Neural Shadows: A Machine Learning Approach to Classifying Entanglement Phases

Syed Emad Uddin Shubha, Joseph Masters
Machine Learning (CSC 7333)
Department of Computer Science, Louisiana State University

November 27, 2025

Abstract

This project develops and evaluates a machine learning system for quantum phase classification, tasked with distinguishing between *entangled* and *separable* phases in noisy multi-qubit states. Leveraging the efficient *classical shadow* formalism, the system serves as a practical demonstration that machine learning can accurately characterize quantum systems using limited measurement data, avoiding the prohibitive costs of full state tomography. We implement and benchmark a suite of classification algorithms, including a linear baseline (Logistic Regression), ensemble methods (Random Forest), a feedforward neural network (FNN), and a Support Vector Machine (SVM) trained on classical-shadow features. To ensure robust performance, we perform systematic hyperparameter tuning and Monte Carlo stability analysis. The comparative study investigates how model capacity and feature representations affect entanglement detection in data-scarce regimes. Our findings show that standard machine learning models trained on classical shadows provide a resource-efficient and scalable pathway for characterizing the properties of near-term quantum devices.

Contents

1 Introduction & Motivation

Quantum entanglement is a central resource for quantum computing, communication, and metrology. However, certifying whether an experimental state is entangled typically requires some form of quantum state tomography, whose number of measurement settings and post–processing cost grow exponentially with the number of qubits. Even for few–qubit systems, repeated calibration runs with many shots per setting can be prohibitively time–consuming on noisy near–term devices.

This project investigates a more resource–efficient route based on *classical shadows.* Instead of reconstructing the full quantum state, we perform randomized local Pauli measurements and use the inverted measurement channel to obtain a compact classical representation of each state. These classical-shadow features are then fed into standard machine learning (ML) models, turning entanglement detection into a supervised classification problem on tabular data.

We focus deliberately on a *low-resource* regime in which both the number of states N and the number of measurement snapshots per state T are small. This regime mimics realistic constraints of near-term quantum hardware and stresses the sample efficiency and robustness of the ML system.

Research Question and Contributions

The central research question of this study is:

> *Given only classical-shadow data from a noisy two-qubit Werner state, how do different machine learning architectures (Logistic Regression, Random Forest, Feedforward Neural Network, and RBF SVM) compare in classifying entangled versus separable states in low-data, low-shot regimes?*

Concretely, we:

- design a physics-informed data pipeline that generates noisy Bell states, computes analytical concurrence, and produces classical-shadow features via joint Pauli measurements;
- implement and tune several ML models on these features, comparing linear and non-linear decision rules under the same measurement budget;
- perform a 100-trial Monte Carlo stability analysis to quantify how sensitive each model is to finite-sample fluctuations in both state preparation and measurements.

2 Background & Related Work

2.1 Werner States & Concurrence

We focus on the family of 2-qubit Werner states, which are mixtures of a maximally entangled Bell state and white noise. Let $|\Phi^+\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$. The Werner state is parameterized by a noise level $p \in [0, 1]$:

$$\rho(p) = (1-p)\left|\Phi^+\right\rangle\left\langle\Phi^+\right| + \frac{p}{4}I_4 \tag{1}$$

The entanglement of these states is analytically known via the concurrence $C(\rho)$. For two-qubit Werner states, the concurrence is given analytically by

$$C(\rho(p)) = \max\left\{0, \frac{3(1-p)}{2} - \frac{1}{2}\right\} \tag{2}$$

so that $C(\rho(p)) = 0$ precisely when $p \geq 2/3$. This closed form allows us to generate "ground truth" entanglement labels without numerical optimization.

2.2 Classical Shadows

Classical shadows, proposed by Huang et al. [1], provide a method to estimate many properties of a state ρ by averaging over randomized Pauli (or Clifford) measurements. For single-qubit Pauli measurements, the inverted-channel estimate for one snapshot is

$$\hat{\rho}_i = 3\left|s_i\right\rangle\left\langle s_i\right| - I_2 \tag{3}$$

where $|s_i\rangle$ is the measured eigenstate in a random Pauli basis. For an N-qubit system, the tensor product estimator from one snapshot is $\hat{\rho} = \bigotimes_{i=1}^{N} \hat{\rho}_i$ and the final estimate is obtained by averaging over T snapshots,

$$\hat{\rho}_{\text{avg}} = \frac{1}{T}\sum_{t=1}^{T} \hat{\rho}^{(t)} \tag{4}$$

In this project we restrict attention to $N = 2$ qubits and randomized single-qubit Pauli measurements. This keeps the physics transparent while still requiring the model to learn non-trivial two-body correlations such as $\langle X \otimes X\rangle$ and $\langle Z \otimes Z\rangle$ from noisy finite-shot data.

2.3 Machine Learning Models

We treat entanglement detection as a supervised binary classification problem on 16-dimensional feature vectors derived from classical shadows. To understand how model capacity and inductive bias affect performance in the low-data regime, we benchmark four standard classifiers:

- **Logistic Regression (LR).** LR fits a linear decision boundary $\sigma(w^\top x + b)$ using the logistic (sigmoid) link and cross-entropy loss. It corresponds to maximizing the conditional likelihood $p(y \mid x)$ under a generalized linear model and provides a baseline for how close the entangled and separable classes are to being linearly separable in the reconstructed feature space.
- **Random Forest (RF).** RF is an ensemble of decision trees trained on bootstrap resamples of the training data, with random feature sub-sampling at each split. Each tree partitions the feature space into axis-aligned regions; the forest aggregates these weak learners by majority vote. This bagging strategy reduces variance and typically performs well on small, tabular datasets with non-linear decision boundaries.
- **Feedforward Neural Network (FNN).** The FNN is a fully-connected network with two hidden layers and ReLU activations. In principle, such networks are universal function approximators, capable of representing highly non-linear boundaries between entangled

and separable states. In practice, with $N \approx 230$ training examples, the network must be regularized (via dropout and early stopping / fixed epochs) to avoid overfitting the finite-shot noise present in the classical shadows.

- **Support Vector Machine (SVM) with RBF kernel.** The SVM seeks a maximum-margin separator in a (potentially infinite-dimensional) feature space induced by the radial basis function kernel

$$K(x, x') = \exp\big(-\gamma \|x - x'\|_2^2\big).$$

  The kernel implicitly maps classical-shadow features into a high-dimensional space where entangled and separable states may become linearly separable. The regularization parameter C controls the trade-off between margin size and misclassification on the training data.

All models are implemented using `scikit-learn` (for LR, RF, and SVM) and `PyTorch` (for the FNN). Class imbalance is mild but non-negligible, so we use class-balanced loss weights where supported.

3 Problem Formulation & Dataset

3.1 Task Definition

The objective is binary classification. Given a classical-shadow representation of a noisy 2-qubit state $\rho(p)$, we aim to predict whether the state is **entangled** ($y = 1$) or **separable** ($y = 0$).

3.2 Labeling & Data Generation

To avoid ambiguous labels near the entanglement-separability boundary, we introduce a safety gap. The labeling rule is:

$$y = \begin{cases} 0 & \text{if } C(\rho) \le 10^{-5}, \\ 1 & \text{if } C(\rho) \ge 0.1 \quad \text{(GapThreshold)}, \\ \text{discard} & \text{otherwise.} \end{cases} \tag{5}$$

Here 10^{-5} is a numerical tolerance: due to floating-point error in the concurrence computation, we treat any $C(\rho) \le 10^{-5}$ as "numerically separable". This cutoff is chosen to be much smaller than the entangled-side threshold (0.1), so it lies well inside the safety gap and does not affect the physical decision boundary.

We generate up to $N_{\text{samples}} = 250$ states per trial, sampling $p \sim U[0, 1]$ and applying the Werner channel to the Bell state. After discarding samples in the gap region, we obtain roughly $N \approx 230$ labeled examples per trial with a reasonably balanced class distribution.

3.3 Feature Extraction from Classical Shadows

For each state, we simulate $T = 25$ randomized joint Pauli measurements. We preserve quantum correlations by sampling outcomes from the joint probability distribution $P(o_1, o_2|b_1, b_2)$ for the two qubits, where b_i denotes the Pauli basis and $o_i \in \{\pm 1\}$ the measurement outcome. The raw shadows are converted into a 4×4 density-matrix estimator $\hat{\rho}_{\text{avg}}$, and we then flatten the real part into a 16-dimensional feature vector.

3.4 Rationale for the low-resource regime

In principle, simulations allow us to generate arbitrarily large datasets and take thousands of classical-shadow snapshots per state. In practice, however, near-term quantum devices are limited by state-preparation and measurement time, calibration overhead, and decoherence. Taking even a few hundred distinct states with tens of repeated measurements per state is already experimentally nontrivial. For this reason we concentrate on a realistic low-resource regime ($N_{\text{samples}} = 250$, $T_{\text{snapshots}} = 25$), rather than on asymptotic settings where tomography-like performance can be recovered with extremely large N and T. Our results should therefore be interpreted as a case study of what can be learned from shadows under experimentally plausible constraints.

4 Methodology: ML System Design

4.1 Feature Pipeline from Classical Shadows

Each quantum state in the dataset is represented by $T = 25$ classical-shadow snapshots. For each snapshot we perform a joint Pauli measurement on the two qubits by sampling random local bases $(\sigma_{b_1}, \sigma_{b_2}) \in \{X, Y, Z\}^2$ and projective outcomes $(o_1, o_2) \in \{\pm 1\}^2$ according to the Born rule

$$p(o_1, o_2 \mid b_1, b_2) = \mathrm{Tr}[\rho \, P_{b_1,o_1} \otimes P_{b_2,o_2}] \,,$$

where $P_{b,+} = (\mathbb{I} + \sigma_b)/2$ and $P_{b,-} = (\mathbb{I} - \sigma_b)/2$.

Given a basis–outcome pair (b, o) for a single qubit, the one-qubit classical-shadow estimator is

$$\hat{\rho}(b, o) = \frac{3}{2}(\mathbb{I}_2 + o\,\sigma_b) - \mathbb{I}_2$$

and for two qubits we form the tensor product estimator

$$\hat{\rho}^{(t)} = \hat{\rho}(b_1, o_1) \otimes \hat{\rho}(b_2, o_2) \in \mathbb{C}^{4\times 4}.$$

Averaging over T snapshots yields $\hat{\rho}_{\text{avg}}$. We then flatten the real part of $\hat{\rho}_{\text{avg}}$ into a 16-dimensional feature vector $x \in \mathbb{R}^{16}$, which explicitly encodes the reconstructed one- and two-body correlations. This fixed-length feature vector is used as input to all downstream machine learning models.

4.2 Model Architectures

We benchmark four standard classifiers implemented using `scikit-learn` and `PyTorch`:

- **Logistic Regression (LR).** A linear classifier with ℓ_2 regularization and class-balanced weights, serving as a baseline for linear separability of the reconstructed features.
- **Random Forest (RF).** An ensemble of decision trees with bootstrap sampling, where we tune the number of trees ($n_{\text{estimators}} \in \{50, 100\}$) and the maximum depth ($max\_depth \in \{\text{None}, 10\}$).
- **Feedforward Neural Network (FNN).** A fully-connected network with architecture $16 \to 64 \to 32 \to 1$ using ReLU activations, a dropout layer with rate 0.4 after the first hidden layer, and a final sigmoid output. The network is trained with the Adam optimizer and binary cross-entropy loss in `PyTorch`.
- **Support Vector Machine (SVM).** A margin-based classifier with a radial basis function (RBF) kernel operating directly on the 16-dimensional shadow features. We tune the regularization parameter C and kernel width γ.

4.3 Training and Hyperparameter Tuning

For each trial we split the dataset into training and test subsets using an 80/20 stratified split to preserve the entangled/separable balance. All reported F1 scores are computed on the held-out test set.

Hyperparameters are tuned as follows:

- LR, RF, and SVM use `GridSearchCV` with 3–5-fold cross-validation on the training set and F1 score as the selection metric.
- The FNN uses a manual grid search over hidden size, dropout rate, and learning rate. For each configuration we perform 5-fold stratified cross-validation and select the model with the highest mean validation F1. The final model is retrained on the full training set before evaluation on the test set.

Algorithm TrainAndEvaluate summarizes the training and evaluation loop for a single trial, while Algorithm MonteCarloStabExperiment describes the Monte Carlo stability protocol used to obtain error bars.

5 Experimental Setup

5.1 Implementation Platform and Software

All simulations and experiments are implemented in Python and executed on Google Colab with access to an NVIDIA T4 GPU. We use:

- QuTiP for quantum state preparation and measurement simulation,
- NumPy and SciPy for numerical operations,

- scikit-learn for LR, RF, SVM, grid search, and train/test splitting,
- PyTorch for implementing and training the FNN,
- Matplotlib for visualization.

5.2 Parameter regime

All reported experiments are carried out in a single, experimentally motivated low-resource regime. For each trial we request $N_{\text{samples}} = 250$ candidate states and perform $T_{\text{snapshots}} = 25$ joint Pauli measurements per state. After discarding ambiguous samples in the safety gap, this yields roughly $N \approx 230$ labeled examples per trial.

In principle, our simulator could generate arbitrarily large datasets and use thousands of snapshots per state. However, on near-term quantum hardware the number of distinct preparations and repeated measurements is limited by calibration overhead, decoherence, and wall-clock experiment time. The chosen (N, T) therefore emulate a realistic measurement budget rather than an asymptotic, tomography-like setting. Our goal is to study how far standard machine learning methods can go when constrained to such low-shot, low-data conditions.

5.3 Monte Carlo Stability Evaluation

Small datasets and finite-shot noise can induce large variance in performance across different random seeds. To avoid overinterpreting a single lucky split, we adopt a Monte Carlo evaluation protocol with $R = 100$ trials. For each trial:

- We regenerate a fresh dataset (new p values and new measurement outcomes),
- We perform an 80/20 stratified split,
- We train all models and record their F1 scores on the held-out test set.

We reported the mean and standard deviation of the F1 score across the 100 trials for each model.

6 Results & Discussion

In this section we present quantitative results for the low-resource setting $N_{\text{samples}} = 250$, $T_{\text{snapshots}} = 25$, using the safety gap described in Section Labeling & Data Generation. For each trial, this leads to roughly $N \approx 230$ labeled examples after discarding ambiguous states. We first show a representative single train–test split, and then aggregate over $R = 100$ Monte Carlo trials to assess the statistical stability of each model. Finally, we investigate the impact of hyperparameter tuning on the Logistic Regression and FNN baselines.

6.1 Representative low-resource trial

Figure Single-trial performance of all four models on a representative dataset ($N \approx 230$, $T = 25$). In this particular realization, the Shadow SVM achieves the highest F1 score (≈ 0.84), followed closely by Logistic Regression and Random Forest. The FNN lags slightly but remains in the same performance band reports the F1 scores of all four models on a single representative dataset generated in the low-resource regime. In this trial, the class distribution after applying the safety gap is reasonably balanced (95 separable vs. 135 entangled states), and the train–test split is stratified.

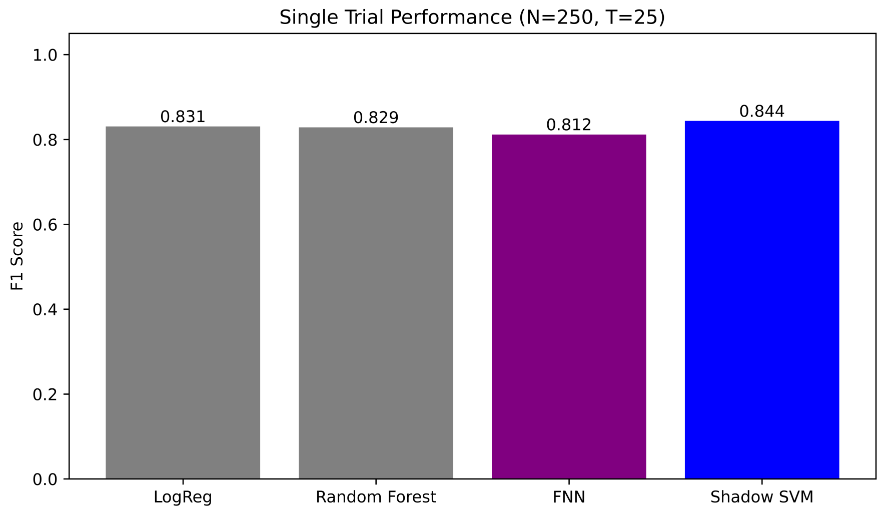

Figure 1: Single-trial performance of all four models on a representative dataset ($N \approx 230$, $T = 25$). In this particular realization, the Shadow SVM achieves the highest F1 score (≈ 0.84), followed closely by Logistic Regression and Random Forest. The FNN lags slightly but remains in the same performance band.

For this run, we obtain the F1 values approximately 0.83, 0.83, 0.81 and 0.84 for LR, RF, FNN and Shadow SVM respectively. This snapshot illustrates an important qualitative point: once the classical shadows are mapped to a 16-dimensional feature space, even a linear classifier performs competitively, and all non-linear models fall within a narrow F1 band on the order of 0.8–0.85. However, a single trial is not sufficient to draw strong conclusions about relative model performance, which motivates the Monte Carlo stability analysis below.

6.2 Monte Carlo stability analysis

To account for randomness in both the physics (sampling p and measurement outcomes) and the train–test split, we repeat the entire pipeline (GenerateDataset $\to$ TrainAndEvaluate) $R = 100$ times without fixing a global seed. For each trial we regenerate the dataset, retrain all four models from scratch, and record the test F1 scores. Algorithm MonteCarloStabExperiment in the Appendix summarizes this protocol.

The resulting mean and standard deviation over $R = 100$ trials are reported in Table Performance comparison over $R = 100$ independent trials in the low-resource regime ($N \approx 230$, $T = 25$). All three non-linear models cluster around F1 ≈ 0.8, with overlapping standard deviations and visualized in Figure Monte Carlo stability analysis over $R = 100$ independent trials in the low-resource regime ($N \approx 230$, $T = 25$). Error bars show one standard deviation around the mean F1 score. All non-linear models form a statistical tie around F1 ≈ 0.8.

Model	**Mean F1 Score (± Std)**
Logistic Regression	0.802 ± 0.049
Random Forest	0.820 ± 0.044
Neural Network (FNN)	0.814 ± 0.048
Shadow SVM (RBF kernel)	0.799 ± 0.053

Table 1: Performance comparison over $R = 100$ independent trials in the low-resource regime ($N \approx 230$, $T = 25$). All three non-linear models cluster around F1 ≈ 0.8, with overlapping standard deviations.

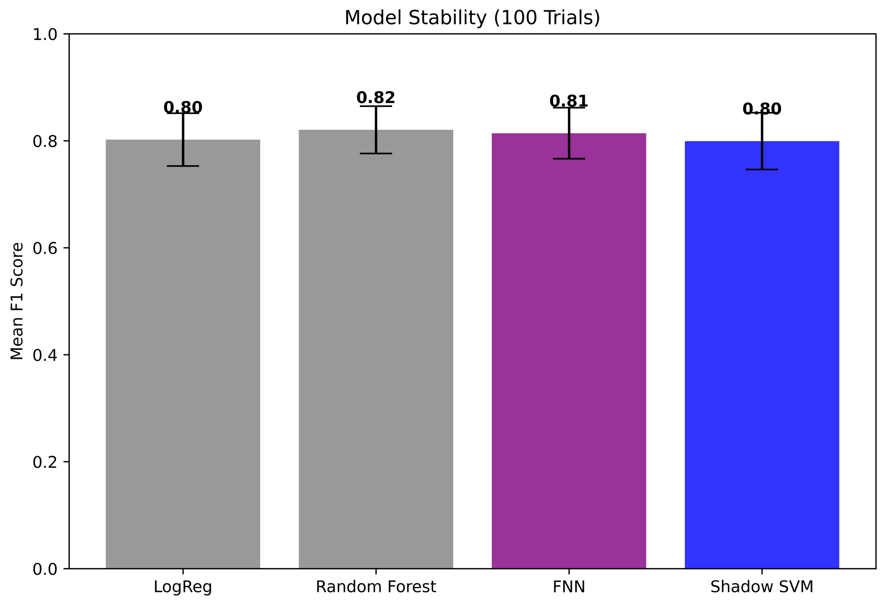

Figure 2: Monte Carlo stability analysis over $R = 100$ independent trials in the low-resource regime ($N \approx 230$, $T = 25$). Error bars show one standard deviation around the mean F1 score. All non-linear models form a statistical tie around F1 ≈ 0.8.

Several observations emerge:

- The spread in F1 across trials is non-negligible (standard deviation ≈ 0.04–0.05), reflecting the intrinsic variability induced by the small dataset size and finite measurement budget.
- Random Forest and FNN achieve the highest mean F1 scores (≈ 0.82 and ≈ 0.81, respectively), but their error bars overlap strongly with those of the Shadow SVM.
- Logistic Regression, despite being linear, only underperforms the best non-linear model by about 0.02 in mean F1, again with overlapping standard deviations.

Overall, the Monte Carlo study shows that in this low-resource regime no model emerges as a clear winner: all three non-linear approaches form a statistical tie, and even the linear baseline is competitive once classical-shadow features are provided.

6.3 Effect of hyperparameter tuning

The main Monte Carlo experiment fixes a simple, hand-chosen configuration for Logistic Regression and the FNN. To quantify how much gain can be obtained from more careful tuning on a *fixed* dataset, we perform an additional experiment on the representative dataset saved in Figure Single-trial performance of all four models on a representative dataset ($N \approx 230$, $T = 25$). In this particular realization, the Shadow SVM achieves the highest F1 score (≈ 0.84), followed closely by Logistic Regression and Random Forest. The FNN lags slightly but remains in the same performance band.

For Logistic Regression, we perform a grid search over the inverse regularization strength C and the solver. For the FNN, we conduct a manual 5-fold cross-validation over the hidden-layer width, dropout rate, and learning rate. Figure Impact of hyperparameter tuning on Logistic Regression and the FNN for a fixed representative dataset. "Baseline" denotes the hand-chosen configurations used in the main Monte Carlo experiment, whereas "Optimized" refers to models selected via grid search / cross-validation. Both models benefit from tuning, with the FNN showing the largest relative improvement compares baseline scores to tuned scores on the same train–test split.

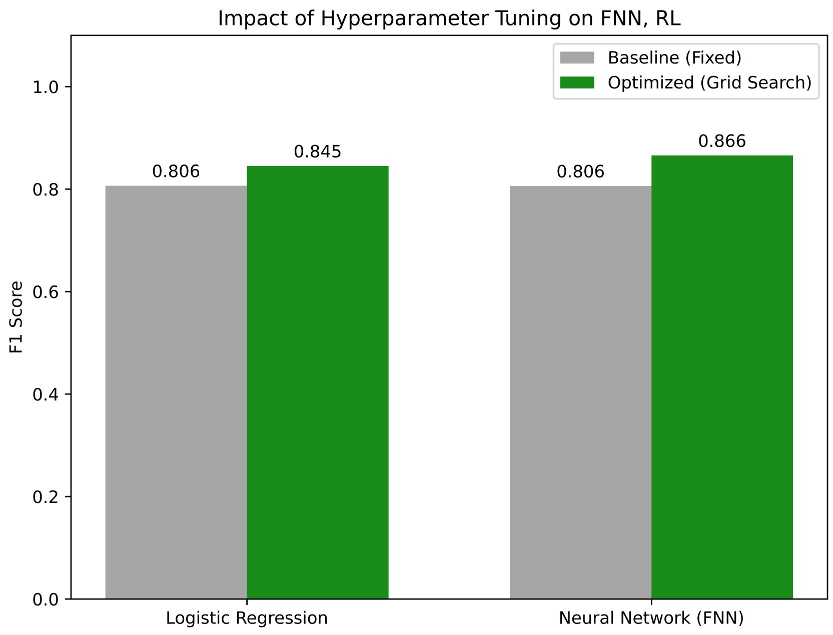

Figure 3: Impact of hyperparameter tuning on Logistic Regression and the FNN for a fixed representative dataset. "Baseline" denotes the hand-chosen configurations used in the main Monte Carlo experiment, whereas "Optimized" refers to models selected via grid search / cross-validation. Both models benefit from tuning, with the FNN showing the largest relative improvement.

On this dataset, the tuned Logistic Regression improves from an F1 of approximately 0.81 to 0.85, while the tuned FNN improves from about 0.81 to 0.87. This demonstrates that, given a fixed dataset, careful hyperparameter selection can recover some of the capacity of the neural model even in the low-data regime. However, these tuned scores are still bounded by the information content of the shadows: they do not contradict the Monte Carlo result that all models saturate near F1 ≈ 0.8 on average when we vary the underlying physics and measurement noise.

6.4 Discussion

Taken together, the experiments support three main conclusions:

1. **Classical shadows are the primary bottleneck.** Once we reconstruct a 4×4 density-matrix estimator from only $T = 25$ joint measurements, the resulting 16-dimensional feature vector already contains almost all the information that can be exploited by our models. Increasing model complexity (from LR to RF/FNN/SVM) yields only modest gains in mean F1, and all non-linear models form a statistical tie.

2. **Model choice matters less than measurement budget in this regime.** While the Shadow SVM can win on some individual trials (as in Figure Single-trial performance of all four models on a representative dataset ($N \approx 230$, $T = 25$). In this particular realization, the Shadow SVM achieves the highest F1 score (≈ 0.84), followed closely by Logistic Regression and Random Forest. The FNN lags slightly but remains in the same performance band), these advantages wash out when we average over many realizations. The dominant factor is the limited dataset size and shot noise, not the expressive power of the classifier.

3. **Tuning helps, but does not break the information-theoretic ceiling.** Hyperparameter optimization significantly improves the FNN and Logistic Regression on a fixed dataset (Figure Impact of hyperparameter tuning on Logistic Regression and the FNN for a fixed representative dataset. "Baseline" denotes the hand-chosen configurations used in the main Monte Carlo experiment, whereas "Optimized" refers to models selected via grid search / cross-validation. Both models benefit from tuning, with the FNN showing the largest relative improvement), but the overall scale of achievable F1 remains constrained by the low N and T. From a practical standpoint, this suggests that experimental effort is often better invested in collecting slightly more data (or reducing hardware noise) rather than deploying increasingly sophisticated models on a fixed, very small dataset.

For near-term quantum devices, where measurement budgets are costly, these results are encouraging: they show that relatively simple classical models, when combined with classical-shadow feature reconstruction, already provide a resource-efficient tool for binary entanglement detection. More elaborate architectures may become advantageous in higher-dimensional settings (e.g. many-body phases or multi-class problems), but in the two-qubit Werner case their benefits are largely saturated by the available information in the shadows.

7 Summary, Limitations, and Future Work

We implemented an end-to-end classical-shadow pipeline for entanglement phase classification of 2-qubit Werner states and benchmarked four standard machine learning models (Logistic Regression, Random Forest, Feedforward Neural Network, and RBF-kernel SVM). In a realistic low-resource regime with $N_{\text{samples}} = 250$ targeted states (yielding roughly $N \approx 230$ labeled examples after the safety gap) and $T_{\text{snapshots}} = 25$ joint Pauli measurements per state, we observed:

- All three non-linear models (RF, FNN, SVM) form a statistical tie in F1 score around 0.8, with overlapping standard deviations over $R = 100$ Monte Carlo trials.
- Even the linear Logistic Regression baseline remains competitive, trailing the best non-linear model by only ≈ 0.02 in mean F1.
- On a fixed representative dataset, careful hyperparameter tuning (grid search / cross-validation) can significantly improve both Logistic Regression and the FNN, but these improvements remain bounded by the information content of the classical shadows.

Taken together, these results suggest that in the two-qubit Werner setting with a tight measurement budget, the dominant bottleneck is the limited shadow information rather than the expressive power of the classifier. Once we reconstruct a 4×4 density-matrix estimator from $T = 25$ snapshots, relatively simple classical models already saturate the achievable performance for binary entanglement detection.

This study has several limitations:

- **Physics model:** We restrict attention to 2-qubit Werner states with a simple depolarizing channel. Real quantum hardware exhibits additional error sources such as readout bias, coherent control errors, and crosstalk, which are not included here.
- **System size:** For $N = 2$ and $T = 25$, the reconstructed density matrices are still reasonably well estimated. This regime does not fully stress-test the asymptotic advantages of classical shadows in high dimensions, where full reconstruction is impossible.
- **Task complexity:** The classification task is binary (entangled vs. separable) with a clean analytical label (concurrence). More complex phase diagrams (e.g., multiple entangled phases or topological order) are not considered.

Several extensions would make the framework both more realistic and more challenging:

- **Larger systems and richer states:** Extending to more general families of entangled states might better probe the scaling behavior of classical-shadow based classifiers.
- **Shadow kernels and raw-shadow learning:** Instead of first reconstructing $\hat{\rho}$, we could build kernel methods directly on shadow overlaps or train models on raw shadow data, comparing their performance and sample complexity to our feature-based approach.

Overall, the results indicate that for small-scale, noisy quantum systems, classical shadows combined with standard machine learning models already provide a practical and sample-efficient tool for characterizing basic quantum properties, while leaving ample room for future work in more complex and realistic settings.

References

[1] H.-Y. Huang, R. Kueng, and J. Preskill, “Predicting many properties of a quantum system from very few measurements,” *Nature Physics*, vol. 16, pp. 1050–1057, 2020.

[2] F. Pedregosa et al., “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[3] J.R. Johansson, P.D. Nation, and F. Nori, “QuTiP: An open-source Python framework for the dynamics of open quantum systems,” *Computer Physics Communications*, vol. 183, pp. 1760–1772, 2012.

Appendix

A. Code Availability

The full code for this project, including data generation and model training, is available at: `https://github.com/syedshubha/ShadowTomography`

B. Algorithm Details

Algorithm 1 ShadowToFeatures

```
1: Input: Shadow tensor $S$ of shape $(T, 2, 2)$
2: Initialize accumulator $A \leftarrow 0_{4\times4}$
3: for $t = 1$ to $T$ do
4:     Read bases $b_1, b_2$ and outcomes $o_1, o_2 \in \{\pm1\}$ from $S_t$
5:     Map $b \in \{0, 1, 2\}$ to Pauli $P_b \in \{Z, X, Y\}$
6:     $r_1 \leftarrow 3\big((I_2 + o_1 P_{b_1})/2\big) - I_2$
7:     $r_2 \leftarrow 3\big((I_2 + o_2 P_{b_2})/2\big) - I_2$
8:     $A \leftarrow A + (r_1 \otimes r_2)$
9: end for
10: $\hat{\rho}_{\text{avg}} \leftarrow A/T$
11: $x \leftarrow \text{Flatten}(\text{Re}(\hat{\rho}_{\text{avg}})) \in \mathbb{R}^{16}$
12: return $x$
```

Algorithm 2 GenerateDataset

```
1: Input: $N_{\text{samples}}, T_{\text{snapshots}}$, GapThreshold
2: Initialize $Dataset \leftarrow []$
3: while $|Dataset| < N_{\text{samples}}$ do
4:     Sample noise $p \sim U[0, 1]$
5:     Construct $\rho(p)$ and compute concurrence $C$
6:     if $C \leq 10^{-5}$ then
7:         $y \leftarrow 0$
8:     else if $C \geq$ GapThreshold then
9:         $y \leftarrow 1$
10:    else
11:        continue                                   ▷ Discard ambiguous state
12:    end if
13:    Simulate $T_{\text{snapshots}}$ joint Pauli measurements $\rightarrow$ shadow $S$
14:    Reconstruct feature vector $x \leftarrow$ ShadowToFeatures($S$)
15:    Append $(x, y)$ to $Dataset$
16: end while
17: return $Dataset$
```

Algorithm 3 TrainAndEvaluate

```
1: **Input:** $N_{\text{samples}}, T_{\text{snapshots}}$
2: $(X, y) \leftarrow$ GenerateDataset($N_{\text{samples}}, T_{\text{snapshots}}$)    ▷ Alg. GenerateDataset
3: Stratified split: $X_{\text{train}}, X_{\text{test}}, y_{\text{train}}, y_{\text{test}}$
4: **for** each model $M \in \{\text{LR}, \text{RF}, \text{FNN}, \text{SVM}\}$ **do**
5:     Perform hyperparameter search for $M$ on $(X_{\text{train}}, y_{\text{train}})$
6:     Train best model $M^\star$ on $(X_{\text{train}}, y_{\text{train}})$
7:     Predict $\hat{y} \leftarrow M^\star(X_{\text{test}})$
8:     Compute $F1(M) \leftarrow$ F1_Score($y_{\text{test}}, \hat{y}$)
9: **end for**
10: **return** $\{F1(\text{LR}), F1(\text{RF}), F1(\text{FNN}), F1(\text{SVM})\}$
```

Algorithm 4 MonteCarloStabExperiment

```
1: **Input:** $R$ (number of trials), $N_{\text{samples}}, T_{\text{snapshots}}$
2: Initialize $Results \leftarrow \{\text{LR} : [], \text{RF} : [], \text{FNN} : [], \text{SVM} : []\}$
3: **for** $trial = 1$ to $R$ **do**
4:     Optionally set random seed $\leftarrow trial$
5:     $\{F1_{\text{LR}}, F1_{\text{RF}}, F1_{\text{FNN}}, F1_{\text{SVM}}\} \leftarrow$ TrainAndEvaluate($N_{\text{samples}}, T_{\text{snapshots}}$)
6:     Append each $F1$ value to the corresponding list in $Results$
7: **end for**
8: For each model $M$, compute mean $\mu_M$ and standard deviation $\sigma_M$ of F1 in $Results[M]$
9: **return** $\{\mu_M \pm \sigma_M \mid M \in \{\text{LR}, \text{RF}, \text{FNN}, \text{SVM}\}\}$
```