Introduction:

- If you are doing something new, detail how you arrived at this topic and what motivated you.
    - Watching how volatile and seemingly irrational the crypto market can be, we became interested in whether deep learning could make sense of it. Our goal was not just to trade but to predict future price movements with measurable accuracy. Traditional models and rule-based bots often fall short in the face of sudden swings and nonlinear relationships. Deep learning, with its ability to recognize complex patterns in chaotic time-series data, felt like a natural fit.
    - What motivated us most was the challenge of predicting the price of Bitcoin. This is considered near-impossible by many, but we were drawn to the idea of making it probabilistically predictable. While it is not perfectly accurate, our model should be consistently close within a ±5% margin. That target reflects a realistic, risk-aware view of crypto forecasting. It's also a way to quantify success without falling into the trap of overfitting or hindsight bias.
- What kind of problem is this? Classification? Regression? Structured prediction? Reinforcement Learning? Unsupervised Learning? Etc.
    - This is a regression problem because the goal is to predict a continuous numerical value (the future price of Bitcoin). The model learns from historical time-series data to estimate the price at a given date, and success is measured by how close the prediction is to the actual price, ideally within a ±5% margin of error.

Challenges:

- The most difficult part of the project so far has been building a unified and flexible codebase that supports all of our different model architectures (MLP, CNN, RNN, LSTM, and Transformer). Preprocessing the crypto time-series data and ensuring consistency across input formats for each model was more complex than expected. We also spent a lot of time designing the main.py interface so that each model could be run and evaluated seamlessly from the command line. Another challenge has been making sure the system outputs interpretable metrics like MAE and loss in a clean, consistent way, which is key for comparing model performance.

Insights:

- We're getting promising early results. Although we haven't yet nailed down which model performs best overall, the MLP model gave us solid predictions with a low mean absolute error (MAE) and loss during training and validation. For example, from the output logs shown in the screenshot, the MLP model achieved validation MAEs around 0.0025–0.0084 and losses as low as 1.7738e-05. Sample predictions for Bitcoin's closing price were very close to actual values, with errors well within our ±5% target range. These results are encouraging, but we're still running tests across models to finalize comparisons.

Plan:

- We're on track with the project timeline. Most of the heavy lifting (data prep, model integration, and early testing) is complete. Now we're focusing on improving our results further by fixing small bugs, fine-tuning hyperparameters, and improving the performance of certain models. We also plan to streamline our main interface to better support experiment logging and model evaluation.