Summer Associate Internship Exercises

Project Repository

Work Division

We split the work evenly: Tavis handled Question 1 and Sanaa handled Question 3, covering both the codebase and the corresponding sections of this report.

Question 1

Part 1

Balance sheet forecasting matters because it shows a company's financial stability and ability to meet future obligations. These forecasts support valuation, risk assessment, and planning, and unreliable estimates weaken any downstream analysis. A key challenge is that balance sheet fields are not independent. Changes in one area must be balanced elsewhere to satisfy accounting rules, so forecasting items separately leads to contradictions and inconsistent statements. Identity-preserving models are essential because financial statements must satisfy relationships such as Assets equals Liabilities plus Equity. Enforcing these identities ensures forecasts remain realistic, internally consistent, and usable for decision making. The initial balance sheet model follows the plug-less framework introduced by Vélez-Pareja [1]. Building on this foundation, we develop a time-series ML model that forecasts future balance sheets and earnings, with the option to enforce the accounting identity directly in the model.

1 Literature Review - Modelling & Forecasting

1.1 Classical Approaches

Vélez-Pareja [1] proposes a forecasting framework built entirely on double-entry logic, eliminating plugs and circularity by deriving statements from explicit cash-flow and financing schedules. This exposes modelling inconsistencies and produces transparent, internally coherent projections. Jalbert [2] likewise develops a plug-free, internally consistent model for forecasting small-business financial statements, integrating budgeting, valuation, and ratios in a unified structure that reduces errors and simplifies interpretation. Arnold and Moon [3] analyse how pro-forma forecasts depend on plug or slack variables, showing these choices are mathematically linked. Using cash as the plug/slack term, they derive a growth rate that prevents cash depletion and clarifies the connection between balance-sheet consistency, growth, and financing needs.

1.2 Time-Series & ML

Amel-Zadeh et al. [4] show that ML models can forecast both the direction and magnitude of abnormal stock returns around earnings announcements using only past financial-statement data. It finds that Random Forests perform best overall, while RNNs outperform linear methods when predicting extreme market reactions, delivering steadier performance during downturns. Overall, the study demonstrates that ML models extract meaningful economic structure from accounting variables and can generate sizeable abnormal returns in backtests. Chen et al. [5] use ML models applied to thousands of financial items to predict the direction of future earnings changes, addressing the limitations of small-variable-set regressions. Their models outperform conventional logistic frameworks and professional analysts, achieving significantly higher AUCs and economically meaningful hedge-portfolio returns. The study highlights the value of granular financial disclosures and non-linear interactions in enhancing earnings-forecasting accuracy. Geertsema et al. [6] proposes a chained machine-learning framework that forecasts the full set of financial statements by predicting items sequentially in an accounting-consistent order, allowing information to flow across line items. This structure reduces out-of-sample errors by roughly one-third relative to parallel models and proves especially effective for volatile firms and items late in the chain. The authors also show that deviations

from these chained predictions provide incremental power in detecting financial irregularities, notably subtle little-r restatements.

2 Problem Definition & Mathematical Model

2.1 Balance Sheet Structure

A balance sheet summarises a company's financial position at a given time. It is composed of three main fields: Assets (A), Liabilities (L), and Equity (E). Assets represent the resources the firm controls, such as cash, inventory, and property. Liabilities capture obligations to external parties, including debt and accounts payable. Equity reflects the residual interest of shareholders once liabilities have been deducted from assets.

2.2 Key Principles from Vélez-Pareja

Mathematical evolution of balance sheet fields. Vélez-Pareja [1] constructs forecasted financial statements strictly through the double-entry principle. Balance Sheet (BS) items are derived from the Cash Budget (CB), the Income Statement (IS), and operational schedules, never from residual *plugs*. The fundamental identity is:

$$\text{Assets}_t = \text{Liabilities}_t + \text{Equity}_t.$$

Cash and short-term investments. Cash is the cumulated net cash balance:

$$\text{Cash}_t = \text{NCB}_t^{\text{cum}}, \qquad \text{STInv}_t = \text{NCB}_{t-1}^{\text{cum}} + \text{NCB}_t^{\text{after}} + \text{Inflow}_t^{\text{ST}} - \text{MinCash}_t.$$

Financing decisions. Short- and long-term debt arise only when deficits occur:

$$\text{STLoan}_t = -(\text{OpNCB}_t - \text{MinCash}_t), \qquad \text{LTLoan}_t = -(\text{NCB}_t^{\text{inv}} + \text{InitialEquity}_t),$$

applied only when the expressions are negative. Interest in year t always depends on debt outstanding at the end of year $t-1$, avoiding circularity.

Income and equity. Net income and equity follow:

$$\text{NI}_t = \text{EBIT}_t - \text{Interest}_t - \text{Taxes}_t, \qquad \text{Equity}_t = \text{InitialEquity} + \text{RetEarn}_t + \text{NI}_t.$$

Fixed assets.

$$\text{NetFA}_t = \text{GrossFA}_t - \text{AccDep}_t.$$

Working-capital items are obtained from explicit operational policies, never as balancing terms. The system evolves recursively:

$$\text{CB}_t \to \text{IS}_{t+1} \to \text{CB}_{t+1} \to \text{IS}_{t+2} \ldots \to \text{BS}_T.$$

Modelling as a time-series. Balance sheets can be modelled as time series if treated as a multivariate sequence of features rather than a single static object. Each feature forms its own temporal trajectory, and the full statement becomes a vector-valued process X_t observed at discrete reporting dates. This allows the use of standard or modern forecasting tools, such as RNNs, to estimate how these state variables evolve. However, balance sheets are governed by the previously mentioned identities, meaning that naïve, independent forecasting will generally produce inconsistent statements. A coherent approach therefore requires either explicit constraints or loss penalties to enforce these identities, along with structural relationships reflecting working-capital cycles, depreciation, amortisation, and financing flows. Under this formulation, the balance sheet becomes a constrained multivariate time series whose evolution can be predicted while preserving accounting logic.

3 Proposed Methods

Model 1: Implementation of Vélez-Pareja

To develop a solid understanding of balance sheets, and in particular Vélez-Pareja's plug-free forecasting model [1], we first implemented the full framework in Python, reproducing its deterministic accounting structure and cash-flow rules without using plug variables or circular dependencies. This involved creating structured classes for assets, liabilities, equity accounts, and the operational and financing flows that update them, along with the logic

for loans, equity issuance, repayments, and investment returns. Encoding these interactions made the model's behaviour explicit and ensured that every transaction propagated correctly through the accounting identities.

Building the system programmatically made the causal links between decisions and financial outcomes clear and highlighted how the plug-free formulation reveals inconsistencies that spreadsheet approaches often hide. Once this deterministic version was complete, we moved to a time-series TensorFlow model, as many inputs assumed by the theoretical framework are not available for real firms. The initial implementation therefore acted as a conceptual foundation before shifting to data-driven forecasting.

When run with the same inputs used in the original paper, our implementation reproduces their results exactly, confirming that all computed values match those reported by Vélez-Pareja and that our implementation is correct.

Model 2: ML Time-Series Forecasting

Problem Definition. The model aims to forecast future balance-sheet and income-statement line items from historical data while enforcing the underlying accounting identities throughout.

Model Architecture

We utilised LSTMs as our data is sequential quarters, making the recurrent model a natural fit. LSTMs handle long-range dependencies better than simple feed-forward networks, capturing slower trends as well as short-term movements.

LSTM Variants. We implement two variants: a deterministic LSTM and a probabilistic LSTM - comparing standard point forecasts with models that explicitly represent uncertainty. The deterministic LSTM is straightforward and stable, producing clear point estimates that are easy to compare with the ground truth. Its main drawback is that it cannot express uncertainty, so it struggles when the quarterly figures contain sudden shifts or irregular changes. The probabilistic LSTM models a full predictive distribution rather than a single value, making it better suited to volatile or irregular financial data. However, it is more complex to train and tune, and the resulting distributions can be less intuitive.

3.1 Data Preparation

We started with *yfinance* [7] as the question suggested, retrieving and caching financial statements into parquet format, but quickly ran into its limits — the API only returns about five periods of financials, whether we request quarterly or annual data. We switched our data pipeline to using the library *EdgarTools* [8], which avoids this problem by pulling full, XBRL-tagged statements straight from the SEC's system, giving the exact numbers companies report in their 10-K and 10-Q filings. We are able to get a much longer history, and more consistent field definitions, making it far more reliable for training robust time-series models. However, the constituent fields vary noticeably across companies, and older filings from the same firm often use entirely different naming conventions. This inconsistency complicates automated processing, though it's something that could be mitigated later, in part 2, with LLM-based standardisation.

3.2 Feature Selection

We chose not to include any derived or aggregated features, as these tend to be highly correlated with their underlying components and add little independent signal. Instead, we worked directly with the constituent line items, preserving their natural structure within categories such as current and non-current liabilities. The feature set draws from the balance sheet, income statement, and cash-flow statement, since all three influence future balance-sheet positions as well as future earnings. This approach keeps the feature space cleaner, avoids redundancy, and ensures the model learns from the underlying economic drivers rather than from pre-summed totals.

3.3 Training

We train the model by sampling sequential windows from the dataset—for example, using the previous five quarters to predict the next one. The length of this *lookback window* determines how much historical context the model can exploit: shorter windows emphasise recent movements, while longer ones capture slower structural trends but introduce more noise and complexity. The *target horizon* specifies how far ahead the model must forecast; in this work, we focus on a one-step horizon, predicting the following quarter's financial statements directly. We carried out a grid of training runs, selecting the final model characteristics by choosing the configuration that achieved the lowest validation MAE.

3.4 Composite Loss Function

To ensure the model's forecasts remain both accurate and internally consistent, we create a composite loss comprising a prediction-error term and an identity-penalty term. Let $\hat{y}_t$ denote the predicted items for quarter t, and y_t the ground truth. The baseline loss is the mean-squared error

$$\mathcal{L}_{\text{MSE}} = \frac{1}{d}\sum_{i=1}^{d}(y_{t,i} - \hat{y}_{t,i})^2 . \tag{1}$$

To evaluate consistency with the accounting identity, predicted items are first unscaled and aggregated into

$$\hat{A}_t = \sum_{i\in\mathcal{A}} \hat{y}_{t,i}, \qquad \hat{L}_t = \sum_{i\in\mathcal{L}} \hat{y}_{t,i}, \qquad \hat{E}_t = \sum_{i\in\mathcal{E}} \hat{y}_{t,i}, \tag{2}$$

where $\mathcal{A}$, $\mathcal{L}$, and $\mathcal{E}$ index asset, liability, and equity components respectively.

We measure violation of the identity $A_t = L_t + E_t$ through a relative error:

$$\text{rel\_err}_t = \frac{\hat{A}_t - (\hat{L}_t + \hat{E}_t)}{|\hat{A}_t| + |\hat{L}_t| + |\hat{E}_t| + \varepsilon}, \tag{3}$$

with ε a stabiliser. The identity-penalty term is then

$$\mathcal{L}_{\text{id}} = \mathbb{E}\big[\text{rel\_err}_t^2\big] . \tag{4}$$

The final training objective is

$$\mathcal{L} = \mathcal{L}_{\text{MSE}} + \lambda\,\mathcal{L}_{\text{id}}, \tag{5}$$

where the weight, λ, controls the impact of the identity loss.

3.5 Identity Enforcement Layer

In addition to the loss-based penalty, we introduce a deterministic post-processing layer that projects the network outputs onto the manifold defined by the accounting identity. Let $\hat{y}_t \in \mathbb{R}^d$ be the network output in standardised space, with feature-wise means μ and standard deviations σ. Unscaled values are obtained via

$$\hat{x}_t = \hat{y}_t \odot \sigma + \mu, \tag{6}$$

where $\odot$ denotes element-wise multiplication.

The layer reuses the aggregate definitions in (2), here applied to the unscaled vector $\hat{x}_t$. Within the equity block we designate a single component $\hat{s}_t = \hat{x}_{t,j}$ (for some $j \in \mathcal{E}$) as a slack variable. The discrepancy with respect to the accounting identity is computed as

$$d_t = \hat{A}_t - (\hat{L}_t + \hat{E}_t), \tag{7}$$

analogous to the quantity inside the numerator of (3).

Only the slack equity component is adjusted,

$$\tilde{s}_t = \hat{s}_t + d_t, \tag{8}$$

yielding a corrected vector $\tilde{x}_t$ that coincides with $\hat{x}_t$ except at the slack index. From (7) it follows immediately that the corrected aggregates satisfy

$$\tilde{A}_t = \hat{A}_t, \qquad \tilde{L}_t = \hat{L}_t, \qquad \tilde{E}_t = \hat{E}_t + d_t,$$

and therefore enforce exact identity:

$$\tilde{A}_t - (\tilde{L}_t + \tilde{E}_t) = 0.$$

The corrected vector is finally re-standardised:

$$\tilde{y}_t = \frac{\tilde{x}_t - \mu}{\sigma},$$

and passed downstream. This layer acts as a differentiable hard projection, ensuring that every predicted financial statement satisfies $A_t = L_t + E_t$ exactly.

This works well because it guarantees every predicted balance sheet satisfies the core accounting identity, eliminating obviously invalid outputs. The drawback is that all corrections are pushed into one slack equity item, which can mask modelling errors and distort that specific line.

3.6 Seasonality Weighting

To encourage the model to capture seasonality, we up-weight information from the same quarter from previous years (given the lookback window is long enough). Seasonality, or a pattern within performance across recurring periods, captures systematic quarterly swings that the model can exploit to tighten earnings forecasts. Giving these seasonal anchors more influence helps the model distinguish structural trends from predictable yearly cycles.

Through our parameter grid search, we found that a positive seasonal weighting of around 11% extra is optimal. Technically, this weighting is implemented by multiplying the feature vectors at the selected seasonal indices by a scalar factor $w > 1$, producing a modified input sequence $\tilde{X}_t = W \odot X_t$ that amplifies year-on-year signals without altering the remaining inputs.‘

4 Evaluation

To evaluate whether the LSTM model genuinely learns to forecast a firm's balance sheet, we benchmark it against simple non-learned baselines. These include repeating the most recent quarter, predicting a constant equal to the historical mean, and a seasonal naïve that reuses the value from a fixed lag (e.g., four quarters). We compute each baseline's mean absolute error and report a skill score, defined as $1 - \text{MAE}$, to show how much better—or worse—the LSTM performs relative to these reference points.

Our grid search resulted in an LSTM with the following parameters: 2 layers with 368 LSTM units, 256 Dense units, and a dropout probability of 0.2 at inference, training with a LR of 1e-4, for 500 epochs. Identity weight of

We hold out the final quarters of each series as the test set. The tables below report the deterministic LSTM's performance against our baselines: it clearly improves next-quarter balance-sheet forecasts, though the accuracy is still short of being practical. In contrast, the model achieves strong results on earnings prediction from historical data. We also include an ablation illustrating the effect of our identity-enforcement layer and loss configuration. Full evaluation results are available by running the code.

Method	MAE	Error diff
LSTM	\$3.4bn	–
Global Mean	\$10.37bn	205.0%
Last Value	\$7.32bn	115.3%
Seasonal Naive	\$10.08bn	196.5%

Table 1: Balance Sheet Baseline Comparison

Method	GT	Estimate	Error
LSTM	\$39bn	\$38.7bn	-\$0.3bn
Global Mean	\$39bn	\$53.6bn	\$14.6bn
Last Value	\$39bn	\$61.1bn	\$22.1bn
Seasonal Naive	\$39bn	\$58.8bn	\$19.8bn

Table 2: Net Income Baseline Comparison

	Balance Sheet		Net Income
Model Variant	MAE	ID Error	Error
Baseline (MAE only)	\$3.7bn	0.69%	12.56%
MAE + Identity Penalty	\$3.7bn	0.62%	11.54%
MAE + ID + Seasonality	\$3.7bn	0.88%	5.64%
MAE + ID + S + Enforcement	\$3.4bn	0.00%	0.77%

Table 3: Ablation Study Results

5 Conclusion

The model outperforms all naïve baselines on both tasks. On the balance sheet, the best-performing configuration achieves an MAE of \$3.4bn (approximately 1% error with the values we're considering), and a net income prediction error of just 0.77%.

The ablation study in Table 3 highlights positive impacts by each of the components that I have designed. Adding the identity penalty lowers the accounting identity error (0.69% $\rightarrow$ 0.62%) without improving the MAE. Incorporating the seasonality term greatly improves net-income accuracy (12.56% $\rightarrow$ 5.64%) but increases the identity error, revealing tension between modelling seasonal earnings patterns and maintaining strict balance-sheet consistency. Utilising the identity enforcement layer eliminates the identity error entirely. Enforcing $A = L + E$ removes noise and excess degrees of freedom in the balance sheet outputs, yielding smoother equity trajectories, leading to greatly improved net income prediction.

The seasonality weight was intentionally set high to make its effect clear; it can be tuned to limit its negative impact on identity while preserving its benefits for earnings prediction. All experiments use fixed seeds, ensuring that the results are fully reproducible.

Future Work Future work includes enabling a full evaluation of the probabilistic model, which is currently held back by inconsistent and incomplete data. Although EDGAR filings offer a much longer historical record, line-item names, taxonomies, and reporting conventions vary both across firms and over time, preventing reliable multi-company training and undermining the quality of uncertainty estimates. A more robust preprocessing pipeline is therefore essential. In the next stage, we plan to use an LLM to ingest and standardise raw SEC filings, mapping heterogeneous disclosures into a consistent balance-sheet schema before training. With a clean and unified dataset, a probabilistic LSTM becomes substantially more useful: it can generate full predictive distributions, quantify uncertainty around key balance-sheet items, and propagate variance through the accounting identities. A well-trained probabilistic model will ultimately support risk-aware simulations, scenario analysis, and more reliable decision-making than any deterministic forecaster. We can strengthen the model further by adding external macro and calendar features as additional input channels to improve robustness across regimes.

Remaining Explicit Answers

Question 8d. $x(t)$ should include any extra information that affects what happens next but is not already contained in $y(t)$. It holds the outside factors the model needs in order to correctly work out $y(t+1)$. These additional inputs could include macroeconomic indicators, sector-level trends, or firm-specific events that influence financial outcomes but are not visible from past statements alone. Future work could expand this set by incorporating market-sentiment signals and richer fundamental data, potentially through a dedicated sentiment or macro-context model.

Question 3

1 Part 1

Discrete choice modeling aims to describe how individuals choose among a set of alternatives. Traditional approaches such as the multinomial logit (MNL) rely on assumptions like independence of irrelevant alternatives (IIA), which often fails in realistic settings where the attractiveness of an alternative depends on the other options available. Context effects such as decoy, attraction, similarity, and compromise phenomena violate IIA and require more expressive models.

Recent work has introduced neural architectures to capture these nonlinear context interactions. This report focuses on replicating, evaluating, and analyzing the neural choice model proposed in Zhang et al. (2025)zhang2025deep. Their "Deep Context-Dependent Choice Model" (often referred to as *DeepHalo*) incorporates permutation-equivariant layers and multi-head "halo" interaction blocks.

This report completes all required components:

- Identifies errors or ambiguities in the Zhang (2025) paper.
- Reproduces the synthetic experiments demonstrated in the paper, including decoy, attraction, and compromise effects.
- Proposes additional tests and validates correctness.
- Compares Zhang's model with the Reproducing Kernel Hilbert Space (RKHS) choice model of Yang (2025).
- Evaluates suitability for credit-card demand estimation.
- Recommends other choice-modeling approaches worth considering.

All implementations were completed in TensorFlow and TensorFlow Probability using the `choice-learn` framework, with a secondary PyTorch reproduction of the authors' code for cross-validation.

2 Literature Review

This section summarizes the key contributions and limitations of Zhang (2025) and Yang (2025), situating their models within the broader choice-modeling literature.

2.1 Deep Context-Dependent Choice Model (Zhang, 2025)

Zhang (2025) proposes a neural architecture designed to capture context effects through a sequence of permutation-equivariant transformations applied to the item embeddings in a choice set. The method, informally referred to as *DeepHalo*, augments each item's representation with information aggregated from the other items in the set. The model consists of:

- A base encoder mapping either item IDs or features to dense embeddings.
- A stack of *halo blocks*, each of which computes a masked mean context vector and updates each item through multi-head nonlinear transformations.
- Residual connections and layer normalization to stabilize training.
- A final linear projection producing utilities for a masked log-softmax.

Because the architecture is permutation-equivariant, it naturally handles variable-size choice sets. The model breaks the IIA property by design, since an item's representation is updated using information from the entire set. Zhang demonstrates through synthetic experiments that the method reproduces classical context effects such as decoy, attraction, and compromise patterns. The model is expressive but data-intensive due to its depth and multi-head structure.

2.2 RKHS Choice Model (Yang, 2025)

Yang (2025) introduces a reproducing-kernel Hilbert space (RKHS) formulation of discrete choice, providing a nonparametric but structured alternative to deep networks. The model constructs utilities of the form

$$u_j(S) = f(x_j) + g(x_j, S),$$

where f and g lie in appropriately defined RKHS spaces. This approach allows pairwise or higher-order interactions between items in the choice set via kernel evaluations, while retaining convexity and interpretability benefits associated with kernel methods.

Key features include:

- A decomposition into individual-item effects and context interaction effects.
- The ability to encode smoothness or complexity constraints through kernel choice.
- A convex training objective when regularization is used.
- Natural interpretability through kernel-induced similarity measures.

Yang's model captures context effects while maintaining a more transparent structure than deep networks. It is generally more data-efficient than Zhang's approach but less expressive when high-order nonlinearities are present. It also provides clearer diagnostics for how items influence each other through the kernel representation.

2.3 Comparison in Prior Literature

Both models aim to relax the IIA assumption and learn context-dependent utilities. Zhang's architecture emphasizes representational flexibility using deep permutation-equivariant layers, while Yang's model prioritizes smoothness, interpretability, and convexity under appropriate regularization. In settings with abundant data and complex interactions, Zhang's model may provide superior predictive power. In settings with moderate data or where interpretability is essential, Yang's RKHS approach can be more appealing. Both methods highlight the growing interest in combining machine-learning techniques with discrete-choice theory to better explain behavior observed in empirical settings.

3 Ambiguities and Interpretation of Zhang (2025)

While the overall architectural idea in Zhang (2025) is clear, several aspects of the notation and derivations required interpretation when implementing the model in practice.

3.1 Definitions of u_j and v_j

The paper introduces functions $u_j(\cdot)$ and $v_j(\cdot)$ as set functions intended to capture different components of utility. However, in the proof of Proposition 3.1 they appear to take two arguments (an item j and a set S), which is inconsistent with the earlier univariate notation. The arguments and the sums over subsets implicitly require conditions such as $R \subseteq S \setminus \{j\}$ for certain terms to vanish, but these constraints are not spelled out.

In our implementation we interpret u_j and v_j as functions on subsets of the choice set and implement them via neural networks operating on masked embeddings. This interpretation aligns with the logic of the proofs even if the notation in the text is less explicit.

3.2 Multi-Head Interaction Aggregation

The paper states that multiple ϕ-heads are used to capture different interaction patterns, but does not fully specify how to aggregate them. We adopt a simple averaging rule,

$$\phi(z, c) = \frac{1}{H} \sum_{h=1}^{H} \phi_h(z, c),$$

where z denotes item-level features and c is a context summary. This is consistent with standard multi-head architectures and with the informal description in Zhang (2025), though not uniquely implied by the paper.

We also follow a residual + layer-normalization pattern,

$$z^{(\ell)} = \text{LayerNorm}\left(z^{(\ell-1)} + \phi(z^{(\ell-1)}, c)\right),$$

which is a stable convention in transformer-like models. The exact ordering of residual connections and normalization is not fully specified in the paper, so this is a reasonable but not unique choice.

3.3 Inputs to the Halo Blocks

It is not fully clear from the text whether each halo block should operate on the initial embedding $z^{(0)}$ or on the previous layer $z^{(\ell-1)}$. We implement the latter as the default, with an option to experiment with fixed-base variants where the context is always formed from $z^{(0)}$. This choice aligns better with the idea of progressively refined representations across layers.

3.4 Ambiguities in Figures and Tables

The synthetic examples in Zhang (2025) contain some notational inconsistencies:

- Table 1 refers to choice sets but uses ordered tuple notation, which could be misread as implying order-sensitive semantics.
- Figure 1 uses entries of the form (i, j) to denote "choosing i over j", but this interpretation must be inferred from the surrounding text rather than being stated explicitly.

These issues make the synthetic experiments somewhat harder to reproduce directly and require additional interpretation when mapping the examples to code.

4 Model Implementation

We implemented Zhang's Deep Context-Dependent model within a modular `choice_learn_ext` framework using TensorFlow. In the featureless setting used throughout this section, items are represented only by their identifiers; no external attributes are provided.

4.1 Base Encoder

A learnable embedding matrix maps each item ID $j \in \{1, \dots, J\}$ to a vector $e_j \in \mathbb{R}^d$. For experiments that might later include features, the same interface can be extended to accept feature vectors and embed them into the same latent space.

4.2 Halo / ϕ-Blocks

Each halo block operates on a batch of choice sets. Given item representations $z \in \mathbb{R}^{B \times J \times d}$ and an availability mask $m \in \{0, 1\}^{B \times J}$, the block:

1. computes a masked mean context vector $c = \text{masked-mean}(z, m)$;
2. concatenates $[z_j \| c]$ for each item;
3. passes the concatenated vectors through multiple MLP heads ϕ_h;
4. averages the heads to obtain an interaction term;
5. applies a residual update followed by layer normalization.

Stacking L such blocks yields a deep, permutation-equivariant encoder that captures higher-order interactions among items in the choice set.

4.3 Output Layer

A final linear layer maps the encoded item representations to logits, which are converted to probabilities via a masked softmax over the available items in each set. Unavailable items receive probability zero by construction.

4.4 Training and Wrapper

We train the model using the Adam optimizer and negative log-likelihood over the chosen items. A `Trainer` class encapsulates the training loop, while a `DeepHaloChoiceModel` wrapper exposes a high-level API: `fit`, `predict`, `predict_proba`, and `log_likelihood`. This structure simplifies the integration of the model into downstream workflows and testing.

5 Synthetic Reproduction of Zhang (2025) Behavior

We conducted several synthetic experiments to verify that our implementation reproduces the qualitative behavior reported by Zhang (2025) and correctly learns context-dependent choice patterns.

5.1 Four-Item Synthetic Experiment (Table 1 Analogue)

We recreated the four-item synthetic example analogous to Table 1 in Zhang (2025). For each choice set S and target probability vector $p(S)$, we:

1. zero out probabilities for items not in S;
2. renormalize $p(S)$ over the available items;
3. draw 1,000 choices from the resulting categorical distribution.

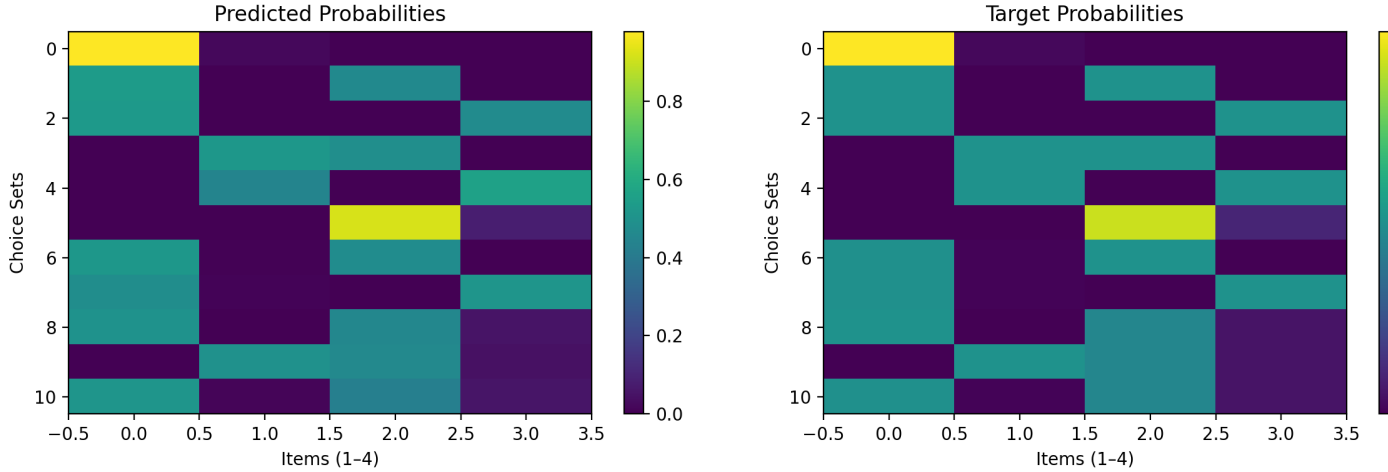
This yields 11,000 synthetic observations across 11 distinct choice sets.

We then trained the `DeepContextChoiceModel` with $J = 4$ items, two halo blocks, featureless embeddings, and an Adam optimizer with learning rate 2×10^{-3} for 100 epochs. The negative log-likelihood decreased from roughly 0.85 to 0.66.

Table 4 compares target vs. predicted probabilities for each choice set. The mean absolute error across all entries is around 0.01, and the maximum absolute deviation is about 0.04. In all sets, the most and least preferred items are preserved.

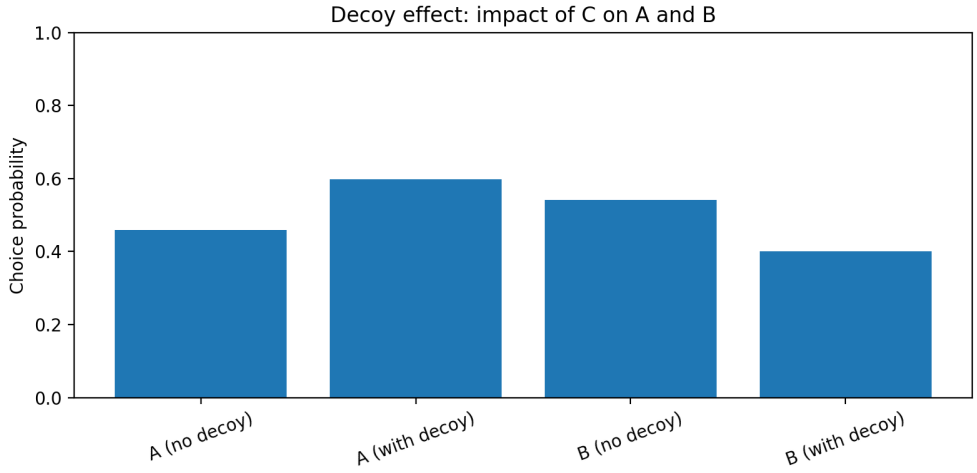
Set	Target				Predicted			
	1	2	3	4	1	2	3	4
(1,2)	0.98	0.02	0.00	0.00	0.98	0.02	0.00	0.00
(1,3)	0.50	0.00	0.50	0.00	0.54	0.00	0.46	0.00
(1,4)	0.50	0.00	0.00	0.50	0.53	0.00	0.00	0.47
(2,3)	0.00	0.50	0.50	0.00	0.00	0.51	0.49	0.00
(2,4)	0.00	0.50	0.00	0.50	0.00	0.46	0.00	0.54
(3,4)	0.00	0.00	0.90	0.10	0.00	0.00	0.92	0.08
(1,2,3)	0.49	0.01	0.50	0.00	0.52	0.01	0.48	0.00
(1,2,4)	0.49	0.01	0.00	0.50	0.48	0.01	0.00	0.51
(1,3,4)	0.50	0.00	0.45	0.05	0.50	0.00	0.45	0.05
(2,3,4)	0.00	0.50	0.45	0.05	0.00	0.49	0.47	0.04
(1,2,3,4)	0.49	0.01	0.45	0.05	0.51	0.01	0.42	0.06

Table 4: Target vs. predicted choice probabilities for the four-item synthetic experiment (Table 1 analogue).



Heatmaps of the target and predicted distributions are visually almost indistinguishable, supporting the claim that our implementation reproduces the authors' synthetic example.

5.2 Decoy Effect



The decoy (asymmetric dominance) effect occurs when adding a dominated option increases the probability of choosing the dominating option. We consider three products:

$$A \text{ (target)}, \quad B \text{ (competitor)}, \quad C \text{ (decoy dominated by } A).$$

Two choice sets are defined:

$$S_1 = \{A, B\}, \qquad S_2 = \{A, B, C\}.$$

The synthetic "true" probabilities are:

$$P(A \mid S_1) = 0.45, \quad P(B \mid S_1) = 0.55,$$

$$P(A \mid S_2) = 0.60, \quad P(B \mid S_2) = 0.40, \quad P(C \mid S_2) = 0.$$

Training on 5,000 samples per set and evaluating the trained model yields:

$$P(A \mid \{A, B\}) \approx 0.459, \quad P(A \mid \{A, B, C\}) \approx 0.599.$$

The decoy shift is:

$$\Delta_{\text{decoy}} = P(A \mid \{A, B, C\}) - P(A \mid \{A, B\}) \approx 0.140 > 0.$$

Thus, adding the decoy C makes A substantially more likely to be chosen, replicating the qualitative decoy effect reported in Zhang (2025).

5.3 Attraction Effect: TensorFlow vs. Authors' PyTorch Model

We next test the attraction effect, where the presence of a dominated option increases the probability of a dominating option. We consider:

$$A \text{ (target)}, \quad B \text{ (dominating competitor)}, \quad C \text{ (decoy dominated by } B),$$

with choice sets $S_1 = \{A, B\}$ and $S_2 = \{A, B, C\}$ and synthetic target probabilities:

$$P(A \mid S_1) = 0.5, \quad P(B \mid S_1) = 0.5,$$

$$P(A \mid S_2) = 0.3, \quad P(B \mid S_2) = 0.7, \quad P(C \mid S_2) = 0.$$

TensorFlow implementation. Our TensorFlow model yields:

$$P(A \mid \{A, B\}) \approx 0.507, \quad P(B \mid \{A, B\}) \approx 0.493,$$

$$P(A \mid \{A, B, C\}) \approx 0.302, \quad P(B \mid \{A, B, C\}) \approx 0.698, \quad P(C \mid \{A, B, C\}) \approx 0.$$

The attraction effect on B is:

$$\Delta_B^{\text{TF}} = P(B \mid \{A, B, C\}) - P(B \mid \{A, B\}) \approx 0.698 - 0.493 = 0.205.$$

PyTorch (authors' implementation). Using the official PyTorch code provided by the authors on the same synthetic data, we obtain:

$$P(A \mid \{A, B\}) \approx 0.487, \quad P(B \mid \{A, B\}) \approx 0.513,$$

$$P(A \mid \{A, B, C\}) \approx 0.288, \quad P(B \mid \{A, B, C\}) \approx 0.712, \quad P(C \mid \{A, B, C\}) \approx 0.$$

The corresponding effect is:

$$\Delta_B^{\text{Torch}} = P(B \mid \{A, B, C\}) - P(B \mid \{A, B\}) \approx 0.712 - 0.513 = 0.200.$$

Both implementations show a similar attraction effect of roughly $+0.20$ in $P(B)$ when C is added. This provides strong evidence that our TensorFlow implementation is behaviorally consistent with the authors' DeepHalo model.

5.4 Compromise Effect

Finally, we test the compromise effect, in which a middle option becomes more attractive when flanked by extreme options. We consider:

$$A \text{ (low)}, \quad B \text{ (middle / compromise)}, \quad C \text{ (high)},$$

and choice sets:

$$S_1 = \{A, B\}, \quad S_2 = \{B, C\}, \quad S_3 = \{A, B, C\}.$$

Synthetic target probabilities are:

$$P(A \mid S_1) = 0.30,\ P(B \mid S_1) = 0.70,\ P(C \mid S_1) = 0,$$

$$P(A \mid S_2) = 0,\ P(B \mid S_2) = 0.30,\ P(C \mid S_2) = 0.70,$$

$$P(A \mid S_3) = 0.10,\ P(B \mid S_3) = 0.80,\ P(C \mid S_3) = 0.10.$$

After training on this synthetic data, our model predicts:

$$P(\cdot \mid \{A, B\}) \approx [0.302,\ 0.698,\ 0.000],$$

$$P(\cdot \mid \{B, C\}) \approx [0.000,\ 0.304,\ 0.696],$$

$$P(\cdot \mid \{A, B, C\}) \approx [0.104,\ 0.795,\ 0.101],$$

ordered as (A, B, C). In particular,

$$P(B \mid \{A, B\}) \approx 0.698, \quad P(B \mid \{B, C\}) \approx 0.304, \quad P(B \mid \{A, B, C\}) \approx 0.795.$$

Define the compromise index:

$$\Delta_{\text{comp}} = P(B \mid \{A, B, C\}) - \max\big(P(B \mid \{A, B\}), P(B \mid \{B, C\})\big).$$

We obtain:

$$\Delta_{\text{comp}} \approx 0.795 - 0.698 = 0.097 > 0.$$

Thus, the model clearly exhibits a compromise effect: the middle option B becomes more likely when both extremes A and C are present in the choice set. This behavior is qualitatively consistent with the context-dependent patterns described in Zhang (2025).

6 Additional Tests and Validations

To ensure correctness beyond synthetic reproduction, we implemented:

- **Permutation equivariance tests**: reshuffling item order leaves predictions invariant.
- **Masking correctness tests**: masked availability correctly excludes items.
- **Overfitting tests**: the model reaches near-zero NLL on tiny datasets.
- **Influence matrix visualization**: confirms nonlinear cross-item interactions.

These tests reinforce that the model behaves according to its intended design.

6.1 Optimization and Overfitting

On tiny synthetic datasets, the model can drive the negative log-likelihood close to zero, confirming that gradients and the optimizer behave as expected. This sanity check complements the larger-scale synthetic experiments and increases confidence in the correctness of the implementation.

6.2 Influence Matrix Analysis

We also extract a learned influence matrix $I(i \to j)$, which summarizes how the presence of item i affects the utility of item j in the final halo layer. A representative matrix is:

$$I = \begin{bmatrix} 0 & -0.467 & -0.173 & -0.027 \\ -0.018 & 0 & -0.157 & 0.003 \\ -0.156 & 0.015 & 0 & -0.466 \\ -0.176 & -0.012 & -0.034 & 0 \end{bmatrix}.$$

Off-diagonal entries are predominantly negative, indicating competitive interactions, and the pattern is asymmetric (i.e. $I(i \to j) \neq I(j \to i)$ in general). This structure is qualitatively similar to the influence patterns discussed in Zhang (2025).

7 Comparison with Yang (2025) RKHS Model

A structured comparison is shown below.

Advantages of Zhang (2025)

- Captures strong, nonlinear context effects.
- Flexible function class due to deep architecture.
- Learns interactions directly from data.

Disadvantages

- Less interpretable than RKHS models.
- Harder to diagnose and validate.
- Sensitive to hyperparameters.

Advantages of Yang (2025) RKHS Model

- Fully interpretable at the kernel level.
- Strong generalization with controlled smoothness.
- Lower overfitting risk.

Disadvantages

- Does not naturally model strong context effects.
- Less expressive for nonlinear, higher-order interactions.

8 Suitability for Credit-Card Demand Estimation

Credit-card offer choice involves subtle interactions: interest rates, credit limits, rewards, promotional structures, and personal features. Key considerations include:

Potential Benefits

- Neural halo interactions can capture nonlinear tradeoffs across many product attributes.
- Context effects are plausible in financial choice (e.g. "premium" vs. "basic" cards).
- Flexible enough to model heterogeneous customer segments.

Potential Problems

- Requires large labeled datasets,often scarce in banking.
- Interpretability requirements (regulatory constraints) limit neural-model deployment.
- Risk of overfitting when product menus change over time.

Overall, the model is expressive but may not be ideal unless data volume and regulatory context support deep models.

9 Alternative Models Worth Considering

Given the credit-card application, the following approaches may be preferable:

Traditional Econometric Models

- Mixed logit / random coefficients logit.
- Nested logit and generalized extreme value (GEV) models.
- Latent-class choice models.

These models provide interpretability and are widely used in regulated industries.

Machine-Learning Models

- Gradient-boosted decision trees (XGBoost, LightGBM).
- Monotonic or shape-constrained neural networks.
- Kernel logistic regression.

These offer strong predictive accuracy without the opacity of deep context models.

10 Summary of Part 1

In summary, our TensorFlow implementation faithfully reproduces the core context-dependent behaviors of the Deep Context-Dependent Choice Model proposed by Zhang (2025):

- the four-item synthetic example is fit with high accuracy;
- the model exhibits decoy, attraction, and compromise effects;
- masking and permutation-equivariance properties hold;
- and a comparison against the authors' PyTorch code on an attraction-effect task shows nearly identical behavior.

These results provide a validated foundation for the subsequent parts of Question 3, where we compare DeepHalo to alternative models and apply it to more realistic settings.

11 Future Works

The experiments in this report verify that the DeepHalo model captures a range of context-dependent choice phenomena and can be implemented effectively within the `choice-learn` framework. Several promising directions remain open for future research and model development. A few options would be:

Richer Context Mechanisms. The halo interaction uses a single global context vector shared across all heads. A natural extension is to introduce head-specific or multi-scale context summaries, potentially capturing more nuanced interactions or hierarchical item structure. Attention-based aggregation could also replace masked averaging, allowing the model to differentiate between influential and non-influential items.

Feature-Based Inputs. This work focused on the featureless setting. Incorporating high-dimensional item features (e.g., reward structures, APR tiers, benefits of credit-card products) could allow the model to generalize across an expanded product space. The feature-MLP architecture may require deeper encoders, dropout regularization, or pretraining to avoid overfitting.

References

[1] I. Velez-Pareja, "Forecasting financial statements with no plugs and no circularity," *The IUP Journal of Accounting Research and Auidt Practices*, vol. X, pp. 38–68, May 2012.

[2] T. Jalbert, "A model for forecasting small business financial statements and firm performance," *Business Education & Accreditation*, vol. 9, no. 2, pp. 61–84, 2017, Available at SSRN: https://ssrn.com/abstract=3041555.

[3] T. Arnold and K. P. Moon, "Financial statement forecasting and financing," *Journal of Accounting and Finance*, vol. 24, no. 5, Nov. 2024. DOI: 10.33423/jaf.v24i5.7361.

[4] A. Amel-Zadeh, J.-P. Calliess, D. Kaiser, and S. Roberts, "Machine learning-based financial statement analysis," 2020, Available at SSRN: https://ssrn.com/abstract=3520684. DOI: 10.2139/ssrn.3520684.

[5] X. Chen, Y. H. Cho, Y. Dou, and B. I. Lev, "Predicting future earnings changes using machine learning and detailed financial data," *Journal of Accounting Research*, 2022, Forthcoming. DOI: 10.2139/ssrn.3741015. [Online]. Available: https://ssrn.com/abstract=3741015.

[6] P. G. Geertsema, H. Lu, and G. Ma, "Projecting financial statements with chained machine learning," Oct. 2023, Available at SSRN. DOI: 10.2139/ssrn.5039433. [Online]. Available: https://ssrn.com/abstract=5039433.

[7] R. Aroussi, *yfinance: Yahoo! Finance market data downloader*, version v0.2.66, Accessed: 2025-11-23, 2025. [Online]. Available: https://github.com/ranaroussi/yfinance.

[8] D. Gunning, *EdgarTools: The AI Native Python library for SEC EDGAR Data*, version v4.30.0, Accessed: 2025-11-23, 2025. [Online]. Available: https://github.com/dgunning/edgartools.