Discrete Choice Models for Credit Card Offers

Amirhossein Javaheri

Dec. 2025

Introduction

Discrete choice models represent a fundamental framework in economics, marketing, operations research, and related fields for understanding and predicting how individuals make choices among a finite set of alternatives. These models are essential for analyzing consumer behavior, estimating demand functions, and optimizing product assortments.

A discrete choice model describes how a decision-maker selects one alternative from a finite choice set $S \subseteq \mathcal{S}$. A decision-maker i confronted with a choice set $S_i \subseteq \mathcal{S}$ selects an alternative $j \in S_i$ based on an underlying utility-maximization principle.

There have been significant methodological advances to discrete choice modeling, ranging from classical econometric approaches to modern deep learning techniques. This report reviews the primary methodologies for solving discrete choice problems, categorizing them into foundational random utility maximization (RUM) frameworks, classical econometric specifications, advanced approaches for handling endogeneity and market-level phenomena, and modern neural network-based methods. We emphasize both the theoretical foundations and practical implementation considerations for each approach.

Methodologically, the field spans from classical parametric random utility models (RUM) to modern neural-network-based architectures. This review organizes methods into: (i) foundational RUM and classical logit-type models; (ii) latent class and interaction-augmented MNL variants such as Halo MNL and its low-rank generalization; (iii) neural models including RUMnet, TasteNet, Learning-MNL, and ResLogit; and (iv) related neural and kernel methods for context-dependent choice models.

Throughout, individual-level features (covariates) are denoted by $\mathbf{x}_i \in \mathbb{R}^{d_i}$, alternative features by $\mathbf{x}_j \in \mathbb{R}^{d_x}$ (if fixed for different choice situations) or $\mathbf{x}_{ij} \in \mathbb{R}^{d_x}$ (if they vary with choices), parameter vectors $\boldsymbol{\alpha} \in \mathbb{R}^{d_i}$, $\boldsymbol{\beta}, \boldsymbol{\gamma} \in \mathbb{R}^{d_x}$, and random shocks (errors) by ε_{ij}. The total number of products or items is also denoted with $J = |\mathcal{S}|$.

Random Utility Maximization Framework

Under the RUM method [Hess et al., 2018], the utility that individual i gains from alternative $j \in S_i$, denoted as $u_{ij} := u_j(S_i)$, is

$$u_{ij} = v_{ij} + \varepsilon_{ij}, \tag{1}$$

where v_{ij} is the deterministic (systematic) component and ε_{ij} is the random component capturing unobserved heterogeneity. Individual i chooses

$$j^* = \underset{j \in S_i}{\mathsf{argmax}}\, u_{ij}. \tag{2}$$

The choice probability is

$$\mathbb{P}(i \text{ chooses } j \mid S_i) = \mathbb{P}\left(u_{ij} \geq u_{ik},\ \forall k \in S_i\right). \tag{3}$$

Identifying assumptions concern the distribution of ϵ_{ij} and the stability of the distribution of preferences across choice contexts. Classical results characterize when observed choice probabilities are rationalizable by some RUM, through, e.g., block-Marschak inequalities [Block, 1974] and revealed stochastic preference axioms.

Classical Logit-Type Models

Multinomial Logit (MNL)

In the standard MNL model McFadden [1974], ϵ_{ij}s are i.i.d. Type-I Extreme Value (Gumbel) errors. The systematic utility is typically

$$v_{ij} = \boldsymbol{\beta}^\top \mathbf{x}_{ij}, \tag{4}$$

for some parameter vector $\boldsymbol{\beta}$. Then the choice probability is

$$\mathbb{P}(i \text{ chooses } j \mid S_i) = \frac{\exp(v_{ij})}{\sum_{k \in S_i} \exp(v_{ik})}. \tag{5}$$

The MNL model is computationally convenient and yields closed-form likelihood, but it implies the independence of irrelevant alternatives (IIA) property: for any $j, k \in S_i$,

$$\frac{\mathbb{P}(j \mid S_i)}{\mathbb{P}(k \mid S_i)} = \frac{\exp(v_{ij})}{\exp(v_{ik})}, \tag{6}$$

which does not depend on other alternatives in S_i and can be too restrictive in practice.

Conditional Logit

The conditional logit formulation emphasizes alternative attributes [Train et al., 1987, Train, 2009]. A typical specification is

$$v_{ij} = \boldsymbol{\alpha}^\top \mathbf{x}_i + \boldsymbol{\beta}^\top \mathbf{x}_j + \boldsymbol{\gamma}^\top \mathbf{x}_{ij}, \tag{7}$$

where $\mathbf{x}_i$ contains individual-specific features, $\mathbf{x}_j$ fixed item-specific features, and $\mathbf{x}_{ij}$ (item-individual) interactions. The probability, however, retains the classical MNL form. Train et al. [1987] also present an application to local telephone service and calling patterns, illustrating full discrete models of service choice.

Nested Logit

Nested logit relaxes IIA by partitioning alternatives into nests $\{\mathcal{G}_g\}_{g=1}^{G}$ [McFadden, 1978]. For alternative j in nest g, the utility often takes

$$v_{ij} = \boldsymbol{\beta}^\top \mathbf{x}_{ij}, \quad j \in \mathcal{G}_g, \tag{8}$$

and the choice probability decomposes as

$$\mathbb{P}(j \mid S_i) = \mathbb{P}(j \mid g, S_i)\, \mathbb{P}(g \mid S_i). \tag{9}$$

The within-nest probability is

$$\mathbb{P}(j \mid g, S_i) = \frac{\exp\left(\frac{1}{\lambda_g} v_{ij}\right)}{\sum_{k \in \mathcal{G}_g \cap S_i} \exp\left(\frac{1}{\lambda_g} v_{ik}\right)}, \tag{10}$$

and the nest probability is

$$\mathbb{P}(g \mid S_i) = \frac{\exp\left(\lambda_g \, \mathrm{IV}_{ig}\right)}{\sum_h \exp\left(\lambda_h \, \mathrm{IV}_{ih}\right)}, \tag{11}$$

with inclusive value

$$\mathrm{IV}_{ig} = \log \sum_{k \in \mathcal{G}_g \cap S_i} \exp\left(\frac{1}{\lambda_g} v_{ik}\right), \tag{12}$$

and $\lambda_g \in (0, 1]$ controlling within-nest correlation. McFadden [1978] and Forinash and Koppelman [1993] discuss nested logit for residential location and intercity mode choice.

Latent Class MNL

Latent Class MNL (LC-MNL) models preference heterogeneity via a discrete mixture of MNL components [Greene and Hensher, 2003]. Let $c \in \{1, \ldots, C\}$ index classes, with class-specific parameters $\boldsymbol{\beta}^{(c)}$. The class-conditional probability is

$$\mathbb{P}(j \mid S_i, c) = \frac{\exp\left((\boldsymbol{\beta}^{(c)})^\top \mathbf{x}_{ij}\right)}{\sum_{k \in S_i} \exp\left((\boldsymbol{\beta}^{(c)})^\top \mathbf{x}_{ik}\right)}. \tag{13}$$

Class membership is modeled as

$$\mathbb{P}(c \mid \mathbf{z}_i) = \frac{\exp\left(\boldsymbol{\gamma}_c^\top \mathbf{z}_i\right)}{\sum_{c'=1}^{C} \exp\left(\boldsymbol{\gamma}_{c'}^\top \mathbf{z}_i\right)}, \tag{14}$$

where $\mathbf{z}_i$ captures individual covariates. The overall probability marginalizes over classes:

$$\mathbb{P}(j \mid S_i) = \sum_{c=1}^{C} \mathbb{P}(c \mid \mathbf{z}_i)\, \mathbb{P}(j \mid S_i, c). \tag{15}$$

Model parameters $\{\boldsymbol{\beta}^{(c)}, \boldsymbol{\gamma}_c\}_{c=1}^{C}$ are typically estimated via EM or direct maximum likelihood.

Interaction-Augmented Models

Halo MNL

Halo MNL [Maragheh et al., 2018] augments MNL to capture pairwise interaction (halo) effects among products. For $j \in S_i$, the deterministic utility is

$$v_{ij} = \boldsymbol{\beta}^\top \mathbf{x}_{ij} + \sum_{k \in S_i,\, k \neq j} h_{jk}, \tag{16}$$

where h_{jk} quantifies the (possibly asymmetric) effect of offering product k on the attractiveness of j. The probability retains the MNL form. In matrix notation, let $\mathbf{H} \in \mathbb{R}^{J \times J}$ with $(\mathbf{H})_{jk} = h_{jk}$ and zeros on the diagonal (J is the total number of items). Then, for a given choice set, interactions correspond to row sums of $\mathbf{H}$ restricted to S_i.

Parameter identifiability requires structural constraints on $\mathbf{H}$, and the total number of interaction parameters is $O(J^2)$ for J products, implying $\Omega(J^2)$ samples for reliable estimation. Halo MNL captures complementarities and substitution patterns but can overfit in large assortments.

Low-Rank Halo / Self-Attention Choice Models

Ko and Li [2024] show that Halo MNL can be generalized via a low-rank factorization

$$h_{jk} = \mathbf{u}_j^\top \mathbf{v}_k, \quad \mathbf{u}_j, \mathbf{v}_k \in \mathbb{R}^r,\ r \ll J, \tag{17}$$

dramatically reducing the number of free parameters from $O(J^2)$ to $O(Jr)$. They implement this with self-attention:

$$\mathrm{Att}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{r}}\right)\mathbf{V}, \tag{18}$$

where rows of $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ are learned embeddings of alternatives. The resulting utility for j becomes

$$v_{ij} = \boldsymbol{\beta}^\top \mathbf{x}_{ij} + f_{\mathrm{att}}(j, S_i; \boldsymbol{\Theta}), \tag{19}$$

with f_{att} parameterized by attention weights. Ko and Li [2024] provide theoretical guarantees: under mild conditions, their self-attention estimator achieves near-optimal stationary points with $O(J)$ samples, a polynomial improvement over Halo MNL. This yields a scalable interaction-aware logit model for large assortments.

Neural Discrete Choice Models

RUMnet

RUMnet [Aouad and Désir, 2023] represents RUM-consistent choice models with neural networks. For each alternative j, consider T samples of latent utility:

$$u_{ij}^{(t)} = f_j(\mathbf{x}_i; \boldsymbol{\theta}^{(t)}) + \varepsilon_{ij}^{(t)}, \quad t = 1, \dots, T, \tag{20}$$

with f_j parameterized by neural networks and $\boldsymbol{\theta}^{(t)}$ representing sampled network parameters or input noise. The choice probability is approximated via sample average:

$$\mathbb{P}(j \mid S_i) \approx \frac{1}{T} \sum_{t=1}^{T} \mathbb{1}\Big(u_{ij}^{(t)} \geq u_{ik}^{(t)},\ \forall k \in S_i\Big). \tag{21}$$

Aouad and Désir [2023] prove that RUMnets can approximate any RUM-derived choice model arbitrarily well with sufficient capacity and samples, and that any RUMnet is rationalizable by some RUM. They also derive generalization bounds depending on $\|\boldsymbol{\theta}\|$, sample size, and architecture depth/width.

TasteNet-MNL

TasteNet [Han et al., 2022] embeds a neural network into MNL to learn taste heterogeneity. Let $\mathbf{z}_i$ denote socio-demographic features. TasteNet parametrizes individual-level coefficients as

$$\boldsymbol{\beta}_i = g(\mathbf{z}_i; \boldsymbol{\theta}_{\mathrm{taste}}), \tag{22}$$

where g is a feed-forward network. The utility is

$$v_{ij} = \boldsymbol{\beta}_i^\top \mathbf{x}_j, \tag{23}$$

and the choice probability is standard MNL. The key idea is to model $\boldsymbol{\beta}_i$ flexibly as a nonlinear function of $\mathbf{z}_i$, capturing rich heterogeneity while retaining interpretable $\boldsymbol{\beta}_i$ as marginal sensitivities.

On synthetic data, TasteNet-MNL recovers ground-truth nonlinear mappings between $\mathbf{z}_i$ and $\boldsymbol{\beta}_i$. On the Swissmetro dataset [Bierlaire, 2001], it outperforms classic MNL while revealing more realistic value-of-time distributions.

Learning-MNL

Learning-MNL [Sifringer et al., 2020] decomposes systematic utility into a knowledge-driven and a learned component:

$$v_{ij} = v_{ij}^{\mathrm{KN}} + v_{ij}^{\mathrm{NN}}, \tag{24}$$

with

$$v_{ij}^{\mathrm{KN}} = \boldsymbol{\beta}^\top \mathbf{x}_{ij}, \quad v_{ij}^{\mathrm{NN}} = h(\mathbf{x}_{ij}; \boldsymbol{\theta}_{\mathrm{rep}}), \tag{25}$$

and h implemented by a neural network. The choice probability is MNL with v_{ij} above. This structure leverages domain knowledge via V^{KN} while allowing V^{NN} to correct misspecification. Estimation is via joint maximum likelihood over $(\boldsymbol{\beta}, \boldsymbol{\theta}_{\mathrm{rep}})$, with regularization on $\boldsymbol{\theta}_{\mathrm{rep}}$ to mitigate overfitting and preserve interpretability of $\boldsymbol{\beta}$.

ResLogit

ResLogit [Wong and Farooq, 2021] uses residual neural networks to model v_{ij}:

$$\mathbf{v}_{ij}^{(0)} = \mathbf{x}_{ij}, \quad \mathbf{v}_{ij}^{(\ell+1)} = \mathbf{v}_{ij}^{(\ell)} + F^{(\ell)}(\mathbf{v}_{ij}^{(\ell)}; \boldsymbol{\theta}^{(\ell)}), \tag{26}$$

for $\ell = 0, \dots, L-1$, and

$$v_{ij} = \mathbf{w}^\top \mathbf{v}_{ij}^{(L)}. \tag{27}$$

Residual mappings $F^{(\ell)}$ are typically shallow networks. The ResNet structure facilitates training deep architectures by preserving gradient flow. As in MNL,

$$\mathbb{P}(j \mid S_i) = \frac{\exp(v_{ij})}{\sum_{k \in S_i} \exp(v_{ik})}. \tag{28}$$

Wong and Farooq [2021] empirically show that ResLogit outperforms shallower NN-logit and classical models on transport datasets, especially when utility is highly nonlinear in

$\mathbf{x}_{ij}$.

Context-Dependent Models

Deep Context-Dependent and RKHS Choice Models

Deep context-dependent choice models [Zhang et al., 2025] decompose the systematic utility of j as

$$v_{ij} = v_j(S_i) = v_j + \sum_{T \subseteq S_i \setminus j} v_j(T), \tag{29}$$

where v_j and $v_j(T)$ are parameterized via permutation-equivariant neural networks. This allows explicit control over the order of interaction (pairwise, triplets, etc.) and preserves exchangeability over alternatives.

RKHS-based models [Yang et al., 2025] embed $\mathbf{X}_{S_i} = (\mathbf{x}_i, \mathbf{x}_{ij})$ into a reproducing kernel Hilbert space $(\mathcal{H}, \langle \cdot, \cdot \rangle_{\mathcal{H}})$ with kernel $\bar{\mathbf{K}}(\cdot, \cdot)$, and represent systematic utility as

$$v_{ij} = \mathbf{e}_j^\top \sum_{n=1}^{N} \alpha_n \bar{\mathbf{K}}\big((\mathbf{X}_{S_i}, S_i), (\mathbf{X}_{S_{in}}, S_{in})\big), \tag{30}$$

with coefficients $\{\alpha_n\}$. Regularization in $\mathcal{H}$ (e.g. via $\|\mathbf{v}\|_{\mathcal{H}}^2$) controls complexity and mitigates overfitting.

Sparse Market-Product Shocks and Endogeneity

When analyzing market-level aggregated data, unobserved product characteristics ε_{ij} can be correlated with prices, creating endogeneity bias. Berry et al. [1995] pioneered the BLP approach using instrumental variables. Recent work by Lu and Shimizu [2025] develops methods exploiting sparsity in market-product shocks, providing alternatives to instrumental variable assumptions when exogenous instruments are weak or unavailable.

Based on this literature review, we will now address the questions posed in the credit card offer demand estimation challenge. The corresponding answers are provided in the sections below, each aligned with the letter of the original question.

a Errors or Unclear Parts in [Zhang et al., 2025]

The following is a list of typos, mathematical and structural errors, and unclear parts in the paper:

- In section 2.1, when the authors define $P_j(S)$, they say that $u_j(T)$ represents the utility of item j when the context subset is $T \subseteq S \setminus \{j\}$ (and hence, T should not contain j. But later in the definition of $P_j(S)$, they say that $j \in S$ and they use the notation $u_j(S)$ which is a little bit confusing. The correct form is to say $u_j(T)$ represents the utility of item j in set $T = \{j, T'\} \mid T' \subseteq S \backslash \{j\}$.

- In section 3, in the first paragraph it is told that $\mathbf{x}_j \in \mathbb{R}^{d_x}$, i.e., each item (alternative) has d_x-dimensional feature. However, later d_x is mixed with d, like for example in the definition of $\mathbf{X}_S$ which should be of size $d_x \times J$ not $d \times J$.

- In equation (3), the notation used for $u_j(\mathbf{x}_j, \mathbf{X}_R)$ should be $u_j(\mathbf{X}_{\{j\}\cup R})$ to be consistent with the notation in equation (2). Moreover, the representation seems a little bit different from that given in equation (2). In that equation, $v_j(\phi)$ represents the intrinsic utility of alternative j, while in equation (3) $v_j(\mathbf{x}_j)$ represents the same value. Additionally, the notation $\subset$ should be replaced with $\subseteq$.

- The authors' proposal to capture higher-order context effects via the neural network design in equations (4) and (5) is less explicit than the feature concatenation approach in Pfannschmidt et al. [2022]. However, their approach is more computationally feasible; as with increasing network depth (interaction order), the size of the network will exponentially increase with feature concatenation.

- On page 4, column 2, nonlinear embedding function should be $\chi : \mathbb{R}^{d_x} \to \mathbb{R}^d$ (replacing d_0 with d) to be consistent with the remaining notations.

- The function Φ_h^l in equation (7) should be a polynomial function on $\mathbb{R} \times \mathbb{R}^d$ (not $\mathbb{R} \times \mathbb{R}$).

- Equation (8) is incorrect. It is unclear how setting $\sigma(\cdot)$ as a quadratic function and $\Phi_h^l(\cdot, \cdot)$ as a linear transformation of its second argument leads to this equation. Apparently, $\mathbf{z}_j^0$ should appear in equation (8). Let $\mathbf{W}^l \in \mathbb{R}^{H \times d}$. Also assume $\Phi_h^l(x, \mathbf{y}) = f(x)\mathbf{y}$. Then, according to equations (4) and (5) in Zhang et al. [2025], we have:

$$\bar{\mathbf{z}}^l = \mathbf{W}^l \left(\frac{1}{S} \sum_{k=1}^{S} \sigma(\mathbf{z}_k^{l-1}) \right), \tag{31}$$

$$\mathbf{z}_j^l = \mathbf{z}_j^{l-1} + \frac{1}{H} \sum_{h=1}^{H} f(\bar{z}_h^l) \phi^l(\mathbf{z}_j^0). \tag{32}$$

The last equation is clearly different form what is given in (8). This equation may only hold under a specific condition. For example, if $\mathbf{z}_j^0 = \mathbf{e}_j$, $\phi^l(\mathbf{z}_j^0) = \mathbf{e}_j$, $H = J$, and $f(\bar{z}_h^l) = \bar{z}_h^l$, then we get:

$$\mathbf{z}_j^l = \mathbf{z}_j^{l-1} + \frac{1}{J}\sum_{j=1}^{J} \bar{z}_j^l \mathbf{e}_j = \mathbf{z}_j^{l-1} + \frac{1}{J}\bar{\mathbf{z}}^l = \mathbf{z}_j^{l-1} + \frac{1}{S}\sum_{k=1}^{S} (\mathbf{W}^l/J)\sigma(\mathbf{z}_k^{l-1}). \tag{33}$$

Note: Comparing the last equation on page 4 of Zhang et al. [2025] with equation (5), one may think there is an error in equation (5) and $\phi_h^l(\mathbf{z}_j^0)$ should be replaced with $\phi_h^l(\mathbf{z}_j^{l-1})$. However, equation (5) is already correct. This equation states that the hidden state (variable) from previous layer $\mathbf{z}_j^{l-1}$ is combined with the initial influence of each alternative $\phi_h^l(\mathbf{z}_j^0)$, through modulation with head-specific coefficients of the context summary from the previous layer ($\bar{z}_h^l$), to produce the hidden state for the current layer $\mathbf{z}_j^l$. Hence, with $\mathbf{z}_j^1$ capturing the first order halo effects, i.e., effect of $\{k\}$, $k \in S \setminus \{j\}$ on j, then $\mathbf{z}_j^2$ captures the second order halo effects (effect of $\{k, l\}$, $k, l \in S \setminus \{j\}$ on j).

- The authors propose the following decomposition for the utility function:

$$u_j(\mathbf{X}_S) = \sum_{p=0}^{|S|-1} u_j^{(p)}(\mathbf{X}_S), \tag{34}$$

where $u_j^{(p)}(\mathbf{X}_S)$ denotes the contribution of the p-th order interactions and is defined as

$$u_j^{(p)}(\mathbf{X}_S) := \sum_{T \subseteq S\setminus\{j\}, |T|=p} v_j(\mathbf{X}_{T\cup\{j\}}). \tag{35}$$

However, the authors do not clarify how the marginal utility terms $v_j(\mathbf{X}_{T\cup\{j\}})$ are obtained. Furthermore, in the last paragraph of page 4 of Zhang et al. [2025], it is suggested that $u_j^{(p)}(\mathbf{X}_S) = \boldsymbol{\beta}^\top \mathbf{z}_j^{(p)}$. Given this and using (34), we yield:

$$u_j(\mathbf{X}_S) = \sum_{p=0}^{|S|-1} \boldsymbol{\beta}^\top \mathbf{z}_j^{(p)}. \tag{36}$$

However, later, on page 5 (after equation (5)), the final utility function is given as $u_j(\mathbf{X}_S) = \boldsymbol{\beta}^\top \mathbf{z}_j^L$, which is in contradiction with (36) (if $L = |S| - 1$). Hence, the p-th order interaction cannot be simply formulated as $u_j^{(p)}(\mathbf{X}_S) = \boldsymbol{\beta}^\top \mathbf{z}_j^{(p)}$.

Note: In our implementation, we obtain the final utility as $u_j(\mathbf{X}_S) = \boldsymbol{\beta}^\top \mathbf{z}_j^L$, where L is the last layer index.

- A utility function $u_j(\mathbf{X}_S)$ is permutation equivariant if $u_j(\mathbf{X}_{\pi(S)}) = u_{\pi(j)}(\mathbf{X}_S)$. The featured implementation of the DeepHalo method, based on equations (4) and (5) of Zhang et al. [2025], satisfies permutation equivariance. The context vector $\bar{\mathbf{z}}^l$

in equation (31) is clearly permutation equivariant as it is a linear function of a (symmetric) sum over $\mathbf{z}_k^{l-1}$. For each item, the update $\mathbf{z}_j^l$ also only depends on $\mathbf{z}_j^{l-1}$ and $\mathbf{z}_j^0$ which represent the embeddings of the same item. Hence, the featured implementation is permutation equivariant. However, the featureless implementation based on equation (10) of Zhang et al. [2025] is not permutation equivariant. To prove that, let $\mathbf{P} \in \{0,1\}^{J \times J}$ be a permutation matrix corresponding to the permutation π. We need to show that $\mathbf{u}(\mathbf{X}_S\mathbf{P}) = \mathbf{P}\mathbf{u}(\mathbf{X}_S)$. Consider equation (10) in Zhang et al. [2025] as

$$\mathbf{y}^l = \mathbf{y}^{l-1} + \mathbf{\Theta}^l \sigma(\mathbf{y}^{l-1}), \tag{37}$$

which starts from $\mathbf{y}^0(\mathbf{X}_S) = \sum_j \mathbf{x}_j = \mathbf{X}_S\mathbf{1}$. Let $\mathbf{u}(\mathbf{X}_S) = \mathbf{y}^L(\mathbf{X}_S)$. Then, we have to verify the equality $\mathbf{y}^L(\mathbf{X}_S\mathbf{P}) = \mathbf{P}\mathbf{y}^L(\mathbf{X}_S)$. It is straightforward to show that $\mathbf{y}^0(\mathbf{X}_S)$ is invariant to the permutation of the columns (items) of $\mathbf{X}_S$ as

$$\mathbf{y}^0(\mathbf{X}_S\mathbf{P}) = \mathbf{X}_S\mathbf{P}\mathbf{1} = \mathbf{X}_S\mathbf{1} = \mathbf{y}^0(\mathbf{X}_S). \tag{38}$$

Hence, $\mathbf{y}^L(\mathbf{X}_S) = \mathbf{y}^L(\mathbf{X}_S\mathbf{P})$ (since the initial point is the same in both cases). This implies invariance of the utility function to the permutation of the items, which is different from equivariance. The latter requires $\mathbf{y}^L(\mathbf{X}_S\mathbf{P}) = \mathbf{P}\mathbf{y}^L(\mathbf{X}_S)$ which does not hold.

- Calculation of the relative context effect $\alpha_{jk}(T)$ requires:

$$\alpha_{jk}(T) = [v_j(T) + v_j(T \cup \{k\})] - [v_k(T) + v_k(T \cup \{j\})]. \tag{39}$$

Each $v_j(T)$ needs $2^{|T|}$ evaluations via (3) as

$$v_j(\mathbf{X}_{T\cup\{j\}}) = \sum_{R \subseteq T} (-1)^{|T|-|R|} u_j(\mathbf{X}_{\{j\}\cup R}), \tag{40}$$

which is infeasible for large choice sets, when $|S| \gg 1$.

b Replication of Synthetic Data Test in [Zhang et al., 2025]

b.1 A Short Tutorial on Choice-Learn Python Package

The `choice-learn` Python package is a modular framework for estimating discrete choice models supporting both single and multiple choice (basket) models . This package is based on Tensorflow and is designed for optimized Data handling with the `ChoiceDataset` class and ready-to-use (as well as customized) models with the `ChoiceModel` class.

The data object instantiated by the `ChoiceDataset` class is characterized by four main tensors [Auriau et al., 2024]:

- `shared_features_by_choice`: context- or decision-maker-specific features (a matrix of shape $(n_{\text{choices}}, n_{\text{shared-features}})$).
- `items_features_by_choice`: alternative- and interaction-specific features (a matrix of shape $(n_{\text{choices}}, n_{\text{items}}, n_{\text{items-features}})$).
- `available_items_by_choice`: availability indicators for each alternative in each choice set (of shape $(n_{\text{choices}}, n_{\text{items}})$).
- `choices`: the index of the chosen alternative for each choice (of shape $(n_{\text{choices}}, n_{\text{items}})$).

On top of this data layer, the package implements a set of models built upon the base class `ChoiceModel`, which provides a common interface for specification, prediction, and learning of a choice model [Auriau et al., 2024].

The `ChoiceModel` base class provides several methods and attributes that every model (built-in or custom) is expected to use. Model specification occurs in `__init__()` and optionally `instantiate()`, where hyperparameters such as optimizer, learning rate, and number of epochs are set. Model estimation is handled by the `fit()` method, which takes a `ChoiceDataset`, computes the negative log-likelihood of the implied choice probabilities, and runs the chosen TensorFlow optimizer (by default `lbfgs` for small to medium problems, or stochastic gradient optimizers such as Adam for large-scale data) [Auriau et al., 2024]. Once trained, the model can be used via `evaluate()` (to compute average negative log-likelihood on a dataset), `predict_probas()` (to obtain choice probabilities $\hat{P}(j \mid S_i)$ for each choice situation), and `compute_batch_utility()` (to obtain the matrix of utilities for a batch of choices and items.

Custom models are created by subclassing `ChoiceModel` and implementing two main elements: (i) the initialization of trainable parameters in `__init__()`, and (ii) the utility computation in `compute_batch_utility()`, which receives four arguments: `shared_features_by_choice`, `items_features_by_choice`, `available_items_by_choice`, and `choices`. The output must be a tensor of shape $(n_{\text{choices}}, n_{\text{items}})$ containing the systematic utility u_{ij} of each item for each choice in the batch.

For example, in Train's formulation [Train, 2009] for the utility of alternative j in choice set i as

$$u_{ij} = \boldsymbol{\alpha}^\top \mathbf{x}_i + \boldsymbol{\beta}^\top \mathbf{x}_j + \boldsymbol{\gamma}^\top \mathbf{x}_{ij},$$

$\mathbf{x}_i$ denotes `shared_features_by_choice`, while the vectors $\mathbf{x}_j$ and $\mathbf{x}_{ij}$ are both represented with `items_features_by_choice`, where item-only attributes (corresponding to $\mathbf{x}_j$) are repeated across choices when they are time-invariant, and interaction attributes (corresponding to $\mathbf{x}_{ij}$) are defined at the (i, j) level as additional feature columns for each item in each choice. The fields `choices` and `available_items_by_choice` then specify, for each choice situation i, which alternatives $j \in S_i$ were available and which alternative was actually chosen.

b.2 Our Contribution

In the main formulation of the paper, the input feature matrix $\mathbf{X}_S$ is constructed as

$$\mathbf{X}_S = [\mathbf{x}_1, \ldots, \mathbf{x}_{|S|}, \mathbf{O}_{J-|S|}] \in \mathbb{R}^{d_x \times J} \tag{41}$$

where $\mathbf{O}_{J-|S|} \in \mathbb{R}^{d_x \times (J-|S|)}$ is a matrix of all zeros. In other words, the right-most columns of the matrix corresponding to the indices of the unavailable items are always padded with zeros. This requires rearranging the columns of the feature matrix (resorting $\mathcal{S} = \{1 \ldots, J\}$), so that the first $|S|$ columns represent available items features. We revised the code so that it works with fixed data features (fixed item indices) and availability mask, instead of resorting the feature matrix to have available items at the beginning and giving the lengths of the available items. In our implementation, the columns of the matrix are fixed and we instead, mask the feature matrix with the availability input vector $\boldsymbol{\mu} \in \{0, 1\}^J$. Then, equation (4) becomes:

$$\bar{\mathbf{z}}^l = \frac{1}{|\sum_j \mu_j|} \mathbf{W}^l \sum_{k=1}^{J} \mu_k \mathbf{z}_k^{l-1} \tag{42}$$

Our choice is more convenient and generalizable.

We have developed a python package, named `DeepHalo`, for the implementation of the deep context-dependent choice model of Zhang et al. [2025] in `choice-learn`. We implemented two network setups addressing the featured and featureless data scenarios (especially for the synthetic experiment in section 5.1 of the paper). The featured model can be built from the `DeepHaloFeatured` class in `Featured_DeepHalo` python module. The featureless model can also be instantiated from `DeepHaloFeatureless3D` class inside the `Featureless_DeepHalo` module. The neural network implementation in feature-based setting is based on equations (15) and (16) in Section B of [Zhang et al., 2025]. The original featureless model in, based on formulations given in Section 4.2 of [Zhang et al., 2025], is not permutation equivariant (this is discussed in the previous answer). Hence, we implemented a corrected version in the `DeepHaloFeatureless3D` model which deals with 3D one-hot encoded $\mathbf{1}_j(S) = \mathbb{1}(j \in S)$ item features. Here, instead of initial global sum pooling of features tensor as proposed in Zhang et al. [2025], we apply per-item transformations that preserve equivariance. The implementation is indeed, similar to the featured model. The difference is that here we choose initial identity map for $\mathcal{X}$

and linear head-specific transformations for $\phi_h^l(\mathbf{z}^0)$.

To implement our models (both featured and featureless versions) with `choice-learn`, we inherit from the `SimpleMNL` class in the `choice_learn.models` module. We use `Adam` as the default optimizer and we choose negative log-likelihood for the loss function (MSE can also be chosen by changing the `loss_name` attribute of the developed `DeepHaloChoiceModel` model. We pass:

- `items_features_by_choice`: 3D tensor of shape (B, J, D), where B is the batch-size,
- `available_items_by_choice`: 2D matrix of shape (B, J) for the availability mask,
- `choices`: 1D vector of shape $(B,)$ for the choices,

to the `compute_batch_utility()` method. This function is subsequently used for training and prediction in the `SimpleMNL` model.

To replicate the synthetic data results in Zhang et al. [2025], we consider the featureless model with the following experimental setup:

- Different network depth: $L \in \{3, 4, 5, 6, 7\}$,
- Different parameter budgets: budget $\in \{200000, 500000\}$,
- Different attention heads H: for each parameter budget, we find a depth-width pair (d, H) such that the number of trainable parameters of the network matches the budget (we also set the embedding dimension to $d = J$).
- Quadratic activation: similar to the synthetic data test in [Zhang et al., 2025], we apply a quadratic aggregation to the embeddings.

For data generation, we perform the following[1]:

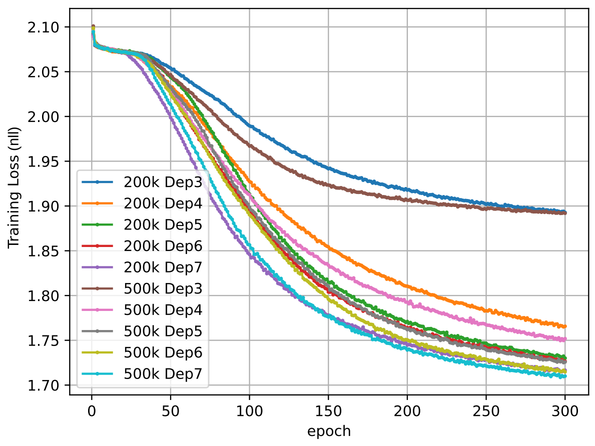
1. Choose $N = \binom{J}{S}$ items for the choice set (each with exactly S alternatives). We choose $S = 15$ and $J = 18$.

   Note: For $J = 20$ (as with the original data test in [Zhang et al., 2025]), the dataset would become excessively large, causing memory exhaustion on our system during training with `choice-learn`. Hence, we instead tested on $J = 18$ for reduced complexity.)

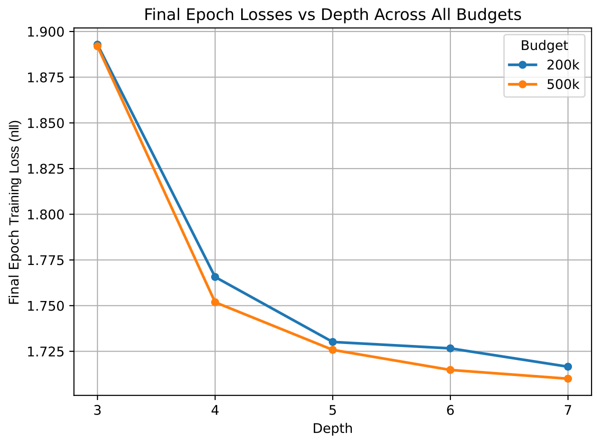
2. Sample choice probability vectors uniformly from the simplex on $\mathbb{R}^S$

3. For each choice set with S items, generate 40 i.i.d. choice observations for training and 10 i.i.d. observations for test

[1]https://github.com/Asimov-Chuang/DeepHalo



(a) Training loss (negative log-likelihood) curve across epochs for different parameter budgets



(b) Final training loss vs. network depth

Figure 1: Evaluating the effect of network depth (Halo effect order) on training loss using a synthetic choice dataset with $S = 15$ and $J = 18$.

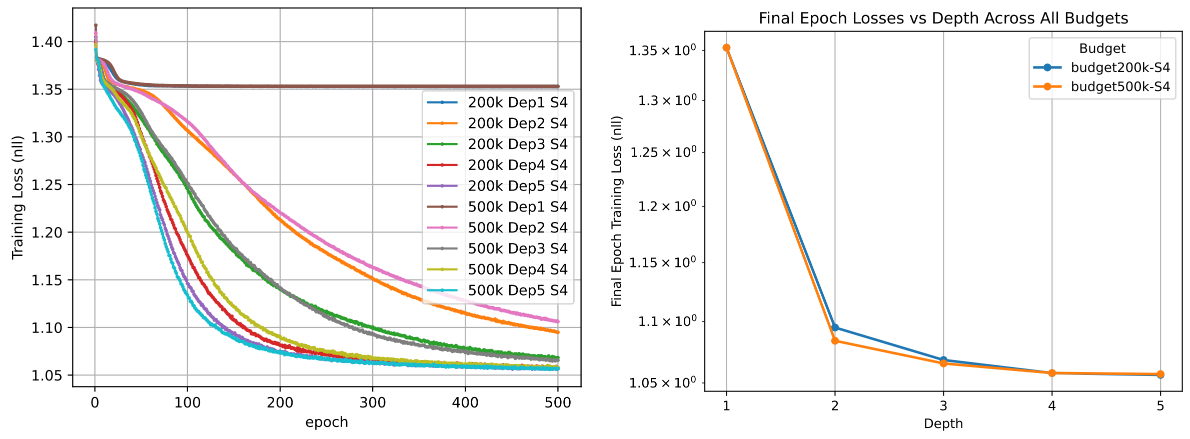
Fig. 1 shows the numerical result for the featureless model of Zhang et al. [2025] over a synthetic choice dataset with $S = 8$ alternatives in every choice set, from a universe of $J = |\mathcal{S}| = 15$ items. Data generation process is similar to the experimental setup given in Section 5.1 of [Zhang et al., 2025] for "Synthetic Data with High-order Effects". To evaluate the effect of depth on Halo order modeling, we change the depth (number of layers) of the network, while fixing the total number of parameters (budget). This is to avoid misinterpretation due to varying model complexity. We plot the negative log-likelihood loss curves over training data for different choices of the network depth $L \in \{3, \ldots, 7\}$ (with H being adjusted accordingly). As shown in this figure, the loss decreases as depth increases up to 5, beyond which 2^{5-1} exceeds the maximum interaction order $S = 15$. Hence, the results align closely with those reported in [Zhang et al., 2025].

c Additional Tests for Verification

The key result in [Zhang et al., 2025] is that with quadratic activations, an L-layer network can represent interactions up to order 2^{L-1}.

As mentioned in the previous answer, this was actually validated through the replicated experimental setup on synthetic data with $J = 18$ and $S = 15$.

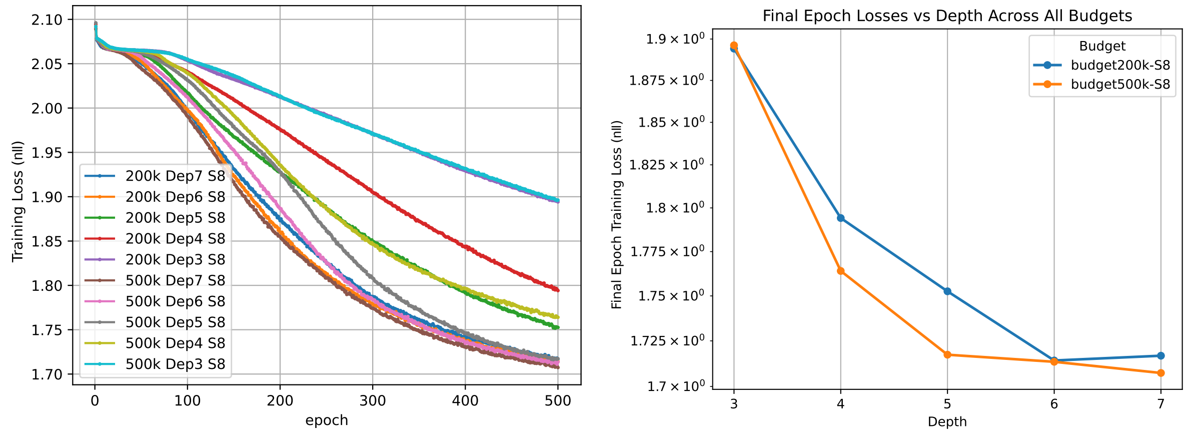
To further verify this bound, we perform additional tests on a variety of experimental setups involving both synthetic and real data described below.



(a) Training loss evolution across epochs for models with varying parameter budgets.

(b) Final training loss vs. network depth

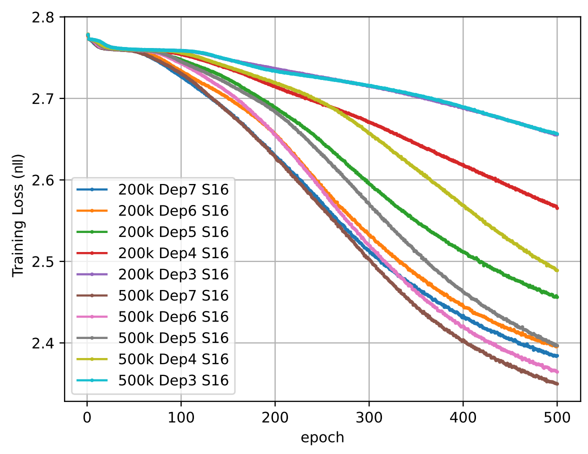
Figure 2: Evaluating the effect of network depth (Halo effect order) on training loss using a synthetic choice dataset with $(S = 4, J = 12)$. The results are based on the equivariant featureless model `DeepHaloFeatureless3D`.



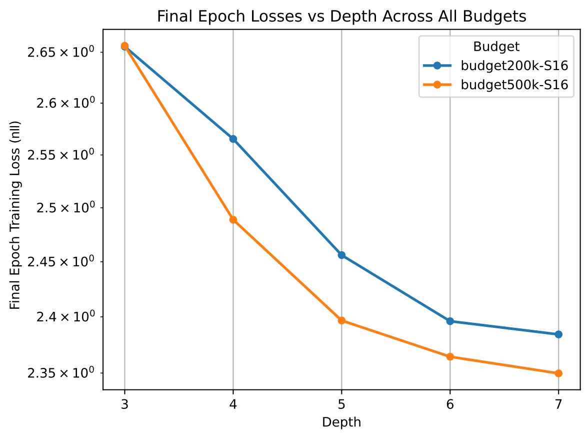
(a) Training loss evolution across epochs for models with varying parameter budgets.

(b) Final training loss vs. network depth

Figure 3: Evaluating the effect of network depth (Halo effect order) on training loss using a synthetic choice dataset with $(S = 8, J = 12)$. The results are based on the equivariant featureless model `DeepHaloFeatureless3D`.

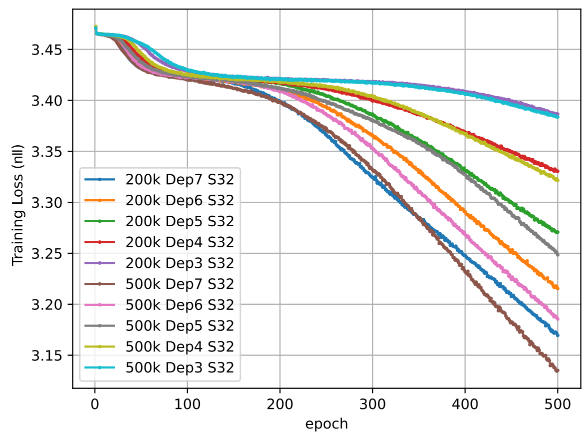


(a) Training loss evolution across epochs for models with varying parameter budgets.

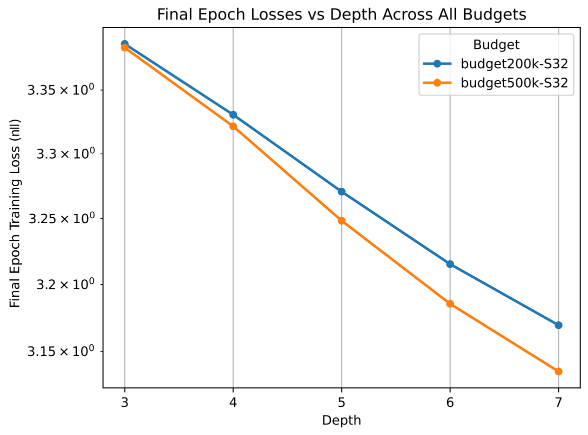


(b) Final training loss vs. network depth

Figure 4: Evaluating the effect of network depth (Halo effect order) on training loss using a synthetic choice dataset with $(S = 16, J = 19)$. The results are based on the equivariant featureless model `DeepHaloFeatureless3D`.



(a) Training loss evolution across epochs for models with varying parameter budgets.



(b) Final training loss vs. network depth

Figure 5: Evaluating the effect of network depth (Halo effect order) on training loss using a synthetic choice dataset with $(S = 32, J = 34)$. The results are based on the equivariant featureless model `DeepHaloFeatureless3D`.

c.1 Synthetic Data Tests: Featureless Setting

Here, we conduct additional tests on synthetically generated data with different choice set sizes S and different universe of alternatives with cardinality J, to verify the theoretical 2^{L-1} order bound. The synthetic data is generated similarly to the previous section. We consider:

- Different network depth: $L \in \{3, 4, 5, 6, 7\}$ if $S > 4$ and $L \in \{1, 2, 3, 4, 5\}$ if $S \leq 4$
- Different choice set/ universe of alternatives cardinality:

$(S, J) \in \{(4, 12), (8, 12), (16, 19), (32, 34)\}$

For each (S, J) pair, we perform the following:

1. Choose $N = \binom{J}{S}$ (available) items for the choice set (each with exactly S alternatives)

2. Sample choice probability vectors uniformly from the simplex on $\mathbb{R}^S$

3. For each choice set, generate 40 i.i.d. choice observations for training and 10 i.i.d. observations for test

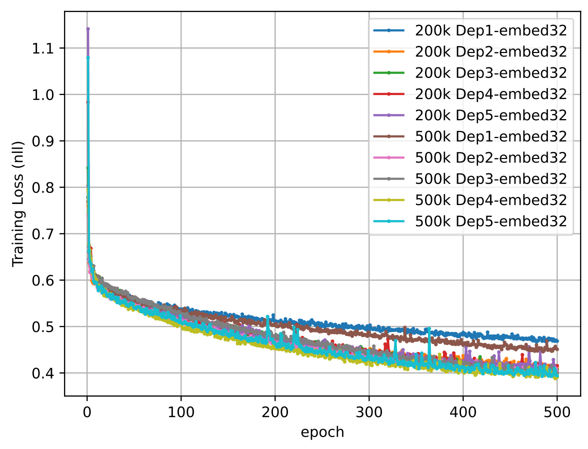
4. For a fixed parameter budget, we adjust the the number of interaction heads H such that the number of trainable parameters of the network matches the budget (we also set the embedding dimension to $d = J$).

Our hypothesis is that training loss should decrease as depth increases until $S - 1 \leq 2^{L-1}$ (i.e., network depth matches or exceeds the maximum interaction order in the choice sets). Beyond this threshold, no significant improvement should occur; the plot of final loss versus depth should begin to flatten. This behavior is visible in Figures 3, 4, and 5, which plot the training loss (negative log-likelihood) versus depth for synthetic datasets with $(S = 4, J = 12)$, $(S = 8, J = 12)$, $(S = 16, J = 19)$, and $(S = 32, J = 34)$, respectively. Here we implemented the `DeepHaloFeatureless3D` model with quadratic activation (for $\sigma(\cdot)$ in equation (6) of [Zhang et al., 2025]) and embedding dimension $d = J$. While loss improvement begins to diminish beyond a certain depth, that depth appears to be one layer more than the theoretical threshold $\lceil \log_2(S - 1) \rceil + 1$. This is perhaps because the chosen choice-set sizes are exact powers of two. Nevertheless, these experiments confirm that the network's expressiveness clearly depends on its depth.

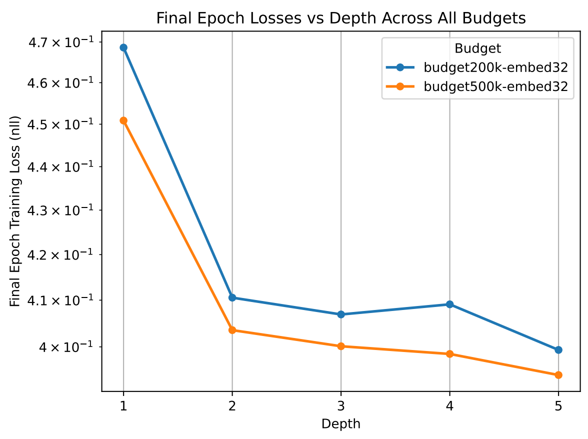
c.2 Real Data Tests: Featured Setting

Next, we test our models using real-world benchmark data with item-level features. For this, we use the transportation mode-choice datasets included in the `choice-learn` package: **ModeCanada** and **SwissMetro**.

ModeCanada (or the Montreal-Toronto Corridor Mode Choice dataset) records individuals' choices among travel modes in Canada including air, train, bus, and car ($J = 4$ items) with item features such as cost, frequency, in-vehicle and out-of-vehicle time, and shared features like trip distance, income, and urban status. SwissMetro is a stated-preference dataset about hypothetical Swiss transport modes (TRAIN, SwissMetro, CAR) with $J = 3$ items, with item features such as availability, travel time, and cost, and shared features like purpose and age. For our experiment, we exclude the shared features and only provide our models with item features, availability mask and



(a) Training loss evolution across epochs for models with varying parameter budgets.



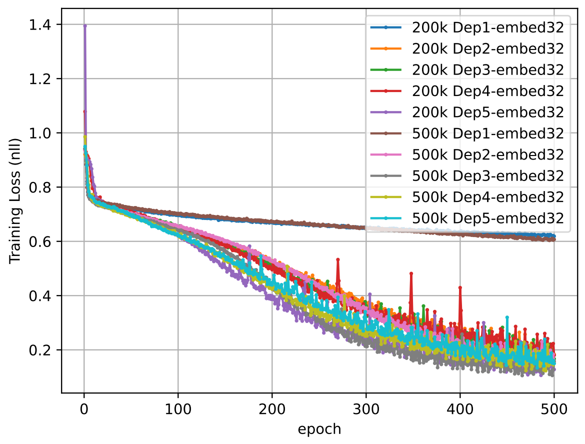
(b) Final training loss vs. network depth

Figure 6: Training loss vs. network depth, evaluated on the (real) ModeCanada transportation choice dataset with $J = 4$ items. The results are based on the `DeepHaloFeatured` model with an embedding dimension of $d = 32$.

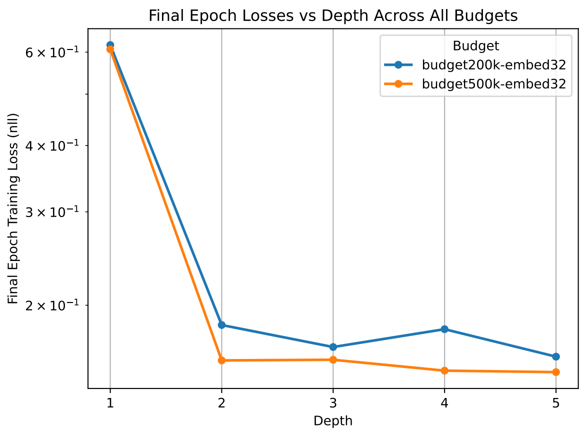
the choices. We choose `'TT'` (travel time) and `'CO'` (cost) as features for the SwissMetro dataset and `'freq'` (frequency of service), `'cost'`, `'ovt'` (out of vehicle time), and `'ivt'` (in-vehicle time) as the items features for ModeCanada dataset. Fig. 6 and Fig. 8 show the results of our `DeepHaloFeatured` model on the ModeCanada and SwissMetro datasets, respectively. As these figures show, for depths beyond $\lceil \log_2(S-1) \rceil + 1$ (which equals 3 for ModeCanada dataset with $S \leq 4$ and 2 for SwissMetro dataset with $S \leq 3$), the training loss improves very little. The fluctuations in these figures arise because the number of parameters is not exactly fixed; with an embedding size of $d = 32$, the exact expected budget cannot be met precisely and may slightly vary with depth.

c.3 Other Tests

We have additionally provided a test suite with `unittest` (within the `tests` directory of the provided `DeepHalo` package) to evaluate different aspects of the designed models including initialization, inheritance, performance, and properties. We particularly test equivariance of the designed models to the permutation of the alternatives in a choice set. As previously discussed, with 3D items features, the featured and featureless implementations of the DeepHalo method based on equations (4) and (5) of Zhang et al. [2025], respectively represented by `DeepHaloFeatureless3D` and `DeepHaloFeatured` models, are permutation equivariant. However, this property does not hold for the original featureless version, which is based on the formulation in equation (10) of Zhang et al. [2025]. This featureless model is, instead, permutation invariant. To verify this, we have also implemented the original featureless version inside the `DeepHaloChoiceModel`. It can be called using the `DeepHaloFeatureless2D` class. The tests can now verify that the



(a) Training loss evolution across epochs for models with varying parameter budgets.



(b) Final training loss vs. depth

Figure 8: Training loss vs. network depth, evaluated on the (real) SwissMetro transportation choice dataset with $J = 3$ items. The results are based on the `DeepHaloFeatured` model with an embedding dimension of $d = 32$.

original model is permutation invariant. These tests can be simply executed with the `DeepHalo_Tests.py` python module inside the `tests` directory.

d Comparison between [Yang et al., 2025] and [Zhang et al., 2025]

d.1 A Summary of [Yang et al., 2025]

Yang et al. [2025] introduce a novel vector-valued Reproducing Kernel Hilbert Space (RKHS) framework for discrete choice modeling.

It assumes that the utility function $\mathbf{u}$ lies in a $\mathbb{R}^J$ vector-valued RKHS $\mathcal{H}_K$ defined on the space of universal sets of items $\mathcal{S}$ with $|\mathcal{S}| = J$. Then, a kernel is defined as $\mathbf{K} : \mathcal{S} \times \mathcal{S} \to \mathbb{R}^{J\times J}$, which must satisfy symmetry and positive definiteness. The paper considers kernels for both featureless and feature-based settings as follow.

Featureless Setting

In featureless model, the entries of the kernel, also called the set kernel, depend only on the choice sets. A common definition is

$$[\mathbf{K}(S, S')]_{jk} = \mathsf{k}_{jk}(S, S') = \mathsf{k}_{jk}(\mathbf{e}_S, \mathbf{e}_{S'}), \tag{43}$$

where $\mathbf{e}_S \in \{0,1\}^J$ represents the availability mask or the indicator for set S. To ensure $u_j(S) = 0$ if $j \notin S$, a masked set kernel $\mathbf{K}^S(S, S')$ is also defined with entries given by

$$\mathsf{k}^S_{jk}(S, S') = \mathbb{1}(j \in S) \cdot \mathbb{1}(k \in S') \cdot \mathsf{k}_{jk}(S, S').$$

For example, [Yang et al., 2025] proposes the following polynomial kernel of order p to capture interaction (Halo) effects of up to p-th order:

$$\mathsf{k}^S_{j,k}(S, S') = \mathbb{1}(j \in S)\mathbb{1}(k \in S')(\boldsymbol{e}_S^\top \boldsymbol{e}_{S'})^p. \tag{44}$$

Feature-Based Setting

For scenarios with item features $\mathbf{X}_S \in \mathbb{R}^{d\times J}$, a product kernel is defined as

$$\bar{\mathbf{K}}((S, \mathbf{X}_S), (S', \mathbf{X}_{S'})) := \mathbf{K}^S(S, S') \cdot \mathbf{K}^{\mathcal{X}}(\mathbf{X}_S, \mathbf{X}_{S'}).$$

where $\mathbf{K}^S(S, S')$ is a masked ste kernel and $\mathbf{K}^{\mathcal{X}}(\mathbf{X}_S, \mathbf{X}_{S'})$ represents a feature kernel defined over items features.

A common choice for the feature kernel is a Gaussian kernel, specified by

$$[\mathbf{K}^{\mathcal{X}}(\mathbf{X}_S, \mathbf{X}_{S'})]_{ij} = \sigma^2 \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}'_j\|^2}{2\ell^2}\right),$$

where $\boldsymbol{x}_i$ is the i-th column of $\mathbf{X}_S$.

Now, consider a dataset $\mathcal{D}_N = \{(S_i, \mathbf{X}_{S_i}, \mathbf{y}_i)\}_{i=1}^N$ where S_i is the choice set, $\mathbf{X}_{S_i}$ is the feature matrix of the items in S_i, and $\mathbf{y}_i$ is the observed choice vector. Then, using a predefined kernel $\bar{\mathbf{K}}((S, \mathbf{X}_S), (S', \mathbf{X}_{S'}))$, the utility function can be formulated using the RKHS reproducing property as

$$\mathbf{u}(\cdot) = \sum_{i=1}^N \bar{\mathbf{K}}(\cdot, (S_i, \mathbf{X}_{S_i}))\boldsymbol{\alpha}_i \tag{45}$$

$$u_j(S) = \mathbf{e}_j^\top \sum_{i=1}^N \mathbf{K}^S(S, S_i)\mathbf{K}^{\mathcal{X}}(\mathbf{X}_S, \mathbf{X}_{S_i})\boldsymbol{\alpha}_i \tag{46}$$

where $\boldsymbol{\alpha}_i \in \mathbb{R}^J$ are the coefficients to be learned. The learning problem is then based on a regularized risk minimization

$$\min_{\mathbf{u}\in\mathcal{H}_{\bar{K}}} \frac{1}{N}\sum_{i=1}^N \ell(\mathbf{y}_i, \hat{\mathbf{y}}_i(\mathbf{u})) + \lambda\|\mathbf{u}\|_{\mathcal{H}^2_{\bar{K}}},$$

where $\hat{\mathbf{y}}_i(\mathbf{u})$ denotes the predicted choice probability vector for set S_i. This problem can finally be converted to a finite-dimensional convex problem over $\boldsymbol{\alpha}_i$s using the representer theorem.

Given this introduction on the RKHS method [Yang et al., 2025], we can now compare it with the DeepHalo model by Zhang et al. [2025]:

d.2 DeepHalo Method by Zhang et al. [2025]

Pros

- **Explicit interpretation of interaction order:** DeepHalo method can interpret interactions by order (pairwise, 3-way, etc.), as given in equation (2) of Zhang et al. [2025]. It clearly models how to handle maximum interaction order via depth L and quadratic activations (expressiveness up to order 2^{L-1}). It also provides criteria (e.g., relative context effect $\alpha_{jk}(T)$) to systematically identify and visualize item–item influences. Thus, it can explicitly identify common context effects such as decoy, similarity, and compromise effects.

- **Permutation-equivariance:** The DeepHalo method provides a permutation equivariant approach to model context effects. This equivariance supported by a theoretical proof (proposition 3.1 of Zhang et al. [2025]) makes the model more sample-efficient and robust to spurious pattern inference (e.g., always favoring the first column) and thus has a better inductive bias. This property is at least valid for featured data settings based on equations (4) and (5) of Zhang et al. [2025]. Although in featureless setup, the aggregated reformulation in equation (10) is not permutation equivariant, one can still implement this model using a similar featured implementation with one-hot encoded features.

- **Scalable training:** Like most common neural architectures, DeepHalo training uses standard backpropagation and mini-batch stochastic gradient-based methods (e.g., Adam), which scales well to large sample sizes as long as the architecture is appropriately designed. It can also be easily integrated with GPU-accelerated frameworks for faster training.

Cons

- **Architectural complexity for high-order interactions:** The complexity of the DeepHalo method increases with the depth (number of layers). Using linear transformation for context effect modeling as in equation (4) of Zhang et al. [2025], one requires $L = |S| - 1$ layers to capture all interaction orders. The number of layers can be reduced with a polynomial activation function (for example, we require $L = \log_2(|S| - 1)$ with quadratic activation). For polynomials of high degrees, however, the optimization becomes unstable as the gradients begin to explode. As a result, the DeepHalo method may not efficiently model all interaction orders in choice sets with very large number of alternatives.

- **Non-convex training and optimization:** DeepHalo formulates a non-convex optimization problem due to non-convex NN activation functions (e.g., softmax). Thus, training can be sensitive to initialization, learning rates, and regularization strategies, and may converge to different local minima. There are also no strong theoretical guarantees of convergence or statistical learning bounds in Zhang et al. [2025].

- **Regularization and hyperparameter selection challenges:** DeepHalo requires careful selection of architectural hyperparameters (depth L, number of attention heads H, size of the embedding d, the nonlinear map and activation functions), regularization (weight decay, dropout), normalization (batch normalization layers) and optimization settings (algorithm, learning rate).

- **Limited feature integration:** DeepHalo, in its basic form, is limited to operate on item-level features, without integrating richer (shared) choice-set or individual-level covariates into the utility. Hence, the model cannot efficiently model market characteristics or person-specific behaviors.

d.3 RKHS Choice Model by Yang et al. [2025]

Pros

- **Convexity and global optimality:** The RKHS choice model leads to a convex optimization problem in the coefficients for a fixed kernel and convex loss (based on Theorem 1 of [Yang et al., 2025]), offering global optimality and greater stability in estimation and initialization.

- **Strong theoretical foundations:** Yang et al. [2025] provides strong theoretic guarantees which analyze both optimization and generalization behavior of the RKHS choice model. Theorem 2 shows that, in the neural regime, stochastic gradient descent on the corresponding neural implementation achieves an expected objective value that converges to the optimal RKHS risk at the classical $O(1/\sqrt{K})$ rate (with K being the number of iterations), despite the underlying nonconvex parameterization. Theorem 3 also provides a finite-sample generalization bound of order $O(1/\sqrt{N})$ for all utilities with bounded RKHS norm, controlling sample size and function complexity trade off in choice prediction.

- **Direct regularization control:** The RKHS model by Yang et al. [2025] offers a principled approach to prevent overfitting via regularization. It provides explicit

Table 1: Comparison of DeepHalo vs. RKHS Choice Model

Aspect	**DeepHalo [Zhang et al., 2025]**	**RKHS Choice Model [Yang et al., 2025]**
Permutation-equivariance	Yes, with symmetric aggregation and shared transformations	No, if considering within-set (test) permutation
Interpretability	High, pair-wise and higher order context effects (visualize item–item influences)	Moderate, global function properties, no submodular decomposition
Scalability	Scales well with large N via back propagation	Often $O(N^2)$ kernel matrix costs (could be reduced to $O(N)$ with random kernel approximation with the cost of losing precision)
Guarantees	Lacks a theoretical guarantee	Strong guarantees, generalization bounds, stability, convergence
Optimization	Non-convex, requires tuning, SGD	Convex, guarantees global optimum
Structural Requirements	Design network layers, activations, depth	Requires kernel selection, feature learning
Feature Integration	Weak: primarily works on item-level features	String: can incorporate individual/assortment-level features into kernel design
Application	Large datasets, need to measure specific context effects	Small/medium datasets, strong guarantees, stable optimization

norm-based regularization (e.g. $\|\mathbf{u}\|^2_{\mathcal{H}}$), enabling direct control of function smoothness and expressiveness with strong generalization bounds derived from RKHS theory.

- **Better feature integration:** In the RKHS method, each observation is $(S, \mathbf{X}_S)$, where columns of $\mathbf{X}_S$ are item features, and the kernel $\mathbf{K}^{\mathcal{X}}$ can be defined on any feature matrix. One can concatenate shared or individual-level features (e.g., market conditions, user attributes) to each column or add extra rows/columns encoding them. Alternatively, the product kernel can be reformulated by incorporating these shared features as:

$$\bar{\mathbf{K}}\left((S, \mathbf{X}_S, , \tilde{\mathbf{x}}_S), (S', \mathbf{X}_{S'}, \tilde{\mathbf{x}}_{S'})\right) = \mathbf{K}^{\mathcal{X}}(\mathbf{X}_S, \mathbf{X}_{S'})\mathbf{K}^{\tilde{\mathcal{X}}}(\tilde{\mathbf{x}}_S, \tilde{\mathbf{x}}_{S'})\mathbf{K}^{S}(S, S') \tag{47}$$

where $\tilde{\mathbf{x}}_S$ denotes the shared choice/individual-level features. The RKHS method can consequently model utilities as functions of item features, choice-level attributes, and individual characteristics within a unified framework.

Cons

- **Not permutation-equivariant in the common sense:** The RKHS method by Yang et al. [2025] is not equivariant to the permutation of the items in the test set. That is because the kernel compares a test set $(S, \mathbf{X}_S)$ against fixed training sets $(S_i, \mathbf{X}_{S_i})$ using a matrix kernel $\bar{\mathbf{K}}$ on the ordered padded feature matrix $\mathbf{X}_S$.

So changing the column (item) order as $\mathbf{X}_{\pi(S)} = \mathbf{X}_S\mathbf{P}$ (while keeping global item identities and training sets fixed) changes the kernel value and thus changes $u_j(\mathbf{X}_S)$

- **Poor scalability with large sample size N:** With fixed stationary kernels, precomputation and storage of kernel values creates substantial memory challenge. The training complexity also scales as $O(J^2N^2)$ making RKHS method infeasible for massive datasets. The complexity can be reduced to $O(N_wNJ)$ with random kernel approximation as described in Section 3.3 of [Yang et al., 2025], where N_w is the number of discrete points used to approximate the distribution of the feature space. However, this efficiency gain comes at the cost of approximation error. It also requires learning a lower-dimensional random feature map as a secondary task.

- **Kernel selection is challenging and ad hoc:** A key disadvantage of the RKHS approach is that kernel selection and design (for both sets and features) is often challenging and somewhat ad hoc, requiring problem-specific choices and tuning and can strongly affect performance. This is evident in Fig. 1 of [Yang et al., 2025] which compares the performance of three different kernels including fixed, random features, and neural tangent kernels (NTK).

- **Limited interpretability of context effects:** The RKHS model by Yang et al. [2025] does not naturally yield a decomposition of the utility function into pairwise and higher-order interaction terms. In fact, in this method, interactions of different orders are simultaneously and implicitly entangled inside the kernel expansion and the method does not provide direct, order-wise decomposition or explicit control over the highest interaction order. Hence, one cannot efficiently model specific context effects such as decoy or compromise effect.

e Credit Card Offer Demand Estimation and Key Challenges

Both RKHS choice model by [Yang et al., 2025] and DeepHalo model by [Zhang et al., 2025] can, in principle, be used for demand estimation of credit card offer. However, each comes with distinct advantages and disadvantages.

e.1 Application of RKHS Choice Model by Yang et al. [2025]

The RKHS-based choice model is a flexible nonparametric method that can capture complex, nonlinear relationships between consumer attributes, item features, and choice probabilities via appropriate kernel design. In a credit card offer setting, one can define kernels over:

- Cardholder features (spending history, demographics),
- Item-level features (merchant category, discount level),
- Offer set itself (assortment-level features such as maximum discount).

This approach lets the model capture diverse consumer behaviors and smooth changes in preference without relying on strict, predefined assumptions. This is especially useful in marketing and finance, where the true nature of demand is often unclear.

However, main challenges include:

- **Kernel design and engineering.** Choosing and tuning a suitable kernel (or combination of kernels) over high-dimensional, mixed-type features and potentially large offer sets is non-trivial. Poor kernel choice can limit the expressiveness of the RKHS method.
- **Scalability:** Standard RKHS methods involve kernel matrix operations that scale at least quadratically in the number of training examples; vector-valued kernels further increase computational cost. For millions of credit card users, one would likely need random feature approximation, introducing additional approximation error and design complexity.
- **Lack of permutation equivariance:** RKHS construction is not permutation equivariant with respect to reordering items within the test choice set. Practically, this means that the model can implicitly depend on how offers are indexed, rather than treating the menu as an abstract set, which is problematic when card offers change, new offers are introduced, or items are arranged in different orders across markets and time; in such settings, lack of built-in set symmetry can hurt robustness, complicate implementation, and make it harder to guarantee that predicted responses depend only on which offers and features are present, not on arbitrary positional encodings.
- **Context effects and high-order interactions.** While context dependence can be modeled via kernels on sets, interaction orders are not explicitly separated. If higher-order offer interactions (e.g., complex cannibalization/complementarity effects across many co-present offers) are important, the RKHS formulation may capture them implicitly but will not provide clean interpretability by interaction order.

e.2 Application of DeepHalo Model by Zhang et al. [2025]

DeepHalo is explicitly designed to handle context-dependent choice behavior, where choice probabilities depend not only on individual offers but also on the composition of the items in an offer set. This is directly relevant to credit card offers, where:

- Multiple offers are shown simultaneously to a customer,
- Certain offers may dominate or reinforce each other (e.g., overlapping merchant discounts),
- Behavioral effects such as attraction, compromise, and decoy effects may arise from the menu of offers.

However, there are also some challenges with the DeepHalo method:

- **Feature Integration Limit:** For the credit card offer problem, a central limitation of the DeepHalo model is its lack of native integration of shared choice-set or individual-level features. That is because the architecture is primarily designed over item-level features, rather than utilities that also depend on market- or person-specific features such as campaign type, or cardholder spending histories. While such information could be cast into the item feature vectors, this is heuristic and quickly becomes messy when the same shared variable applies to all offers in a choice. Hence, it is generally hard to capture realistic demand patterns where both offer attributes and consumer/market characteristics jointly affect response to promotions.
- **Complexity.** DeepHalo's ability to model higher-order interactions grows by adding more layers, but this also increases computational complexity and parameter count roughly with depth and number of items. This can become a disadvantage when estimating demand for credit card offers with many alternatives, as training and inference may become costly and harder to stabilize at scale. It may require large volumes of high-quality financial data with sufficient variation in observed assortments.
- **Non-convex optimization.** Training is non-convex and may be sensitive to initialization, architecture choice, and regularization. In production, this can translate into instability across repeated trainings and the need for careful design strategies to avoid overfitting (dropout, early stopping, normalization).

e.3 Summary

- The RKHS method [Yang et al., 2025] offers a theoretically-founded approach that is well suited to small- or medium-scale credit card offer demand estimation when the goal is to incorporate choice- and individual-level features for richer utility modeling; however, the lack of permutation equivariance means that item indexing and the way offers are aligned between training and test sets must be handled carefully to avoid artefactual biases in predictions.

- The DeepHalo method [Zhang et al., 2025] is particularly suitable for large datasets with millions of users, especially when context effects and higher-order interactions between offers are important and there is a premium on explicitly modeling and interpreting those interaction patterns. DeepHalo is more of a powerful but specialized context-effect module that would need to be embedded into a richer framework that incorporates shared assortment- or individual-level features.

Both methods still face several challenges for realistic credit-card offer demand estimation. For instance, neither model accounts for panel structure learning, habit formation, or stockpiling over time. These limitations will be addressed later in the bonus questions.

f Other (Non-Neural and ML) Approaches to Choice Modeling

Traditional discrete choice models beyond Yang's RKHS [Yang et al., 2025] and Zhang's DeepHalo [Zhang et al., 2025] and broader Machine Learning (ML) techniques may also be highly relevant for credit card offer demand estimation. These methods frequently offer a more practical trade-off, balancing model complexity and predictive power while prioritizing essential real-world needs such as interpretability and robustness.

f.1 Other Discrete Choice Models

The **Multinomial Logit (MNL)** and **Conditional Logit** models [Train, 2009] are still important baselines in marketing and transportation. They are extremely fast to estimate even on very large datasets and yield directly interpretable coefficients (e.g., the marginal utility of price or the effect of specific merchant categories), which is valuable when the ultimate objective is to understand demand drivers and compute elasticities or willingness-to-pay (WTP) rather than just predict choices. Their main limitation is the independence-of-irrelevant-alternatives (IIA) property where they fail to capture context effects; however, as baseline methods, they are hard to beat in terms of simplicity and robustness.

To relax IIA while staying relatively simple, **Nested Logit** allows correlation in unobserved utility within nests of similar alternatives. For example, credit card offers can be grouped into nests such as "grocery", "travel", and "dining" rewards, with each nest having its own scale parameter [Train, 2009]. The nested logit probability can be written as a product of a conditional probability of choosing j within its nest and a probability of choosing the nest itself, both in logit form. This structure produces more realistic substitution patterns, with stronger interaction effects within nests (e.g., between two restaurant offers) than across nests, and is still estimated by relatively simple maximum-likelihood estimation methods. Compared to highly flexible models like DeepHalo, nested logit trades off some expressiveness for a much clearer behavioral interpretation of cross-substitution, which is crucial when designing or evaluating campaign policies.

Another important family is **Latent Class MNL** [Kamakura and Russell, 1989, Greene and Hensher, 2003]. Here the population is assumed to be a mixture of C latent segments, each with its own parameter vector $\boldsymbol{\beta}_c$, and each individual i belongs to class c with probability π_c. The class-conditional choice probabilities have the standard MNL form. This model captures taste heterogeneity in an interpretable way. For example, in credit card demand estimation, classes can be understood as segments of customers such as "reward maximizers" who are highly sensitive to price and discount, "brand loyal" customers who favor particular merchants, etc. It introduces discrete heterogeneity beyond a single MNL, is easy to estimate on large panels, and provides segment-level coefficients that directly support elasticity calculation and segment-specific offer design.

Halo MNL and its Low-Rank extension explicitly introduce context effects while retaining a logit-based structure. In Halo MNL, the utility of an item in an assortment includes not only its own attributes but also interaction terms capturing how the presence of other items (e.g., competing offers) alters its attractiveness, thereby modeling halo and cannibalization effects in a parametric way. Low-Rank Halo MNL further constrains the interaction matrix to be low rank, which both reduces the number of free parameters and admits an efficient implementation, effectively interpolating between standard MNL (no interactions) and full Halo MNL (dense interactions). For credit card offer demand, these models can capture important context effects—such as a strong travel offer depressing response to weaker travel offers.

Taken together, these traditional models may be preferable or at least complementary to Yang's RKHS and Zhang's DeepHalo models in many practical settings: they are computationally lighter, have transparent behavioral structure, integrate naturally with tools for dealing with endogeneity (e.g., IV in BLP-type settings), and provide the kind of elasticity and WTP measures that credit card issuers need for pricing and campaign design.

f.2 ML Methods

ML techniques such as **support vector machines (SVM)**, **random forests**, and **gradient boosting** (e.g. XGBoost, LightGBM) may also be partially suitable for choice modeling: they can predict choices well. However, they do not, by themselves, provide the full structure that discrete choice analysis usually requires.

From a predictive viewpoint, suppose each observation is encoded as a feature *vector* $\mathbf{x}_i \in \mathcal{X}$ and a discrete (multi-class) choice label $y_i \in \{1, \ldots, J\}$. An SVM or gradient-boosted tree essentially learns a classifier function

$$f : \mathcal{X} \to \{1, \ldots, J\},$$

or, in probabilistic form,

$$p_\theta(y_i = j \mid \mathbf{x}_i) \approx \hat{p}_j(\mathbf{x}_i),$$

by minimizing an empirical loss (e.g. hinge loss or cross-entropy) plus regularization. In this sense, such models are fully capable of approximating

$$P(\text{alternative } j \text{ is chosen} \mid \mathbf{x}_i)$$

and often achieve high out-of-sample accuracy when $\mathbf{x}_i$ contains rich individual and alternative-specific features. Empirical benchmarks show that ensemble ML methods and deep learning can match or outperform classical logit-type models in pure prediction tasks [Püschel et al., 2024, Wang et al., 2024].

However, classical discrete choice models typically start from a random utility formulation,

$$u_{ij} = v_{ij} + \varepsilon_{ij}, \qquad P(y_i = j \mid \{v_{ik}\}_k) = \Pr\big(u_{ij} \geq u_{ik},\ \forall k\big),$$

where v_{ij} is a systematic utility with interpretable taste parameters (e.g. coefficients on price, time, and other attributes), and ε_{ij} has a specified distribution (logit, probit, etc.). This structure yields closed-form or numerically tractable choice probabilities, marginal effects, and elasticity measures derived from parameters such as β_{price}. Generic ML models do not impose this utility structure; they learn a flexible mapping $\mathbf{x}_i \mapsto \hat{p}(y_i \mid \mathbf{x}_i)$ without identified taste parameters. This makes it difficult to compute economically meaningful quantities like WTP in a transparent and stable way across counterfactuals.

Moreover, applied demand work often faces endogeneity (e.g. prices or offer targeting correlated with unobserved demand shocks). Identification strategies such as instrumental variables or control functions are naturally formulated in parametric discrete choice frameworks, whereas ML approaches such as SVM/forest/boosting models have no built-in mechanism to enforce the orthogonality conditions required for causal interpretation. Extending them to properly incorporate instruments is possible but nontrivial and not standard practice.

Finally, choice data have an intrinsic set structure: each observation involves a choice set S_i and alternative-specific attributes $\mathbf{x}_{ij}$ for $j \in S_i$. Classical models explicitly encode this via utility functions $v_{ij}(\mathbf{x}_{ij}, \mathbf{z}_i)$ and normalization across $j \in S_i$, ensuring coherent treatment of varying assortments and availability. Generic ML classifiers typically flatten the problem into a single vector $\mathbf{x}_i$, unless one explicitly engineers a representation that respects permutation equivariance and variable choice sets. Without such design, handling changing assortments (permutations) and availability in a principled way is difficult.

In summary, SVMs, random forests, and gradient boosting are suitable as powerful predictive models for discrete choices and useful as benchmarks or components, but they are not, by default, suitable as full replacements for discrete choice models when the goal is economic interpretation, causal demand estimation, or structured counterfactual analysis.

f.3 Experimental Evaluation

For a proven numerical evaluation, we implement different choice models on **SwissMetro** dataset from `choice_learn`. This dataset offers 3 transportation mode choices including TRAIN, SwissMetro, and CAR ($J = 3$), with item features such as availability, travel time, and cost, and shared features like purpose and age. For this experiment, we only include `'TT'` (travel time) and `'CO'` (cost) as the item features.

We compare our developed `DeepHaloChoiceModel` model with built-in choice models in `choice_learn.models` including `SimpleMNL`, `ConditionalLogit`, `LatentClassSimpleMNL`, `NestedLogit`, `HaloMNL`, and `LowRankHaloMNL`. We can build, train and evaluate these models using the methods and tools available in the `choice_learn` package.

We then compare with the ML baseline algorithms such as SVM, Random Forest, and Gradient Boosting. The first two methods can be implemented using the `sklearn` package. For gradient boosting (LGB and XGB), we import `lightgbm` and `xgboost` packages.

For the ML methods, we prepare the training data in two distinct vector formats. First, we provide the plain features of each alternative in a choice set flattened as

$$\mathbf{x}_i = [a_{i1}\mathbf{x}_{i1}, a_{i2}\mathbf{x}_{i2}, \ldots, a_{iJ}\mathbf{x}_{iJ}]^\top, \quad y_i \in \{1, \ldots, J\}, \quad J = 3 \tag{48}$$

where $\mathbf{x}_{ij} = [\mathrm{CO}_{ij}, \mathrm{TT}_{ij}]$ represents cost and travel time for alternative j in choice situation i, and $a_{ij} \in \{0, 1\}$ encodes the availability of item j. Unavailable alternatives, indicated by the availability mask, are assigned zero feature vectors then. This approach may lead to a complex solution for high-dimensional item features. Another alternative is to use relative context features, such as the difference of each item's features with respect to the choice-set average and maximum values. The data vector then takes the form:

$$\mathbf{x}_i^j = [\mathbf{x}_{ij}, \mathbf{x}_{ij} - \bar{\mathbf{x}}_i, \mathbf{x}_{ij} - \max_k \mathbf{x}_{ik}, \frac{\mathbf{x}_{ij} - \bar{\mathbf{x}}_i}{\boldsymbol{\sigma}_i}], \quad y_i^j = \mathbb{1}(y_i = j) \tag{49}$$

where $\mathbf{x}_{ij}$ is the feature vector of item j in choice situation i, $\bar{\mathbf{x}}_i$ is the choice set mean, and

$\boldsymbol{\sigma}_i$ is the standard deviation of the features in the choice set (across items). Specifically, we formulate the problem as a binary classification and treat each item in a choice set as a separate training sample. This increases the number of samples while keeping the dimensionality limited.

Table 2 shows the numerical results in terms of test accuracy and runtime (in seconds). Gradient-boosting methods (XGB and LGB) excel due to sequential error correction and built-in handling of class imbalance, interactions, and missing data. Contextual features also improve the results, especially for SVM and random forests (these models are specified by the "Context." prefix). These ML approaches are also generally faster than conventional choice-model algorithms. Nevertheless, the RUMnet choice model attains the highest accuracy and meets the economic-interpretability and demand-estimation requirements of choice modeling, despite its greater complexity.

Model	**Test Accuracy**	**Time (s)**
XGB	0.732	1.51
Context. XGB	0.702	0.79
LGB	0.697	0.96
Context. LGB	0.704	**0.65**
RandomForest	0.645	1.27
Context. RandomForest	0.693	2.98
SVM	0.645	10.91
Context. SVM	0.665	67.59
DeepHalo	0.667	30.71
SimpleMNL	0.617	4.59
ConditionalLogit	0.608	19.02
NestedLogit	0.678	14.24
HaloMNL	0.606	4.54
LowRankHaloMNL	0.606	40.99
RUMnet	**0.736**	303.06

Table 2: Test accuracy and runtime (in seconds) for classical discrete choice models and machine learning methods on SwissMetro dataset.

References

Ali Aouad and Antoine Désir. Representing random utility choice models with neural networks, 2023. URL https://arxiv.org/abs/2207.12877.

Vincent Auriau, Ali Aouad, Antoine Désir, and Emmanuel Malherbe. Choice-learn: Large-scale choice modeling for operational contexts through the lens of machine learning. *Journal of Open Source Software*, 9(101):6899, 2024. doi: 10.21105/joss.06899. URL https://doi.org/10.21105/joss.06899.

Steven Berry, James Levinsohn, and Ariel Pakes. Automobile prices in market equilibrium. *Econometrica*, 63(4):841–890, July 1995. doi: None. URL https://ideas.repec.org/a/ecm/emetrp/v63y1995i4p841-90.html.

Michel Bierlaire. Acceptance of modal innovation: the case of the swissmetro. 2001. URL https://api.semanticscholar.org/CorpusID:15826408.

H. D. Block. *Random Orderings and Stochastic Theories of Responses (1960)*, pages 172–217. Springer Netherlands, Dordrecht, 1974. ISBN 978-94-010-9276-0. doi: 10.1007/978-94-010-9276-0_8. URL https://doi.org/10.1007/978-94-010-9276-0_8.

Christopher V. Forinash and Frank S. Koppelman. Application and interpretation of nested logit models of intercity mode choice. *Transportation Research Record*, pages 98–106, 1993. URL https://api.semanticscholar.org/CorpusID:152304413.

William H. Greene and David A. Hensher. A latent class model for discrete choice analysis: contrasts with mixed logit. *Transportation Research Part B: Methodological*, 37(8):681–698, 2003. ISSN 0191-2615. doi: https://doi.org/10.1016/S0191-2615(02)00046-2. URL https://www.sciencedirect.com/science/article/pii/S0191261502000462.

Yafei Han, Francisco Camara Pereira, Moshe Ben-Akiva, and Christopher Zegras. A neural-embedded choice model: Tastenet-mnl modeling taste heterogeneity with flexibility and interpretability, 2022. URL https://arxiv.org/abs/2002.00922.

Stephane Hess, Andrew Daly, and Richard Batley. Revisiting consistency with random utility maximisation: theory and implications for practical work. *Theory and Decision*, 84(2):181–204, March 2018. doi: 10.1007/s11238-017-9651-7. URL https://ideas.repec.org/a/kap/theord/v84y2018i2d10.1007_s11238-017-9651-7.html.

Wagner A. Kamakura and Gary J. Russell. A probabilistic choice model for market segmentation and elasticity structure. *Journal of Marketing Research*, 26(4):379–390, 1989. doi: 10.1177/002224378902600401. URL https://doi.org/10.1177/002224378902600401.

Joohwan Ko and Andrew A. Li. Modeling choice via self-attention, 2024. URL https://arxiv.org/abs/2311.07607.

Zhentong Lu and Kenichi Shimizu. Estimating discrete choice demand models with sparse market-product shocks, 2025. URL https://arxiv.org/abs/2501.02381.

Reza Yousefi Maragheh, Alexandra Chronopoulou, and James Mario Davis. A customer choice model with halo effect. *arXiv: Applications*, 2018. URL https://api.semanticscholar.org/CorpusID:88517466.

Daniel McFadden. Conditional logit analysis of qualitative choice behavior. In Paul Zarembka, editor, *Fontiers in Econometrics*, pages 105–142. Academic press, New York, 1974.

Daniel McFadden. Modelling the choice of residential location. (477), 1978. URL https://EconPapers.repec.org/RePEc:cwl:cwldpp:477.

Karlson Pfannschmidt, Pritha Gupta, Björn Haddenhorst, and Eyke Hüllermeier. Learning context-dependent choice functions. *International Journal of Approximate Reasoning*, 140:116–155, 2022. ISSN 0888-613X. doi: https://doi.org/10.1016/j.ijar.2021.10.002. URL https://www.sciencedirect.com/science/article/pii/S0888613X21001614.

Jasper Püschel, Lukas Barthelmes, Martin Kagerbauer, and Peter Vortisch. Comparison of discrete choice and machine learning models for simultaneous modeling of mobility tool ownership in agent-based travel demand models. *Transportation Research Record*, 2678(7):376–390, 2024. doi: 10.1177/03611981231206175. URL https://doi.org/10.1177/03611981231206175.

Brian Sifringer, Virginie Lurkin, and Alexandre Alahi. Enhancing discrete choice models with representation learning. *Transportation Research Part B: Methodological*, 140:236–261, October 2020. ISSN 0191-2615. doi: 10.1016/j.trb.2020.08.006. URL http://dx.doi.org/10.1016/j.trb.2020.08.006.

Kenneth E. Train. *Discrete Choice Methods with Simulation*. Cambridge University Press, 2nd edition, 2009. Chapter ”Logit”.

Kenneth E. Train, Daniel McFadden, and Moshe E. Ben-Akiva. The demand for local telephone service: a fully discrete model of residential calling patterns and service choices. *The RAND Journal of Economics*, 18:109–123, 1987. URL https://api.semanticscholar.org/CorpusID:153685064.

Shenhao Wang, Baichuan Mo, Yunhan Zheng, Stephane Hess, and Jinhua Zhao. Comparing hundreds of machine learning and discrete choice models for travel demand modeling: An empirical benchmark. *Transportation Research Part B: Methodological*, 190:103061, December 2024. ISSN 0191-2615. doi: 10.1016/j.trb.2024.103061. URL `http://dx.doi.org/10.1016/j.trb.2024.103061`.

Melvin Wong and Bilal Farooq. Reslogit: A residual neural network logit model for data-driven choice modelling. *Transportation Research Part C: Emerging Technologies*, 126: 103050, 2021. ISSN 0968-090X. doi: https://doi.org/10.1016/j.trc.2021.103050. URL `https://www.sciencedirect.com/science/article/pii/S0968090X21000802`.

Yiqi Yang, Zhi Wang, Rui Gao, and Shuang Li. Reproducing kernel hilbert space choice model. In *Proceedings of the 26th ACM Conference on Economics and Computation*, EC '25, page 819, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400719431. doi: 10.1145/3736252.3742630. URL `https://doi.org/10.1145/3736252.3742630`.

Shuhan Zhang, Zhi Wang, Rui Gao, and Shuang Li. Deep context-dependent choice model. In *2nd Workshop on Models of Human Feedback for AI Alignment*, 2025. URL `https://openreview.net/forum?id=bXTBtUjb0c`.