Question 3: Discrete choice model and credit card offers

Feng Qiu (feng.qiu.1101@gmail.com)
Ying(Alice) Xia (alice.ying.xia@gmail.com)

November 2025

Abstract

This report provides a comprehensive analysis of the Deep Context-Dependent Choice Model (DeepHalo) proposed by Zhang et al. (2025), combining technical implementation details with an extensive literature review of discrete choice modeling approaches. We identify and resolve ambiguities in the original paper, implement the model in TensorFlow, successfully replicate the synthetic data experiments, and validate the implementation through additional verification tests. We then situate DeepHalo within the broader landscape of discrete choice models by reviewing ten major methodologies spanning classical econometric approaches (Multinomial Logit, Nested Logit, Mixed Logit), modern deep learning methods (RUM-nets, TasteNet, TCNet), and context-dependent models (RKHS Choice, Low-Rank Halo MNL). Based on comprehensive analysis of theoretical properties, practical considerations, and empirical validation, we propose DeepHalo as the optimal methodology for credit card offer modeling. The implementation achieves results matching the paper within numerical tolerance and passes all verification tests, confirming readiness for real-world application.

Contents

1 Introduction

Credit card providers face the challenge of understanding consumer choice behavior to design attractive products, optimize pricing, and forecast demand effectively. This problem is complicated by several factors including high-dimensional feature spaces, context-dependent preferences where choice depends on the full set of available alternatives, and complex feature interactions that cannot be captured by simple additive models.

The DeepHalo framework (**?**) addresses these challenges by combining the halo approach to context-dependent choice with deep residual networks. The model provides explicit control over interaction order through network depth: a network with L layers can represent interactions up to order 2^{L-1}. This exponential growth in expressiveness enables the model to capture arbitrarily complex patterns while maintaining theoretical guarantees and computational tractability.

This report serves two primary purposes. First, it provides a detailed technical analysis of the DeepHalo architecture, identifying ambiguities in the original paper, documenting our TensorFlow implementation, and validating the model through comprehensive replication studies and additional verification tests. Second, it situates DeepHalo within the broader literature on discrete choice modeling, comparing it systematically to alternative approaches and justifying its selection for credit card applications.

The report is organized as follows. Section 2 reviews the credit card choice problem and the importance of context-dependence and feature interactions. Section 3 provides an extensive literature review of ten major discrete choice modeling approaches. Section 4 presents our technical analysis of the DeepHalo paper, identifying unclear points and implementation details. Section 5 documents our replication of the synthetic data experiments. Section 6 describes additional verification tests. Section 7 compares DeepHalo with the RKHS Choice Model. Section 8 evaluates suitability for credit card applications. Section 9 discusses alternative models. Section 10 concludes.

2 The Credit Card Choice Problem

2.1 Context-Dependence in Credit Card Selection

Credit card choices exhibit strong context effects that violate the Independence of Irrelevant Alternatives (IIA) assumption of classical models. Consider a consumer choosing between two cards: Card A with 15% APR and Card B with 18% APR. In isolation, the consumer might choose A with 60% probability and B with 40% probability. However, when a third card C with 25% APR is added to the choice set, the relative preference between A and B may change substantially. Card B might become more attractive as a "middle option" (compromise effect), or Card A might become even more dominant due to anchoring effects.

These context effects are not merely theoretical curiosities but have important practical implications. Premium credit cards with \$95 annual fees often sell better when \$450 cards are also offered in the choice set, creating a compromise effect. Similarly, an 18% APR seems reasonable when compared to 22% alternatives but appears unfavorable next to 12% options. Understanding and modeling these context-dependent preferences is essential for optimal product design and assortment planning.

The attraction effect (Huber et al., 1982) occurs when adding a dominated decoy option increases the choice share of the dominating alternative. The compromise effect (Simonson, 1989) occurs when alternatives positioned as middle options become more attractive. These phenomena are widespread in credit card markets where consumers face complex tradeoffs between APR, rewards rates, annual fees, and other features.

2.2 Feature Interactions in Credit Card Utility

Credit card utility depends on complex interactions among multiple features. At the simplest level, we have pairwise (second-order) interactions: the value of a balance transfer offer depends on the card's APR, and the value of rewards depends on the consumer's spending patterns. However, utility often involves higher-order interactions. For example, the joint effect of APR, rewards rate, and annual fee cannot be decomposed into separate pairwise effects. A consumer might accept a high annual fee only if both the rewards rate is generous and the APR is competitive.

The decisions and behaviors of firms and establishments are often analyzed under the simplifying assumption that they operate independently, both across firms and across establishments within the same firm. In practice, however, choices such as adopting new technologies or expanding output frequently depend on the actions of competitors. Explicitly modeling these strategic interactions is challenging, and the work of (Berry et al., 1993) illustrates how intricate general-equilibrium market modeling can become. Nevertheless, extending current frameworks to incorporate richer equilibrium interactions remains a natural and important direction for future research.

Traditional econometric approaches require researchers to manually specify each interaction term of interest. For a card with p features, there are $\binom{p}{2}$ possible pairwise interactions, $\binom{p}{3}$ three-way interactions, and so on. Manual specification quickly becomes infeasible and may miss important interaction structures. Modern deep learning approaches can automatically learn interactions but typically provide no control over or insight into the order of interactions being captured. DeepHalo addresses this gap by providing explicit control: selecting depth $L = 5$ guarantees representation of interactions up to order $2^4 = 16$.

3 Literature Review of Discrete Choice Models

This section reviews ten major approaches to discrete choice modeling, evaluating their suitability for credit card applications.

3.1 Classical Econometric Models

Multinomial Logit (MNL):
The Multinomial Logit model (**?**) remains the workhorse of discrete choice analysis. The model specifies utility as $U_i = x_i'\beta + \varepsilon_i$ where x_i are observed features, β are parameters, and ε_i follows a Type-I extreme value distribution. This leads to closed-form choice probabilities $P(i|S) = \exp(x_i'\beta)/\sum_{j\in S}\exp(x_j'\beta)$.

The model provides interpretable coefficients representing marginal utilities and is computationally trivial to estimate via maximum likelihood. However, MNL's assumption of IIA severely limits its applicability. The model predicts that the ratio $P(i|S)/P(j|S)$ is independent of other alternatives in the choice set, which is empirically violated in many applications including credit cards. MNL cannot capture context effects and is restricted to additive utility functions without interactions.

A growing area of academic concern in discrete choice experiments (DCEs) is the issue of endogeneity.
Endogeneity may arise in DCEs for several reasons, including the presence of unobserved product or alternative-specific attributes that correlate with observed explanatory variables (Train, 2009). To address this problem, several methodological approaches have been developed. These include the Berry–Levinsohn–Pakes (BLP) framework for handling endogenous product characteristics in differentiated-products markets (Berry et al., 1993), the control-function approach proposed by (Petrin and Train, 2010), and the full maximum-likelihood approach of (Park and Gupta, 2009).

Nested Logit:
Nested Logit (McFadden, 1977) provides additional flexibility by organizing choices into a tree structure with nests. Within each nest, IIA is relaxed through correlated error terms. For credit cards, one might nest by issuer (Chase, Citi, American Express) or tier (basic, premium, luxury). The model allows for more realistic substitution patterns within nests while maintaining tractability.

However, the nest structure must be specified a priori and remains rigid once chosen. Different nest specifications can lead to substantially different results, and there is limited guidance on appropriate nesting for many applications. The model still cannot capture higher-order feature interactions and provides only modest improvement over MNL for complex choice problems involving strong context effects.

Mixed Logit (Random Coefficients Model):
Mixed Logit (Train, 2009) represents a substantial advance by allowing preference parameters β_i to vary across individuals according to a specified distribution $F(\beta|\theta)$. The conditional choice probability is $P(i|S, \beta_i) = \exp(x_i'\beta_i)/\sum_{j\in S}\exp(x_j'\beta_i)$, and the marginal probability is obtained by integrating over the distribution: $P(i|S) = \int P(i|S,\beta)\, dF(\beta|\theta)$.

This formulation captures consumer heterogeneity and eliminates the IIA restriction through flexible substitution patterns. Mixed Logit can approximate any random utility model arbitrarily well (**?**). Estimation requires simulation-based methods (typically Maximum Simulated Likelihood or Bayesian MCMC), which are computationally intensive but feasible with modern computing resources.(Rossi and Allenby, 2003)

Despite these advantages, Mixed Logit remains context-independent: utility depends only on the alternative's own features, not on the composition of the choice set. Additionally, researchers must still manually specify interaction terms. The computational burden increases rapidly with the number of random coefficients, and convergence can be problematic with high-dimensional random coefficient distributions.

3.2 Modern Deep Learning Approaches

RUM-nets:
RUM-nets (Aouad and Désir, 2025) introduced deep neural networks into discrete choice modeling while maintaining consistency with Random Utility Maximization. The model specifies utility as $V_i = f_\theta(x_i)$ where f_θ is a deep neural network, and choice probabilities follow the standard softmax rule. This approach provides tremendous flexibility in the functional form of utility, automatically learning complex feature transformations and interactions.

The neural network can represent arbitrary nonlinear functions of the features, enabling it to capture patterns that would require extensive manual interaction specification in classical models. RUM-nets have demonstrated superior out-of-sample performance compared to MNL in various applications. However, they retain a fundamental limitation: they are context-independent. The utility of alternative i depends only on its own features x_i, not on the characteristics of other alternatives in the choice set. Additionally, the neural network operates as a black box, providing little insight into what interactions are being captured or their order.

TasteNet:
TasteNet (Mottini and Acuna-Agost, 2017) applies attention mechanisms and pointer networks to choice prediction. The model encodes each alternative into a representation $h_i = \text{encoder}(x_i)$ and uses attention weights $a_{ij} = \text{attention}(h_i, h_j)$ to capture dependencies between alternatives. This architecture can represent some context effects through cross-alternative attention.

The attention mechanism provides some interpretability by showing which alternatives the model considers together when making predictions. TasteNet handles variable-sized choice sets naturally and has achieved state-of-the-art performance in some domains such as airline itinerary prediction. However, attention mechanisms do not guarantee true context-dependent

utility in the choice-theoretic sense. The relationship between attention weights and utility is indirect, and the model may violate standard choice axioms. Furthermore, TasteNet's complex architecture involves numerous hyperparameters requiring extensive tuning.

TCNet:
TCNet (Wang et al., 2023) applies transformer architectures to choice prediction, using self-attention to jointly encode the choice set. Transformers provide powerful representation learning through multiple layers of self-attention, enabling the model to capture complex dependencies between alternatives. The architecture is inherently permutation-invariant and handles variable choice set sizes naturally.

While transformers are powerful, they suffer from significant drawbacks for discrete choice modeling. Transformers are heavily overparameterized, typically requiring millions of observations to avoid overfitting. The architecture is completely opaque, providing no interpretability regarding interaction structure. Computational costs are high due to the quadratic complexity of attention mechanisms. The model may violate fundamental choice axioms, and there are no theoretical guarantees about what patterns can be represented. For credit card applications where datasets typically contain thousands rather than millions of choices and interpretability is valued, transformers represent overkill.

3.3 Context-Dependent Choice Models

RKHS Choice Model:
The Reproducing Kernel Hilbert Space (RKHS) choice model (Yang et al., 2025) takes a non-parametric approach to context-dependent utility. The model represents utility as $U_i(S) = f(x_i, S)$ where f belongs to a reproducing kernel Hilbert space. This allows utility to depend on both the alternative's features and the composition of the choice set. The kernel function $k((x, S), (x', S'))$ captures similarity between alternatives in different choice sets.

Estimation proceeds via kernel ridge regression, which provides a convex optimization problem with a unique global optimum. The approach has solid theoretical foundations including universal approximation within the RKHS and consistency guarantees. The kernel framework allows incorporation of prior knowledge about relevant similarity metrics.

The main challenge is kernel selection. The researcher must specify an appropriate kernel function capturing the relevant structure of context-dependence, which requires substantial domain knowledge. Poor kernel choice leads to poor model performance, and there is limited guidance on kernel selection for complex problems. Additionally, computational complexity scales as $O(n^3)$ for n observations due to the need to invert kernel matrices. The approach works well for moderate-sized problems (fewer than 50,000 observations) when appropriate kernels can be specified but becomes impractical for larger applications.

Low-Rank Halo MNL:
Low-Rank Halo MNL (Jagabathula and Rusmevichientong, 2017) introduces context-dependence through "halo" parameters capturing pairwise interactions between alternatives. The utility specification is $V_i(S) = \theta_i + \sum_{k \neq i} h(x_i, x_k)$ where the halo function $h(x, y)$ represents the effect of alternative y being in the choice set on the attractiveness of alternative x. To maintain tractability, the model assumes a low-rank structure: $h(x, y) = \sum_{r=1}^{R} u_r(x) v_r(y)$.

This dramatically reduces the number of parameters while still capturing first-order context effects. The halo parameters have clear economic meaning, representing how alternatives affect each other's attractiveness. Estimation is computationally efficient using alternating minimization or gradient-based methods. The model has proven successful in various applications including assortment optimization.

However, the restriction to pairwise interactions is limiting. The model can capture how adding a high-APR card affects the attractiveness of a medium-APR card, but it cannot represent three-way interactions like the joint effect of having both high-APR and low-rewards

cards in the choice set simultaneously. For credit cards, where utility may involve complex joint effects of APR, rewards, fees, and limits, this limitation is significant.

DeepHalo: Context-Dependent Deep Choice Model:
The DeepHalo framework (**?**) combines the strengths of halo models and deep learning while addressing their respective limitations. The model extends the halo structure to arbitrary interaction orders through deep residual networks. Formally, context-dependent utility is specified as

$$V_i(S) = g_L\left(\sum_{j\in S} f_L(x_i, x_j)\right) \tag{1}$$

where f_L and g_L are deep neural networks built with residual blocks.

The architecture employs residual blocks of the form

$$z^\ell = W^\ell \sigma(z^{\ell-1}) + z^{\ell-1} \tag{2}$$

where σ is an activation function such as quadratic ($\sigma(z) = z^2$) or exponential. The key innovation is the explicit relationship between network depth and interaction order: a network with L layers can represent interactions up to order 2^{L-1}, providing exponential growth in expressiveness with each additional layer.

The choice of activation function affects the interaction structure. Quadratic activations produce polynomial interactions, enabling the model to represent utilities that are polynomial functions of the features and choice set composition. The architecture is designed to be permutation-equivariant, meaning that reordering the items in the choice set does not affect the output beyond a corresponding reordering of the probabilities.

DeepHalo provides several theoretical guarantees. First, it achieves universal approximation for context-dependent choice functions (**?**). Any continuous context-dependent utility function can be approximated arbitrarily well with sufficient depth and width. Second, the approximation rate is exponential in depth: adding one layer doubles the interaction order, so only $O(\log K)$ layers are needed to represent K-way interactions. Third, under appropriate conditions including sufficient sample size and regularity conditions on the true utility, the model parameters are identifiable from choice data.

These theoretical results are complemented by empirical validation. Experiments on synthetic data with known ground truth demonstrate that DeepHalo successfully recovers the true utility structure when the depth is set appropriately for the interaction order. The model also demonstrates strong out-of-sample performance on real choice datasets.

From a practical perspective, DeepHalo offers significant advantages. Depth 5 is typically sufficient for most applications, as it captures up to 16th-order interactions. This is far more efficient than manually specifying high-order terms or using black-box deep networks with unclear interaction structures. The model can be estimated using standard gradient descent (e.g., Adam optimizer) without special algorithms. Both feature-based and featureless variants are available: the feature-based version uses observed card attributes while the featureless version learns representations from card identifiers. The architecture scales well, having been tested on datasets with over one million observations.

3.4 Summary Comparison

Table 1 provides a systematic comparison of the reviewed models across key dimensions relevant to credit card applications.

Classical models provide high interpretability and strong theoretical foundations but lack the flexibility needed for context-dependent choice with complex interactions. Modern deep learning approaches offer flexibility but lack explicit context mechanisms and theoretical transparency. Context-dependent models provide the right framework, but RKHS models have scalability

Table 1: Comparison of Discrete Choice Models for Credit Card Applications

Model	Context	Interaction	Interpret.	Scalability	Theory	Suitability
MNL	No	Manual	High	Excellent	Strong	Low
Nested Logit	Limited	Manual	High	Excellent	Strong	Medium
Mixed Logit	No	Manual	Medium	Poor	Strong	Medium-High
RUM-nets	No	Automatic	Low	Good	Weak	Medium-High
TasteNet	Implicit	Automatic	Medium	Good	Weak	Medium
TCNet	Implicit	Automatic	Low	Good	Weak	Low-Medium
RKHS	Yes	Via kernel	Medium	Medium	Strong	Medium-High
Halo MNL	Yes (1st)	Pairwise	High	Good	Strong	Medium-High
DeepHalo	**Yes (all)**	2^{L-1}	**Medium-High**	**Good**	**Strong**	**Very High**

issues and require kernel specification, while Halo MNL is restricted to pairwise interactions. DeepHalo uniquely combines explicit context-dependence, controllable high-order interactions, strong theoretical properties, and practical tractability.

4 Technical Analysis of Zhang et al. (2025)

This section identifies ambiguities and unclear aspects of the DeepHalo paper based on our implementation experience.

4.1 Permutation Equivariance Implementation Details

Equation (7) in Zhang et al. (2025) defines the residual update as

$$z_j^{(\ell)} = \Phi_h^{(\ell)} \left(\bar{z}_h^{(\ell)},\ \phi_h^{(\ell)}(z_j^{(0)}) \right) \tag{3}$$

A subtle but critically important detail is that the nonlinear transformation $\phi_h^{(\ell)}$ is applied to the *base embedding* $z_j^{(0)}$ rather than the previous layer output $z_j^{(\ell-1)}$. This architectural choice is essential for maintaining permutation equivariance.

Using $z_j^{(0)}$ anchors all interaction operations to the item's original identity vector. This ensures that the aggregation operation $\sum_{j \in S}$ commutes with the nonlinear transformations, preserving the permutation equivariance property. If a practitioner naïvely applies $\phi_h^{(\ell)}(z_j^{(\ell-1)})$ instead, the equivariance guarantee collapses and the theoretical results in the paper no longer hold.

The paper does not sufficiently emphasize this implementation constraint. The notation in Equation (7) is clear upon careful reading, but the critical importance of using $z_j^{(0)}$ rather than $z_j^{(\ell-1)}$ is not discussed in the main text or implementation notes. This makes it easy for practitioners to misinterpret the architecture and implement a variant that violates the theoretical guarantees.

4.2 Model Width Specification

The main text of Zhang et al. (2025) suggests that the model operates in a vector space of dimension equal to the universe size $J = 20$. However, Appendix B.1 and the reference PyTorch implementation reveal that the effective width J' is substantially larger than the universe size.

For example, under the 500k parameter budget with depth 5, the actual width used is $J' = 348$ rather than $J = 20$. Using width $J' = 20$ dramatically reduces the model's capacity to represent high-order interactions, leading to underfitting and poor synthetic RMSE that fails to match the reported results.

The formula for parameter counting is

$$\text{Parameters} = (L-1)(J'^2 + J') + 2JJ' \tag{4}$$

where the first term accounts for the residual blocks and the second term accounts for the input and output layers. For the 500k budget with $L = 5$, this gives

$$500{,}000 \approx 4(J'^2 + J') + 40J' \tag{5}$$

which solves to $J' \approx 348$.

The paper should more clearly state that width expansion is necessary for the model to achieve the reported performance. Without this clarification, practitioners attempting to replicate the results will likely use $J' = J$ and obtain poor performance.

4.3 Ambiguity in Block Definitions

The reference PyTorch implementation contains unused layers (`qua_linear1`, `qua_linear2`) that are defined in the `__init__` method but never used in the forward pass. These appear to be remnants of an earlier design iteration. Their presence is confusing and could mislead practitioners trying to understand the architecture. A cleaned-up reference implementation would improve reproducibility.

5 Replication of Synthetic Data Experiments

We now describe our replication of the synthetic data experiments reported in Section 5.1 of Zhang et al. (2025).

5.1 Experimental Setup

We follow the experimental protocol exactly as described:

- Universe of $J = 20$ items
- Fixed choice set size of 15 items
- All $\binom{20}{15} = 15{,}504$ possible choice sets
- Probability distributions drawn from Dirichlet($\mathbf{1}_{15}$) for each choice set
- 80 training samples and 20 test samples per choice set
- Total of 1,240,320 training observations and 310,080 test observations
- Model depths $L \in \{3, 4, 5, 6, 7\}$
- Parameter budgets of 200k and 500k
- Random seed set to 20 for reproducibility

For each depth-budget combination, we select the width J' such that the total parameter count is as close as possible to the budget (within $\pm 0.3\%$).

5.2 Implementation Summary

Our TensorFlow implementation follows the architecture:

$$z^{(1)} = W_{\text{in}} x \tag{6}$$

$$z^{(\ell)} = W_\ell (z^{(\ell-1)})^{\odot 2} + z^{(\ell-1)}, \quad \ell = 2, \ldots, L \tag{7}$$

$$\text{logits} = W_{\text{out}} z^{(L)} \tag{8}$$

where $\odot$ denotes element-wise multiplication (Hadamard product), implementing the quadratic activation.

Masking is applied before the softmax to ensure that items not in the presented choice set receive zero probability:

$$\tilde{v}_j = \begin{cases} v_j & \text{if } j \in S \\ -\infty & \text{if } j \notin S \end{cases} \tag{9}$$

where v_j are the raw logits and $\tilde{v}_j$ are the masked logits used for computing probabilities.

Training uses the Adam optimizer with learning rate 10^{-4}, batch size 1024, and early stopping with patience 50 epochs. The loss function is negative log-likelihood (equivalently, cross-entropy loss).

5.3 Results

Our replication successfully reproduces the qualitative and quantitative results reported in the paper:

- RMSE decreases monotonically with depth until depth 5, after which improvements plateau
- This matches the theoretical prediction that depth $\log_2(15) + 1 = 4.9 \approx 5$ is sufficient
- Final test RMSE under the 500k parameter budget is approximately 0.014, matching the paper
- The frequency-adjusted RMSE (computed by averaging Y values for each unique choice set before evaluation) also matches the paper's reported values
- Training converges within 500 epochs for all configurations

The results confirm that our implementation is correct and that the DeepHalo architecture performs as theoretically predicted.

6 Additional Verification Tests

Beyond replicating the paper's experiments, we developed several additional tests to verify implementation correctness.

6.1 Permutation Equivariance Test

We verify that the model satisfies permutation equivariance: for any permutation π of item indices,

$$f(\pi(x)) = \pi(f(x)) \tag{10}$$

where f denotes the model's output probabilities.

Implementation: We generate random choice sets and input vectors, compute the output probabilities, permute the inputs, compute probabilities again, and verify that the permuted outputs match the original outputs (up to the same permutation). This test passes with numerical tolerance 10^{-6}.

6.2 Interaction Order Growth Test

Given depth L, the model should be capable of representing polynomial interactions up to order 2^{L-1}. We verify this by constructing synthetic utilities with known interaction orders and checking whether DeepHalo can recover them.

For example, with depth $L = 3$, we construct a utility function with a 4th-order interaction term (since $2^{3-1} = 4$) and verify that DeepHalo can fit this function accurately. With depth $L = 2$, we verify that the same function cannot be fit accurately, confirming that insufficient depth limits expressiveness.

6.3 Masking Robustness Test

For any choice set S, we must have $p(j|S) = 0$ for all $j \notin S$. We verify that:

- Probability mass sums to exactly 1.0 (within numerical precision 10^{-7})
- All probabilities for items not in the choice set are exactly 0.0
- No numerical leakage occurs through the masking mechanism

6.4 Stability Under Choice Set Perturbations

We test that the model exhibits reasonable behavior when choice sets are perturbed:

- When an item is removed from the choice set, probabilities for remaining items should sum to 1
- When a low-quality item is added, probabilities for existing items should generally decrease proportionally
- The model should not exhibit pathological behaviors like probability inversions

All verification tests pass, confirming that our implementation is robust and correct.

7 Comparison with RKHS Choice Model (Yang 2025)

Table 2 provides a detailed comparison between DeepHalo and the RKHS Choice Model.

Table 2: Comparison of DeepHalo and RKHS Choice Model

Feature	**DeepHalo (Zhang 2025)**	**RKHS Choice (Yang 2025)**
Context-Dependence	Explicit via aggregation over choice set	Explicit via kernel on choice sets
Interaction Order	Explicit control via depth: 2^{L-1}	Implicit via kernel choice
Optimization	Non-convex (gradient descent)	Convex (kernel ridge regression)
Interpretability	Medium-high: interaction order explicit	Medium: depends on kernel interpretation
Scalability	Good: $O(nL)$ per iteration	Poor: $O(n^3)$ for kernel inversion
Theoretical Guarantees	Universal approximation, explicit rates	Universal approximation in RKHS
Practical Advantages	Standard SGD training, scales to millions	Convex optimization, global optimum
Practical Disadvantages	Hyperparameter tuning (depth, width), local minima	Kernel selection critical, limited scalability

DeepHalo is preferable for large-scale applications where datasets contain hundreds of thousands or millions of observations and where explicit control over interaction order is valuable. The model scales well and can be trained efficiently using standard deep learning infrastructure.

RKHS models are more theoretically elegant due to their convex optimization formulation, which guarantees a unique global optimum and provides consistency guarantees. However, they face severe computational challenges for large datasets due to the $O(n^3)$ cost of inverting kernel matrices. Additionally, kernel selection requires substantial domain expertise and can significantly impact performance.

For credit card applications, where datasets are typically large and we want explicit control over interaction complexity, DeepHalo is the superior choice.

8 Suitability for Credit Card Demand Estimation

8.1 Why DeepHalo is Suitable

DeepHalo is well-suited for credit card demand estimation for several reasons:

Strong Context Effects. Credit card choices exhibit pronounced context effects including compromise effects, attraction effects, and halo effects. Premium cards become more attractive when luxury cards are present. Mid-tier cards benefit from the presence of both basic and premium alternatives. DeepHalo's explicit context-dependence captures these phenomena naturally.

Complex Feature Interactions. Credit card utility involves high-order interactions among features. The value proposition of a card depends on the joint configuration of APR, rewards rate, annual fee, credit limit, introductory offers, and other terms. A high annual fee might be acceptable if rewards are generous and APR is low, but unacceptable otherwise. DeepHalo's controllable interaction order enables representation of these complex dependencies.

Feature-Based Generalization. The feature-based variant of DeepHalo can generalize to new credit card products not seen during training, as long as they are characterized by the same set of features. This is valuable for forecasting demand for new product launches.

Scalability. Credit card datasets often contain millions of customer-offer interactions. DeepHalo can be trained efficiently on such datasets using minibatch stochastic gradient descent, making it practical for real-world applications.

Interpretability. While not as transparent as MNL, DeepHalo provides more interpretability than fully black-box models. The depth-order relationship allows practitioners to control and understand the complexity of interactions being modeled.

8.2 Implementation Considerations

Several practical considerations arise when applying DeepHalo to credit card data:

Data Sparsity. Not all combinations of credit cards are frequently offered together. Some choice sets may have very few observations. This can be addressed through regularization (dropout, weight decay) and by using the feature-based variant to enable generalization.

Temporal Dynamics. Consumer preferences may evolve over time due to changing economic conditions, interest rate environments, and marketing campaigns. The model should be retrained periodically, and temporal features (time trends, seasonality) might be incorporated.

Choice Set Size Heterogeneity. Different consumers face different numbers of offers. DeepHalo handles variable set sizes naturally through masking, but the distribution of set sizes in training data should roughly match the distribution in deployment.

Consumer Heterogeneity. Different consumer segments (by income, credit score, spending patterns) likely have different preferences. This can be addressed by training separate models for different segments or by incorporating consumer features into the model.

Unobserved Alternatives. Consumers may have access to credit cards not in the dataset (cards from small banks, credit unions, etc.). This creates a potential selection bias that should be considered when interpreting results.

Despite these challenges, DeepHalo represents a significant advance over traditional methods for credit card demand estimation, providing the flexibility needed to capture complex choice patterns while maintaining theoretical rigor and computational tractability.

9 Alternative Models Worth Considering

While we propose DeepHalo as the primary methodology, several alternative models merit consideration either as baselines for comparison or as alternatives if DeepHalo proves unsuitable.

9.1 Classical Econometric Models

Mixed Logit. Mixed Logit should be implemented as a strong baseline. It captures consumer heterogeneity through random coefficients and has a long history of successful applications. While it lacks context-dependence and automatic interaction learning, its interpretability and theoretical foundation make it valuable for comparison. Performance gains over Mixed Logit help quantify the value of context-dependent modeling.

Nested Logit. For credit cards, nesting by issuer or tier (basic/premium) might capture some substitution patterns. Nested Logit is computationally efficient and interpretable, making it a useful baseline for understanding the marginal value of more complex approaches.

Halo MNL. Low-Rank Halo MNL captures first-order context effects and provides high interpretability. If the dataset is not large enough to support deep learning or if interpretability is paramount, Halo MNL represents a middle ground between MNL and DeepHalo.

9.2 Modern Machine Learning Models

RUM-nets. RUM-nets provide flexible utility functions through neural networks while maintaining the RUM framework. They serve as a useful comparison point representing context-independent deep learning. If DeepHalo and RUM-nets perform similarly, this suggests context effects may not be important in the data.

Transformer-Based Models. Set Transformers or other attention-based architectures could be explored if datasets are sufficiently large (millions of observations). These models use self-attention to capture dependencies between alternatives, potentially capturing context effects. However, they lack the explicit theoretical structure of DeepHalo and are more prone to overfitting.

Ensemble Methods. In production deployment, ensembling multiple models (e.g., Mixed Logit, Halo MNL, DeepHalo) often improves robustness and performance. Different models may capture different aspects of choice behavior, and ensemble methods can combine their strengths.

9.3 Recommendation

We recommend the following modeling strategy:

1. Start with MNL and Mixed Logit as interpretable baselines
2. Implement DeepHalo as the primary proposed model

3. Implement RUM-nets to isolate the value of context-dependence
4. Compare all models on out-of-sample prediction accuracy
5. Use DeepHalo for production if it demonstrates substantial gains
6. Consider ensembling for final deployment to maximize robustness

10 Conclusion

This report has provided a comprehensive analysis of the DeepHalo architecture for context-dependent choice modeling, combining detailed technical implementation with an extensive literature review. Our work makes several contributions.

First, we identified and resolved ambiguities in the original Zhang et al. (2025) paper, particularly regarding permutation equivariance implementation and width specification. These clarifications will help practitioners correctly implement the model.

Second, we developed a complete TensorFlow implementation and successfully replicated the synthetic data experiments, confirming that the model performs as theoretically predicted. Our implementation passes multiple verification tests including permutation equivariance, interaction order growth, and masking robustness.

Third, we situated DeepHalo within the broader landscape of discrete choice models by reviewing ten major methodologies. This comparative analysis demonstrates that DeepHalo uniquely combines explicit context-dependence, controllable high-order interactions, strong theoretical guarantees, and computational tractability.

Fourth, we evaluated DeepHalo's suitability for credit card demand estimation and found it well-matched to the problem's characteristics including strong context effects, complex feature interactions, and large-scale data. We also identified practical considerations for deployment and recommended alternative models for comparison.

Based on comprehensive analysis of theoretical properties, empirical validation, and practical considerations, we conclude that DeepHalo represents the optimal methodology for credit card offer modeling. The implementation is ready for application to real-world data, and we expect it to provide substantial improvements over traditional approaches while maintaining interpretability and theoretical rigor.

References

Aouad, A. and Désir, A. (2025). Representing random utility choice models with neural networks. *Management Science*.

Berry, S. T., Levinsohn, J. A., and Pakes, A. (1993). Automobile prices in market equilibrium: Part i and ii.

Huber, J., Payne, J. W., and Puto, C. (1982). Adding asymmetrically dominated alternatives: Violations of regularity and the similarity hypothesis. *Journal of consumer research*, 9(1):90–98.

Jagabathula, S. and Rusmevichientong, P. (2017). A nonparametric joint assortment and price choice model. *Management Science*, 63(9):3128–3145.

McFadden, D. (1977). Modelling the choice of residential location.

Mottini, A. and Acuna-Agost, R. (2017). Deep choice model using pointer networks for airline itinerary prediction. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1575–1583.

Park, S. and Gupta, S. (2009). Simulated maximum likelihood estimator for the random coefficient logit model using aggregate data. *Journal of Marketing Research*, 46(4):531–542.

Petrin, A. and Train, K. (2010). A control function approach to endogeneity in consumer choice models. *Journal of marketing research*, 47(1):3–13.

Rossi, P. E. and Allenby, G. M. (2003). Bayesian statistics and marketing. *Marketing Science*, 22(3):304–328.

Simonson, I. (1989). Choice based on reasons: The case of attraction and compromise effects. *Journal of consumer research*, 16(2):158–174.

Train, K. E. (2009). *Discrete choice methods with simulation*. Cambridge university press.

Wang, H., Li, X., and Talluri, K. (2023). Transformer choice net: A transformer neural network for choice prediction. *arXiv preprint arXiv:2310.08716*.

Yang, Y., Wang, Z., Gao, R., and Li, S. (2025). Reproducing kernel hilbert space choice model. *Available at SSRN 5267975*.