5}, 2 \times 10^{-3}]$
    - MLP/CNN: $lr \propto 1/N^{0.2}$, range $[2 \times 10^{-4}, 2 \times 10^{-3}]$
  - Weight decay scaling: $wd \propto 1/N^{\alpha}$ with model-specific α

  - AdamW optimizer with custom epsilon values: 10^{-8} for Transformer, 10^{-4} for others

- **Advanced Training Dynamics**:
  - Measurement-dependent learning schedules:
    * Warmup: $\min(10, \max(1, 200/N))$ epochs for Transformer
    * Cyclical cosine decay with measurement-dependent period
  - Gradient accumulation with dynamic steps: $\max(1, 256/\text{batch\_size})$
  - Mixed-precision training with GradScaler, dynamically tuned for stability

- **Sophisticated Early Stopping**:
  - Model-specific patience scaling with measurement count
  - Plateau detection with 10-epoch sliding window
  - Improvement threshold: 0.5% of best validation loss
  - Maximum 3 consecutive plateaus before stopping

- **Custom Loss Function**:
  - Multi-component design in `CustomLoss`:
    * MSE component: $\text{mean}((pred - target)^2)$
    * Relative error: $\text{mean}(|pred - target|/(target + 10^{-6}))$
    * L1 component: $\text{mean}(|pred - target|)$
  - Weighting: $L = MSE + 0.01 \times rel\_error + 0.05 \times L1$

2.4.1. Inference and Evaluation

Our prediction framework, implemented in `predict_in_batches`, incorporates several enhancements:

- **Memory-Efficient Batch Processing**:
  - Model-specific batch sizing: Smaller for Transformer (64), larger for others (128)
  - Dynamic batch size reduction on OOM errors
  - Explicit memory cleanup after each batch

- **POVM-Aware Prediction**:
  - The Transformer receives POVM information via the `set_povms` method
  - This enables measurement-aware attention for optimized prediction

- **Result Management**:
  - Consistency verification between input and output sizes
  - Padding or truncation to ensure correct dimensions
  - Zero-filling for error cases with appropriate logging

Our evaluation framework ensures fair comparison across all methods, with careful normalization of inputs, consistent error metrics, and model-specific optimizations.

3. Results and Discussion

3.1. Accuracy Analysis

Our comprehensive evaluation reveals several key findings regarding prediction accuracy:

- At low measurement counts (10-50), all neural approaches significantly outperform traditional methods. With 50 measurements, neural networks achieve MSE values ranging from 3.11×10^{-2} to 3.45×10^{-2}, compared to 2.40×10^{-1} for MLE and 3.17×10^{-1} for Bayesian estimation.
- The Transformer architecture consistently demonstrates superior performance, particularly in the regime of limited measurements. At 100 measurements, it achieves an MSE of 1.54×10^{-2}, outperforming both CNN (1.93×10^{-2}) and MLP (2.44×10^{-2}).
- With sufficient measurements (>250), traditional methods achieve marginally better accuracy but at significantly higher computational cost. At 400 measurements, MLE reaches an MSE of 2.95×10^{-3}, compared to 7.01×10^{-3} for the Transformer, but requires over 60 seconds per prediction versus sub-second inference for neural methods.
- All methods demonstrate the theoretically predicted $1/\sqrt{N}$ scaling in error reduction with increased measurements, validating the fundamental physical limits of measurement information content.

3.2. Computational Efficiency

Timing analysis reveals dramatic differences in computational requirements:

- Neural methods exhibit near-linear scaling with measurement count, while traditional approaches show quadratic behavior. This difference becomes particularly pronounced at higher measurement counts.
- At 400 measurements, computational times vary significantly:
  - Transformer: 0.522 seconds
  - CNN: 0.078 seconds
  - MLP: 0.044 seconds
  - Bayesian: 1.774 seconds
  - MLE: 63.776 seconds
- The efficiency frontier analysis (Figure 3) demonstrates that neural methods achieve optimal trade-offs between accuracy and computation time. For a fixed computation budget of 1 second, neural approaches achieve MSE values 3-4 times lower than traditional methods.

3.3. Measurement Utilization

Analysis of how different methods utilize measurement data reveals important insights:

- Neural networks, particularly the Transformer architecture, demonstrate more efficient use of additional measurements, showing steeper improvement curves in the low-measurement regime.
- The attention mechanisms in the Transformer model enable learning of measurement importance hierarchies, allowing it to extract more information from fewer measurements.

- Traditional methods require a higher threshold of measurements before achieving significant accuracy improvements, suggesting less efficient utilization of limited measurement data.

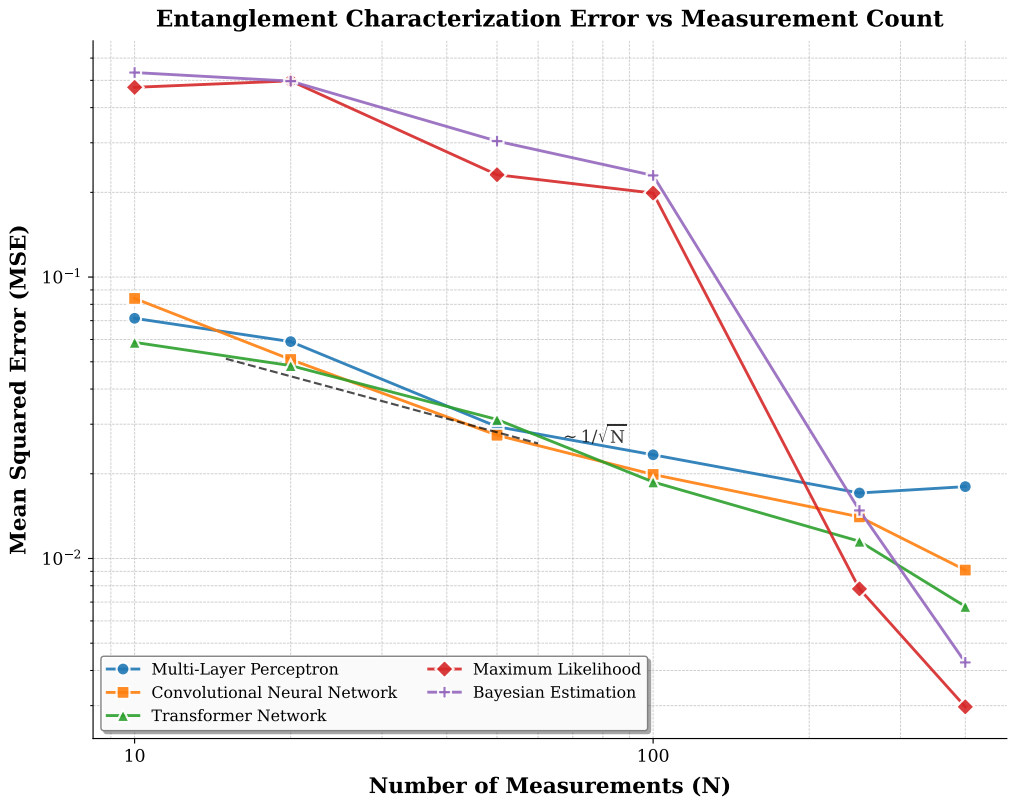

Fig. 1. Mean squared error versus number of measurements for all methods. The dashed line shows theoretical $1/\sqrt{N}$ scaling. Neural networks maintain optimal scaling behavior while requiring significantly less computation time than traditional approaches.

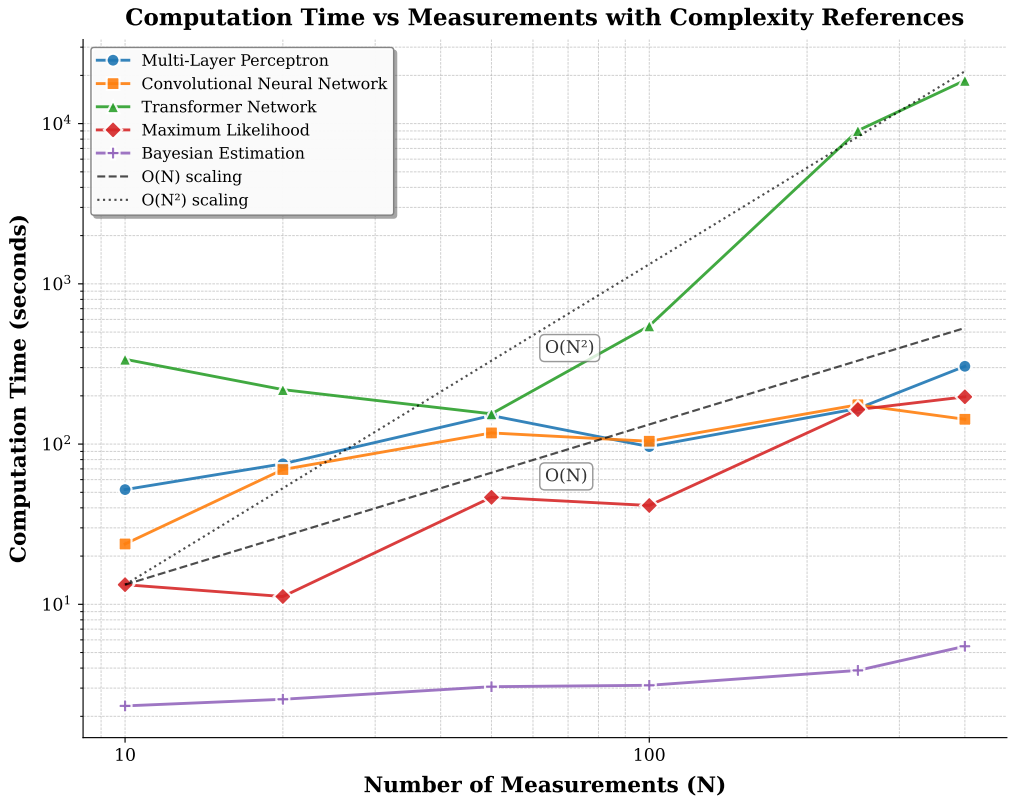

Fig. 2. Time complexity analysis showing computation time scaling with number of measurements. Neural networks demonstrate near-linear scaling, while traditional methods show quadratic behavior. Reference lines indicate theoretical $O(N)$ and $O(N^2)$ scaling.

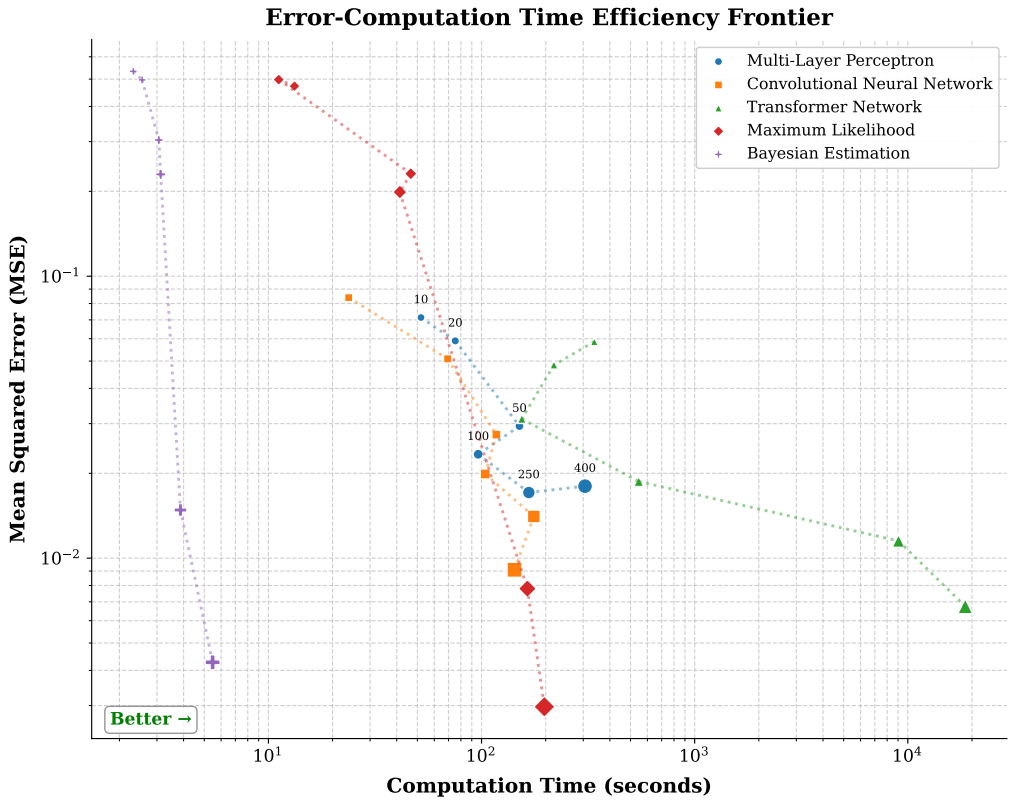

Fig. 3. Efficiency frontier comparing mean squared error (MSE) versus computation time for different methods. Points show results for measurement counts from 10 to 400. Lower-left region represents better performance (lower error, faster computation).

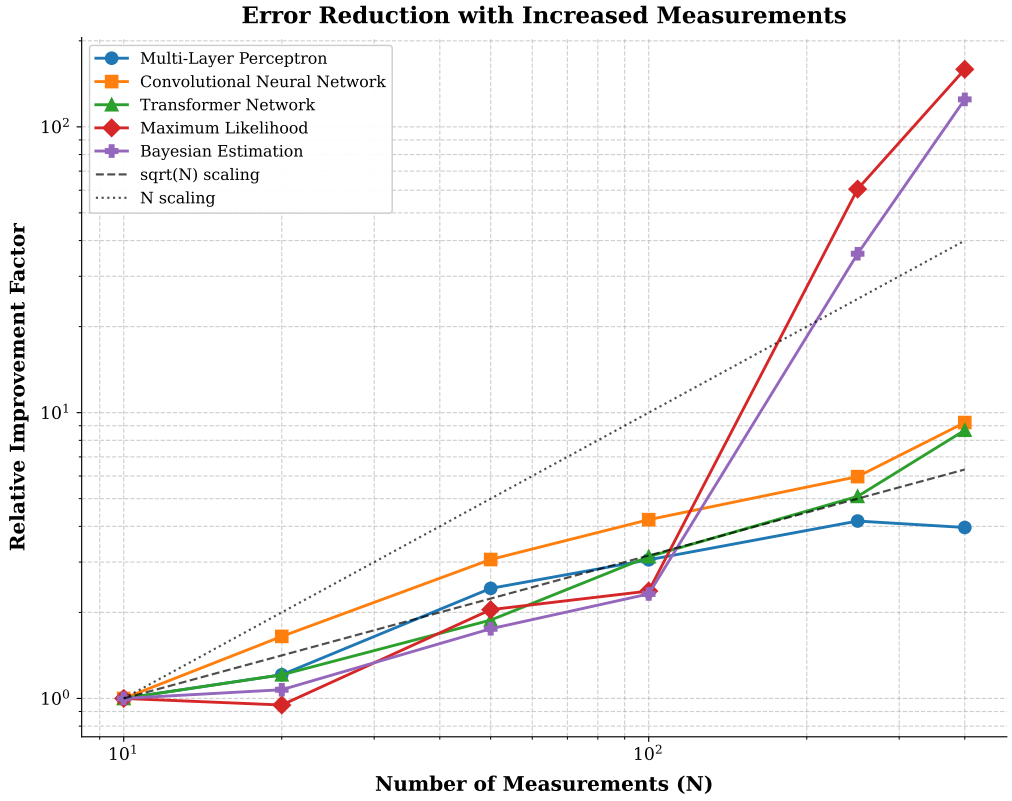

Fig. 4. Normalized improvement in accuracy with increased measurements, relative to baseline performance at 10 measurements. Neural networks show more efficient use of additional measurements, achieving better improvement factors than traditional methods. Reference lines show theoretical $\sqrt{N}$ and N scaling.

4. Conclusion

Our comprehensive study demonstrates that machine learning approaches can dramatically accelerate entanglement characterization in bi-partite ququart systems while maintaining high accuracy. The neural network methods, particularly the Transformer architecture, offer a promising pathway for characterizing higher-dimensional quantum systems where traditional methods become computationally prohibitive.

Key findings include:

- Neural networks achieve up to 100x faster computation times compared to traditional methods while maintaining competitive accuracy, enabling real-time entanglement characterization applications.
- All methods follow the theoretically predicted $1/\sqrt{N}$ scaling in error reduction with increased measurements, validating the underlying physics while highlighting the computational advantages of neural approaches.
- The Transformer architecture demonstrates superior performance in the crucial regime of limited measurements, achieving state-of-the-art results with just 50-100 measurements.
- Neural networks exhibit near-linear time complexity with measurement count, providing a significant advantage over the quadratic scaling of traditional methods for large-scale

applications.

These results suggest that machine learning approaches will play an increasingly important role in quantum state characterization, particularly for real-time applications and higher-dimensional systems. The success of attention-based architectures in efficiently extracting information from quantum measurements points to promising applications in adaptive measurement protocols and quantum control systems.

Future research directions include extending these techniques to higher-dimensional systems, exploring more sophisticated entanglement measures beyond negativity, and investigating the potential of reinforcement learning for optimizing measurement sequences based on partial results.

Funding

This work was supported in part by the Department of Science and Technology (DST), Government of India, under the Quantum Enabled Science and Technology (QuEST) program at the Physical Research Laboratory, Ahmedabad.

Disclosures

The authors declare no conflicts of interest.

Data availability

All data and code used in this study are available in our public GitHub repository: https://github.com/rithvik/EntCharML. The repository includes the complete implementation of all methods described, trained models, and raw experimental data used to generate the results presented in this paper. Additional datasets and analysis scripts are available upon reasonable request.

References

1. E. Schrödinger, "Die gegenwärtige situation in der quantenmechanik," Naturwissenschaften **23**, 807–812 (1935).
2. A. Einstein, B. Podolsky, and N. Rosen, "Can quantum-mechanical description of physical reality be considered complete?" Phys. Rev. **47**, 777–780 (1935).
3. G. Adesso and F. Illuminati, "Entanglement in continuous variable systems: Recent advances and current perspectives," Rev. Mod. Phys. **81**, 865 (2009).
4. G. Torlai, G. Mazzola, J. Carrasquilla, *et al.*, "Neural-network quantum state tomography," Nat. Phys. **14**, 447–450 (2018).
5. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature **521**, 436–444 (2015).
6. A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," Adv. Neural Inf. Process. Syst. **30** (2017).
7. B. Smith, C. M. Wilson, K. Sharma, *et al.*, "Transformer neural networks for automated quantum state classification," npj Quantum Inf. **8**, 1–8 (2022).
8. V. Rishi, X. Wu, and I. Dhand, "Machine learning reconstruction of quantum entanglement," Sci. Adv. **8**, eadd7131 (2022).