Sequential Win-Probability Modeling in *Slay the Spire* Using Floor-Level Representations and Recurrent Neural Architectures

Joseph Russoniello Fernando Martín

Abstract

Predicting the eventual outcome of a partially observed trajectory is a central challenge in sequential decision-making domains [9]. While such problems have been studied in sports analytics and reinforcement learning, they remain underexplored in procedurally driven deck-building games where strategic depth and combinatorial variability complicate state representation. This work introduces a modeling framework for floor-level win prediction in the roguelike deck-builder *Slay the Spire*. Using approximately 126,123 human-played runs comprising roughly 3 million reconstructed floor states, we develop a representation-learning pipeline that encodes each deck as an x-hot card vector and compresses it through truncated singular value decomposition (SVD) into a dense latent embedding. We integrate these embeddings with auxiliary game-state features and evaluate logistic regression [6], multilayer perceptrons [2], and recurrent neural models with and without attention [5].

Sequential models exhibit substantially improved predictive stability and accuracy across all floor horizons, achieving a validation accuracy of $\approx 92\%$, compared to $\approx 82\%$ for non-sequential baselines, and 56% for majority-class voting algorithms. The resulting trajectories display interpretable temporal structure: winning runs show steadily increasing predicted win probability, while losing runs show early stagnation, or sharp decreases before a major loss. These findings demonstrate that sequential modeling with learned deck embeddings offers a viable methodological foundation for state evaluation in complex strategic games.

1 Introduction

Outcome prediction in sequential decision-making is a fundamental machine learning task, appearing in applications ranging from gameplay analytics to sports forecasting and reinforcement learning. In such settings, a learner observes a trajectory of partially informative states and attempts to estimate the likelihood of eventual success. While this problem is well studied in physical sports and continuous-control domains, substantially less attention has been directed toward

environments characterized by high-dimensional compositional states and long-horizon strategic decisions.

Slay the Spire (STS), a prominent roguelike deck-building game, exemplifies these challenges [8]. A run consists of a sequence of floors, each representing a decision point involving combat, card selection, relic acquisition, or path navigation. The player's deck evolves over time through nonlinear interactions among card choices, upgrades, removals, and acquired relics. Predicting whether a run will ultimately succeed, given information available only at intermediate floors, therefore requires expressive state representations and models capable of aggregating evidence temporally.

Existing approaches to STS predominantly focus on reinforcement learning or combinatorial search, emphasizing optimal policy discovery rather than state evaluation. Other community-driven analyses typically rely on end-of-run summaries, which overlook how informative intermediate states become. Predicting win probability at arbitrary floors offers finer-grained insight into run development, enabling model-based reasoning about strategic pivot points and long-run viability of deck archetypes.

This paper formalizes the task of floor-level win prediction, constructs a large-scale floor-level dataset from raw gameplay logs, and evaluates a family of predictive models ranging from linear classifiers to recurrent neural networks with attention. A central contribution is the development of a dense deck embedding derived from SVD, which enables efficient and interpretable modeling of deck composition in a continuous space. Our experiments show that sequential architectures significantly outperform non-sequential baselines, achieving superior accuracy, calibration, and temporal stability.

2 Dataset and Floor-Level Reconstruction

The dataset consists of **126,123** raw gameplay logs taken from a public metrics dump of 77 million runs in November of 2020 [1]. Each run complete run is represented through detailed JSON representation of events. These logs encode the sequence of floors, card rewards, chosen cards, upgrades, removals, relic acquisitions, and combat results. However, the raw format provides only incremental modifications to the deck; explicit deck states must be reconstructed.

We implement a deterministic replay procedure that reconstructs the exact deck at every floor by starting from the initial starter deck and applying all recorded transformations in order. This includes upgrade paths, special-case transformations, removals, multi-card events, and relic acquisitions, which are tracked analogously. Each floor state also includes contextual information such as character class, ascension level, current and maximum HP, gold, room type, and path structure.

The resulting floor-level dataset contains approximately **3 Million** floor states. Each floor is labeled with the terminal outcome of the corresponding run, ensuring that only information available *at that floor* is incorporated into the input.

3 Representation Learning for Deck State

Deck state is central to STS trajectory prediction but is difficult to model directly due to its high-dimensional and sparse structure. The raw x-hot (one-hot with card count) count vector spans **1,380** dimensions, including variant-specific and upgraded cards. Direct use of this representation leads to instability, collinearity, curse of dimensionality concerns and poor generalization.

To obtain a dense and expressive representation, we compute a truncated singular value decomposition (SVD) of the card-count matrix [3]. Let $X \in \mathbb{R}^{N \times D}$ denote the matrix containing card-count vectors for all floors; we approximate $X \approx U_k \Sigma_k V_k^\top$ and use the embedding $Z = U_k \Sigma_k$ as the deck representation. Empirically, $k = 128$ components preserve roughly 86% of the variance while yielding interpretable latent dimensions.

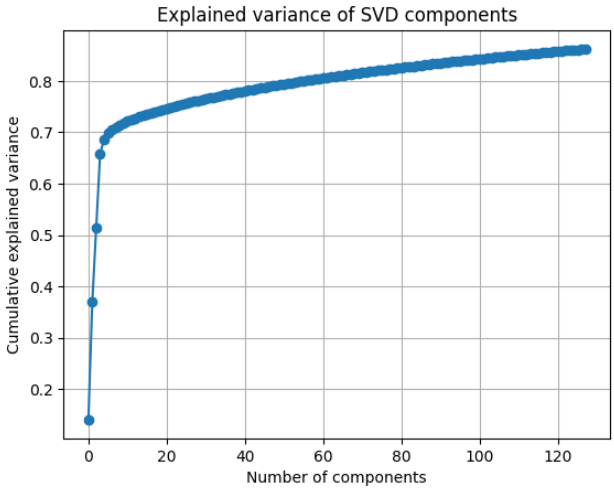

Figure 1: Explained Variance of SVD Components

Inspection of the leading components reveals structure aligned with known deck archetypes, such as strength scaling, poison engines, and orb-generation synergies. These embeddings serve as core inputs to all predictive models. Importantly, the first Singular Value component (SVD0) encodes the similarity to the starting state, a signal that gradually weakens throughout the runs, and the fifth component (SVD4) encodes run-deciding legendary cards that are found frequently in winning runs (see Figure 2). In this SVD0-SVD4 space, we can see the progression of average floor among runs with high SVD4 and low SVD0, indicating strong predictive signal (see Figure 3).

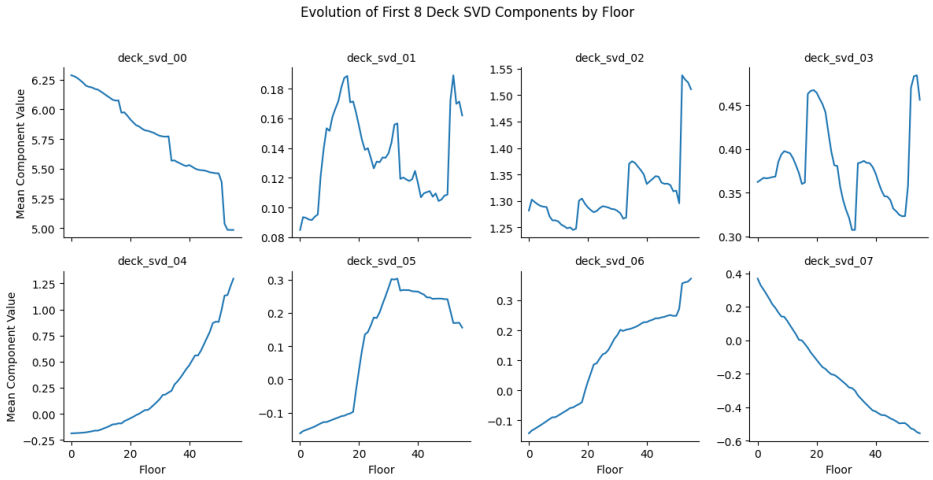

Figure 2: SVD Component Floor Scaling.

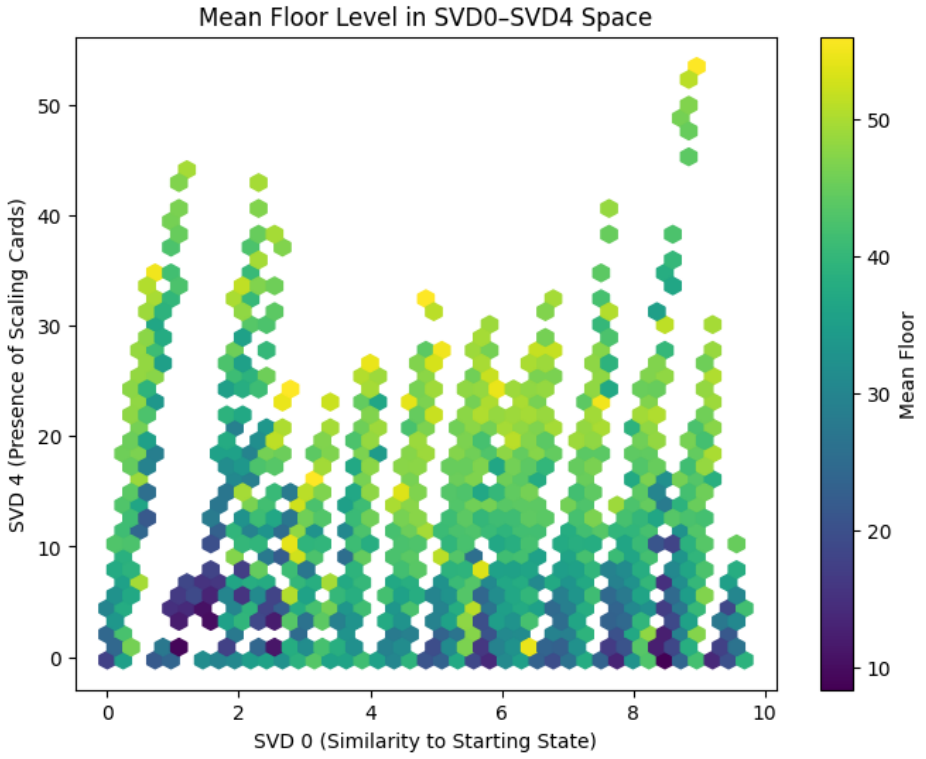

Figure 3: SVD0-SVD4 Space.

4 Predictive Models

The predictive task is to estimate

$$P(\text{win} \mid s_1, \ldots, s_f),$$

where s_f denotes the encoded state at floor f. We evaluate both non-sequential and sequential models.

4.1 Non-Sequential Baselines

Logistic regression on the SVD embeddings provides a natural and interpretable baseline for floor-level win prediction. Because each deck state is compressed into a 128-dimensional representation capturing global structure in card composition and upgrade patterns, the linear classifier is able to exploit broad monotonic relationships between latent deck features and eventual run outcomes. Empirically, this baseline achieves non-trivial predictive performance, indicating that even a single floor snapshot contains useful information about long-horizon viability. Nonetheless, the linearity of the model imposes a severe representational bottleneck: it cannot express nonlinear card interactions, threshold effects (e.g., damage-scaling breakpoints), or the cumulative trajectory of deck evolution over time. More importantly, because each floor is evaluated independently, the resulting probability trajectories exhibit pronounced temporal volatility. Small, strategically insignificant perturbations in deck state often produce large changes in predicted win probability, undermining interpretability and calibration.

To increase expressive capacity while maintaining a per-floor modeling framework, we evaluate a multilayer perceptron (MLP) with two hidden layers of width 128, ReLU nonlinearities, and dropout rates of 0.3 and 0.25 applied to the first and second hidden layers, respectively. This architecture introduces the ability to capture nonlinear synergies among SVD components and interactions between deck composition and auxiliary metadata such as ascension level or current HP ratio. In aggregate performance metrics, the MLP moderately improves over logistic regression, particularly in precision and recall for high-confidence predictions. However, temporal instability persists: the model still assigns probabilities independently across floors, producing irregular, oscillatory trajectories that bear little resemblance to the smooth strategic progressions observed in expert human play. The model is additionally prone to overfitting, requiring substantial regularization and carefully tuned learning rates to avoid memorizing idiosyncratic deck signatures.

4.2 Sequential Models

To capture the inherently sequential nature of deck-building progression, we train models that operate over full floor sequences rather than treating each state independently. Our primary architecture is a two-layer Long Short-Term Memory (LSTM) network with a hidden size of 128 and a recurrent dropout rate of 0.2. This configuration allows the model to accumulate informa-

tion across dozens of floors, integrating early-game pathing decisions, relic acquisition timing, and nonlinear deck transitions into a compact hidden state. The final hidden state is passed through a linear prediction head to generate a probability of eventual victory.

Whereas non-sequential models attempt to infer long-term prospects from a single floor in isolation, the LSTM constructs a temporally coherent latent trajectory of the run itself. This markedly improves predictive stability: win-probability curves produced by the LSTM are smooth, monotonic in expectation, and aligned with qualitative human judgments of run quality. Early experimentation with multiplicative temporal attention layers revealed only marginal gains and, in later epochs, degradation relative to the simpler LSTM architecture. The attention mechanism often overemphasized early floors, amplifying noise rather than selectively highlighting genuinely informative mid-game developments. As training progressed, the plain two-layer LSTM consistently achieved superior validation accuracy, AUC, and calibration, and ultimately became the preferred architecture.

Sequential modeling yields the largest performance gains across all evaluation regimes. Calibration curves demonstrate that the sequential model better aligns predicted probabilities with empirical success frequencies, particularly in the mid-game where nonlinear deck commitments begin to form. Most notably, horizon-restricted evaluations show that the LSTM can reliably differentiate winning and losing trajectories far earlier than non-sequential models, with significant separability emerging by floor 35 (66% through the game) and strengthening thereafter. These findings highlight the importance of temporal aggregation for modeling strategic games with long-range dependencies.

5 Experimental Setup

Runs are partitioned into training, validation, and test splits by run identifier to prevent leakage across sequence fragments. All models are trained with a regularizing weight decay of 10^{-4}, and are trained using early stopping with a patience of 5. All non-sequential models (Neural Networks and Logistic Regression) are trained with an Adam optimizer [7] with learning rate $\gamma = 0.001$. Sequential models were trained using an RMSProp [4] optimizer with learning rate $\gamma = 0.001$, and weight decay of 10^{-4}. To handle different sequence sizes across input batches, all recurrent sequences are padded with zeros, until they reach the length of the longest sequence in the batch. Then, during prediction, the model is given a tensor of sequence lengths to pack all padded sequences, reducing computational time spent processing on padded zeros.

Evaluation is conducted both on complete sequences and on horizon-truncated prefixes to assess how early the final outcome becomes predictable. Metrics include accuracy, F1, Precision, Recall, and AUC Scores.

6 Results

Sequential models achieve state-of-the-art performance for floor-level win prediction. The LSTM with attention achieves a validation accuracy of **91.82%**, compared to **82.28%** for logistic regression, **82.92%** for MLPs. Though, these validation estimates are inflated due to an 80% presence of the majority class (losses). In sequential models, ROC AUC improves substantially, reaching **90%** recall 10 floors before the end of a run.

Non-sequential models produce highly unstable trajectories, with predictions fluctuating sharply from floor to floor. Sequential models produce smooth, interpretable curves: winning runs exhibit steady increases in predicted win probability, while losing runs show early decay or stagnation.

Horizon analysis indicates that early-game floors offer limited predictive information, but mid-game floors yield significant separability between winning and losing outcomes. By the transition into Act 3, the sequential models achieve substantial lead over baselines while maintaining calibration.

The 3 Layer MLP and 2-Layer LSTM predictions for a sample of 4 winning and losing runs can be found and compared below

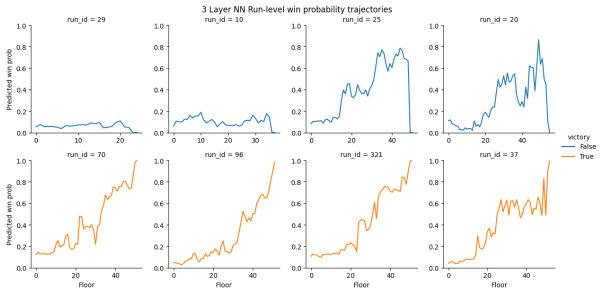

Figure 4: 3-layer MLP predictions.

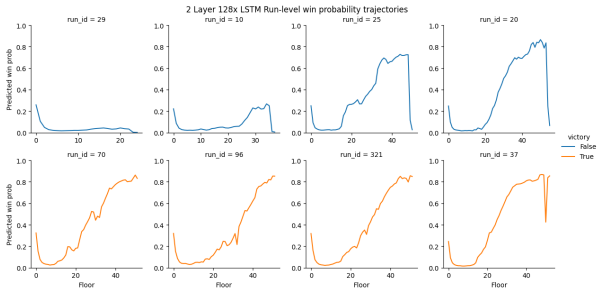

Figure 5: 2-layer LSTM predictions.

To evaluate models in a manner consistent with the temporal structure of a Slay the Spire run, we report results at two meaningful horizons: Floor 35 and Floor 50. Floor 35 corresponds to the transition into Act 3 (standard STS runs have three acts, each made up of 17 floors, the last of which is a boss fight), the point at which deck archetypes have typically solidified and major deck synergies begin to determine long-term survivability. Floor 50, immediately before the final boss encounter, represents the latest stage at which a model must rely solely on the accumulated trajectory rather than the outcome itself. Beyond overall accuracy, we emphasize recall as the most meaningful metric in this setting. Two runs with similar predictive "strength" may diverge in their final outcome due to poor human decision-making or random combat events, and a model that fails to recognize strong-but-unlucky runs as viable is of limited value. High recall therefore reflects the model's ability to correctly identify runs that possess the structural prerequisites for victory, independent of noisy or contingent late-run events.

Table 1: Mid-game predictive performance at the Floor 35 horizon

Model	Accuracy	Precision	Recall	F1 Score	AUC
Logistic Regression	0.92097	0.58125	0.51884	0.54828	0.74039
3-Layer MLP	0.91968	0.55806	0.62971	0.59172	0.78946
2-Layer 128 LSTM	0.91162	0.51362	**0.82647**	**0.63352**	**0.87338**
LSTM w/ Attention	**0.92353**	**0.59180**	0.55653	0.57362	0.75872

Table 2: Late-game predictive performance at the Floor 50 horizon

Model	Accuracy	Precision	Recall	F1 Score	AUC
Logistic Regression	0.93880	0.73125	0.53418	0.61737	0.75709
3-Layer MLP	**0.98080**	**0.87080**	0.93032	**0.89958**	0.95813
2-Layer 128 LSTM	0.97023	0.75948	**0.99211**	0.86035	**0.98006**
LSTM w/ Attention	0.96423	0.77857	0.85671	0.81577	0.91595

7 Discussion

The results highlight the importance of temporal modeling in complex strategic games. Deck composition alone produces moderate predictive power, but incorporating temporal structure enables the model to detect the momentum of a run and accumulate weak signals over time. The hidden state of the LSTM serves as an implicit representation of long-term synergy formation and strategic viability.

The late-game results further reveal an intriguing pattern: although sequential models dominate in AUC and recall, the 3-layer MLP attains the highest precision and F1 score at Floor 50. This indicates that, near the end of a run, single-state information becomes highly expressive, and nonlinear feed-forward architectures can correctly classify a large fraction of terminally decisive states without requiring temporal context. The strong late-game discriminative performance of the MLP suggests that a mixed-model approach that leverages recurrent architectures for early- and mid-game prediction, but switches to a per-floor classifier in the late game may yield further improvements. Such a hybrid strategy may allow the predictor to benefit from temporal aggregation when it matters most while leveraging the flexibility of high-capacity feed-forward models during the final stages of a run.

A potential confounding factor in floor-level win prediction is the simple monotonic relationship between floor number and survival: runs that reach higher floors are, by definition, disproportionately composed of stronger trajectories. Without proper controls, a model might exploit this structural bias rather than learning meaningful properties of deck composition or run dynamics. To

ensure that our models do not rely on this signal, we evaluate a floor-conditioned majority-class baseline that predicts the empirical win rate at each horizon. As shown in Figure 6, the accuracy of this baseline declines sharply across the mid- and late game, reflecting the fact that a large fraction of runs reaching deeper floors still ultimately lose. In contrast, the LSTM maintains consistently high accuracy throughout the entire run, demonstrating that its predictive advantage cannot be attributed to floor progression alone. The widening performance gap between the LSTM and the majority-vote baseline provides strong evidence that the sequential model captures latent structural features such as scaling mechanisms, deck synergies, or deck fragility that remain invisible to purely floor-based heuristics.

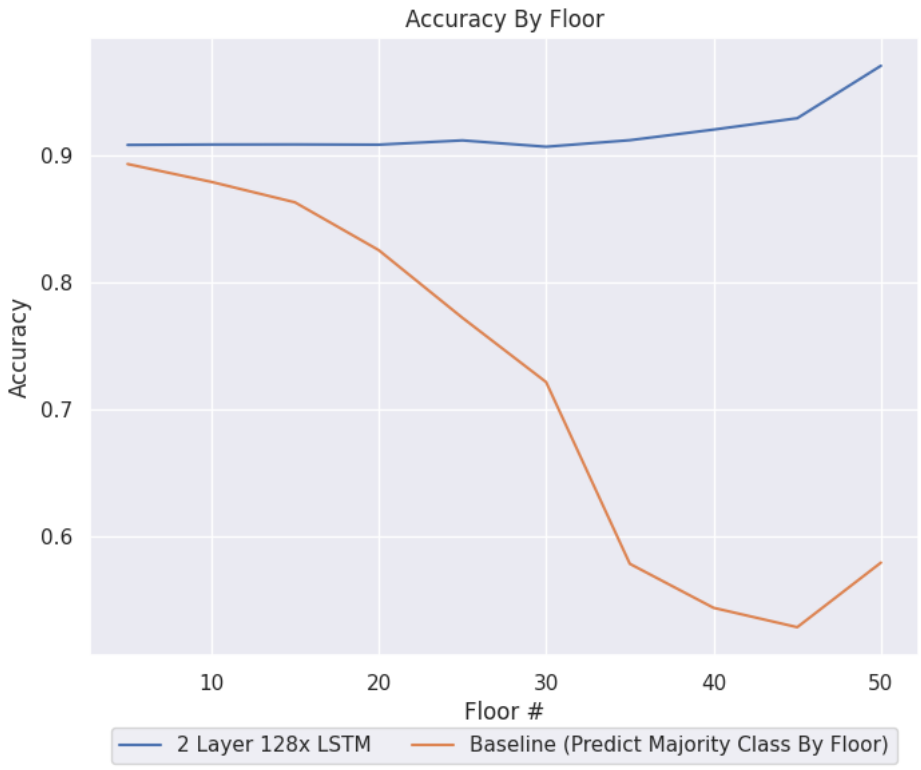

Figure 6: LSTM Floor Accuracy

7.1 What We Learned

This project taught us several concepts and techniques beyond the standard DS340 curriculum:

- How to work with event-based datasets to reconstruct tabular models
- The specifics of working with LSTM architectures. Especially regarding padding sequences across batches, and packing those padded sequences in model internals
- How to create custom PyTorch datasets and dataloaders, and the complexities required of an LSTM dataset to avoid overfitting on models end states

- The **noticable speed improvements (50x)** that come with allocating entire datasets on the GPU instead of using the X.to(device), Y.to(device) pattern discussed in lecture.
- A significant amount of work was put into creating a clean, customizable framework for architecture exploration in Pytorch. Notably, we are exceptionally proud of our ModelConfig $\rightarrow$ ModelBuilder $\rightarrow$ TrainableModel pipelines that **drastically reduced Pytorch boilerplate**, and made tweaks in architecture and hyperparameter tuning easy and quick.
- How to evaluate models when traditional metrics don't tell the full story. Precision, recall, F1 are all dependent on the floor predicted, so we created a custom model evaluation framework.

8 Conclusion

We presented a sequential modeling framework for predicting win probability in *Slay the Spire* from partial trajectories. Through floor-level data reconstruction, SVD-based deck embeddings, and recurrent neural architectures, we achieve substantial accuracy and stability gains over non-sequential models. This approach establishes a foundation for real-time state evaluation and offers insights into strategic progression in procedurally generated environments.

Future work may incorporate richer card semantics (eg: learned card embeddings), relic semantics, explore transformer architectures, multi-model algorithms, or integrate these predictors into reinforcement-learning agents as value estimators.

References

[1] dysonpark. 7.7 million runs: An slay the spire metrics dump. `https://www.reddit.com/r/slaythespire/comments/jt5y1w/77_million_runs_an_sts_metrics_dump/`, 2020. Accessed: 2025-10-15.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[3] Per Christian Hansen. The truncated SVD as a method for regularization. *BIT Numerical Mathematics*, 27(4):534–553, 1987.

[4] Geoffrey Hinton. Neural networks for machine learning: Lecture 6a. `https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf`, 2012.

[5] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9 (8):1735–1780, 1997.

[6] David Hosmer, Stanley Lemeshow, and Rodney Sturdivant. *Applied Logistic Regression*. Wiley, 2013.

[7] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

[8] MegaCrit. Slay the spire. Video Game, 2017. MegaCrit, https://www.megacrit.com/.

[9] Arthur Nelson and Dmitry Ryvkin. Contests with sequential moves: An experimental study. Florida State University, 2024.