Appendix

Appendix Contents

A Proofs and Derivations

A.1 Proof of Proposition 1

Proof. Consider the following form of v:

$$v_j(X_{T\cup\{j\}}) := \sum_{R\subset T} (-1)^{|T|-|R|} u_j(x_j, X_R). \tag{12}$$

Thanks to the permutation equivariance of u_j, the function v_j given by (3) is permutation equivariant. Indeed, for every permutation π on $T \cup \{j\}$, it holds that

$$\begin{aligned} v_j(X_{\pi(T\cup\{j\})}) &= \sum_{Q\subset\pi(T)} (-1)^{|\pi(T)|-|Q|} u_j(X_{Q\cup\pi(j)}) \\ &= \sum_{R\subset T} (-1)^{|T|-|R|} u_{\pi(j)}(X_{R\cup\{j\}}) \\ &= v_{\pi(j)}(X_{T\cup\{j\}}), \end{aligned}$$

where the second equality follows from the change of variable $R = \pi^{-1}(Q)$.

Next, let us verify that the function v given in (3) can express the utility function $u_j(X_S)$. Indeed, for every S and $j = 1, \ldots, |S|$, using (3) we write $\sum_{T\subset S\setminus\{j\}} v(x_j, X_T)$ as

$$\sum_{T\subset S\setminus\{j\}} \sum_{R\subset T} (-1)^{|T|-|R|} u(x_j, X_R).$$

By switching the order of summation, this equals

$$\sum_{R\subset S\setminus\{j\}} u(x_j, X_R) \sum_{T\supset R, T\subset S\setminus\{j\}} (-1)^{|T|-|R|}.$$

When $R = S \setminus \{j\}$, the inner sum is 1. Otherwise, the inner sum equals

$$\sum_{k=0}^{|S|-1-|R|} (-1)^k \binom{|S|-1-|R|}{k} = 0^{|S|-1-|R|} = 0.$$

This shows that

$$\sum_{T\subset S\setminus\{j\}} v(x_j, X_T) = u_j(X_S),$$

which completes the proof.

□

A.2 Derivation in Section 4.2

A.2.1 Recursion for y^l

Let us first write $y^l = \sum_{j=1}^J z_j^l$ with $y^0 = e_S$ where z^l is defined in (9). Since $y^0 = e_S = \sum_{j=1}^J e_j = \sum_{j=1}^J z_j^0$, it implies $\chi(x) = x$ being an identity mapping such that $z_j^0 = e_j$. Recall

$$\Theta^l = \frac{1}{HS} \sum_{h=1}^H (q_{h,\cdot}^l)^\mathsf{T} W_{h,\cdot}^l, \quad \Theta_{jk}^l = \frac{1}{HS} \sum_{h=1}^H q_{hj}^l W_{hk}^l.$$

We write

$$z_j^1 = e_j + \frac{1}{HS} \sum_{h=1}^H \Big(\sum_{k\in S} W_{hk}^1 \Big) \cdot q_{hj}^1 e_j = e_j + \sum_{k\in S} \Theta_{jk}^1 e_j.$$

It follows that

$$y^1 = \sum_{j=1}^J e_j + \sum_{j=1}^J \sum_{k\in S} \Theta_{jk}^1 e_j = \sum_{j=1}^J e_j + \sum_{j=1}^J \sum_{k\in S} \Theta_{jk}^1 e_k e_j = y^0 + \Theta^1(e_S \odot e_S) = y^0 + \Theta^1(y^0 \odot e_S).$$

For $l = 2, \ldots, L$,

$$\begin{aligned} y^l =& \sum_{j=1}^{J} z_j^{l-1} + \sum_{j=1}^{J} \frac{1}{HS} \sum_{h=1}^{H} \Big(\sum_{k \in S} W_{h,\cdot}^l z_k^{l-1} \Big) \cdot q_{hj}^l e_j \\ =& y^{l-1} + \sum_{j=1}^{J} \frac{1}{HS} \sum_{h=1}^{H} \sum_{k=1}^{J} W_{hk}^l q_{hj}^l y_k^{l-1} e_j \\ =& y^{l-1} + \sum_{j=1}^{J} \sum_{k=1}^{J} \Theta_{jk}^l y_k^{l-1} e_j \\ =& y^{l-1} + \Theta^l (y^{l-1} \odot e_S) \end{aligned}$$

as desired.

A.2.2 Universal approximation

We consider a flexible model by setting $y^l \in \mathbb{R}^{J'}$ with $J' > J$. Let $\Theta^1 \in \mathbb{R}^{J' \times J}$, $\Theta^l \in \mathbb{R}^{J' \times J'}$ for $l = 2, \ldots, L$ and $W^l \in \mathbb{R}^{J \times J'}$.

$$\begin{cases} y^1 = \Theta^1 e_S, \\ y^l = y^{l-1} + \Theta^l \big(y^{l-1} \odot y^1 \big), \quad l = 2, \ldots, L. \end{cases}$$

We show inductively that there exists $W^l \in \mathbb{R}^{J \times J'}$ such that $W^l y^l$ can represent any up to l-th order interactions.

We first consider the base case $l = 1$. By definition of y^1,

$$y_j^1 = \sum_{k=1}^{J} \Theta_{jk}^1 \mathbb{I}(k \in S).$$

For $j \in S$, we can further write

$$\begin{aligned} y_j^1 =& \Theta_{jj}^1 \mathbb{I}(j \in S) + \sum_{k=1:k \neq j}^{J} \Theta_{jk}^1 \mathbb{I}(k \in S) \\ =& v_j^1(\varnothing) \mathbb{I}(j \in S) + \sum_{k=1:k \neq j}^{J} v_j^1(\{k\}) \mathbb{I}(k \in S), \end{aligned}$$

where $v_j^1(\varnothing) =: \Theta_{jj}^1$, $v_j^1(\{k\}) := \Theta_{jk}^1$. For $j > J$, it can be set to any function value parameterized by $\Theta_{j,\cdot}^1 e_S$ provided that S is not empty, which is independent of other y_j^{l-1}'s with $j \leq J$. Without loss of generality, we can simply set $W^1 = [I_J; 0] \in \mathbb{R}^{J \times J'}$ to output $W^1 y^1 = \{y_j^1\}_{j \in S}$, which represent any first-order interaction model.

Now suppose $W^{l-1} y_j^{l-1}$ can represent any interaction model up to $(l-1)$-th order with $W^{l-1} = [I_J; 0] \in \mathbb{R}^{J \times J'}$, which means for any $j = 1, \ldots, J$,

$$y_j^{l-1} = \sum_{T \subseteq \{1, \ldots, J\}, |T| \leq l-1} v_j^{l-1}(T) \prod_{k \in T} \mathbb{I}(k \in S).$$

and for any $j > J$, y_j^{l-1} can be set to any function value independent of other y_j^{l-1}'s with $j \leq J$.

For $j = 1, \ldots, J'$,

$$\begin{aligned} y_j^l =& y_j^{l-1} + \sum_{j'=1}^{J'} \Theta_{jj'}^l y_{j'}^{l-1} \sum_{k=1}^{J} \Theta_{j'k}^1 \mathbb{I}(k \in S) \\ =& y_j^{l-1} + \sum_{j'=1}^{J'} \Theta_{jj'}^l \sum_{T \subseteq \{1, \ldots, J\}, |T| \leq l-1} v_j^{l-1}(T) \prod_{k \in T} \mathbb{I}(k \in S) \sum_{k'=1}^{J} \Theta_{j'k'}^1 \mathbb{I}(k' \in S) \\ =& \bar{y}_j^{l-1} + \sum_{|T| \subseteq \{1, \ldots, J\}, k' \in \{1, \ldots, J\}, |T| \cup \{k\} = l} \sum_{j'=1}^{J'} \Theta_{jj'}^l \Theta_{j'k'}^1 v_j^{l-1}(T) \prod_{k \in T \cup \{k'\}} \mathbb{I}(k' \in S), \end{aligned}$$

where $\bar{y}_j^{l-1}$ is another function that can capture any up to $l-1$-th order interaction by combining original y_j^{l-1} with the terms in the summation where $k \in T$. Since $\Theta^l_{jj'}$ is independent of $v_j^{l-1}(T)$, we can identify $\hat{v}_j^l(T \cup \{k\}) = \sum_{j'=1}^{J'} \Theta^l_{jj'} \Theta^1_{j'k'} v_j^{l-1}(T)$ for any $T \cup \{k\} = l$, which use J' basis to approximate any l-th order interaction. Thus

$$y_j^l = \bar{y}_j^{l-1} + \sum_{T \subseteq \{1,\dots,J\}, |T|=l} \hat{v}_j^l(T) \prod_{k \in T} \mathbb{I}(k \in S).$$

With $W^l = [I_J; 0] \in \mathbb{R}^{J \times J'}$, $W^l y^l = [y_j^l]_{j \in S}$ approximate any up to l-th order interactions. By induction, y_j^L can capture up to L-th order interaction with sufficiently large J', where W^L can be set to $[I_J; 0] \in \mathbb{R}^{J \times J'}$ without loss of generality.

A.2.3 Quadratic activation

Here we explain why the formulation (10) needs at most $\lceil 1 + \log_2(J-1) \rceil$ layers to capture all orders of interactions of the J alternatives.

Starting $l = 1$, given $y^0 = e_S$,

$$y^1 = e_S + \Theta^1(e_S \odot e_S) = e_S + \Theta^1 e_S,$$

thus

$$y_j^1 = \mathbb{I}(j \in S) + \sum_{k=1}^{J} \Theta^1_{jk} \mathbb{I}(k \in S) =: v_j^1(\varnothing)\mathbb{I}(j \in S) + \sum_{k=1:k\neq j}^{J} v_j^1(\{k\})\mathbb{I}(k \in S),$$

which contains up to first-order interaction.

Now consider $l = 2, \dots, L$. Assume y^{l-1} contains up to p-th order interaction such that

$$y_j^{l-1} := v_j^{l-1}(\varnothing)\mathbb{I}(j \in S) + \sum_{\{j_1\} \subseteq S \setminus \{j\}} v_j^{l-1}(\{j_1\})\mathbb{I}(j_1 \in S) + \dots + \sum_{\{j_1,\dots,j_p\} \subseteq S \setminus \{j\}} v_j^{l-1}(\{j_1,\dots,j_p\}) \prod_{i=1}^{p} \mathbb{I}(j_i \in S).$$

Then the term

$$y_j^l = y_j^{l-1} + \sum_{k=1}^{J} \Theta^l_{jk} (y_j^{l-1})^2.$$

can contain a high-order term for $j_1' \neq \dots \neq j_p' \neq j_1' \neq \dots \neq j_p'$,

$$\sum_{k=1}^{J} \Theta^l_{jk} v_j(\{j_1,\dots,j_p\}) v_j(\{j_1',\dots,j_p'\}) \prod_{i=1}^{p} \big(\mathbb{I}(j_i \in S)\mathbb{I}(j_i' \in S)\big),$$

which is a $2p$-th order interaction. By induction, with L recursion, the model contains up to 2^{L-1}-th order interactions. Therefore, to represent the full order interaction for J products with up to $J-1$-th order interaction, we need at most $\lceil 1 + \log_2(J-1) \rceil$ recursions.

A.3 Derivation in Section 4.3

In this section, we derive the identifiability property for the parameter α as defined in (11). Recall $\mathcal{S}$ denotes the full set of products. Suppose for a fixed product pair $\{i, j\} \subseteq \mathcal{S}$, the dataset contains observed market shares for all choice sets in the collection

$$E = \{\{i, j\} \cup B : B \subseteq \mathcal{S} \setminus \{i, j\}\}.$$

We claim that the system of equations defined by (1) is square and full-rank such that the relative halo effect parameters $\{\alpha_{jk}(T)\}_{B \subseteq \mathcal{S} \setminus \{i,j\}}$ are uniquely identifiable from the observed data.

To prove this claim, we need to prove that the number of new variables, denoted as N_α, is the same as the number of equations associated with them (from the choice probability of each choice set), denoted as N_{ep}, and

$$N_{ep} = \sum_{q=2}^{n} \binom{n}{q} \times (q-1). \tag{13}$$

Note that the summation $\sum_{q=2}^{n} \binom{n}{n-q} = \sum_{q=2}^{n} \binom{n}{q}$ is the number of subset of all items whose size is smaller than $n-2$. These sets can be the source set T of halo effect difference α, since there is at least one pair of items out of these sets. Suppose the number of items in the subsets is q. We only need $q-1$ halo effect differences to calculate all pair-wise differences within q items. Therefore,

$$N_\alpha = N_{ep} = \sum_{q=2}^{n} \binom{n}{q} \times (q-1). \tag{14}$$

Furthermore, prior work such as Batsell and Polking [1985a] and Park and Hahn [1998b] has shown that the corresponding coefficient matrix is full-rank, as the system represents a saturated model. This guarantees the linear independence of equations, and hence, the parameters α are identifiable.

B Implementation Details

B.1 Featureless Models

Model Width In our formulation (Equation 9 and Equation 10), we initially constrain the model width to match the size of the item universe J, setting the shape of Θ^l to $\mathbb{R}^{J\times J}$. This aligns our model with prior work, such as the low-rank Halo MNL model [Ko and Li, 2024] and contextual MNL [Yousefi Maragheh et al., 2020a].

While a width of J suffices to capture zeroth- and first-order Halo effects, it is inefficient for higher-order interactions. As the number of possible interactions grows exponentially with order, deeper layers must activate and compose increasingly complex patterns from lower-order terms. However, under a fixed width, scaling model capacity necessitates increasing depth, which complicates optimization and training stability.

To alleviate this limitation, we expand the width of the first layer to $\Theta^1 \in \mathbb{R}^{J'\times J}$, and all subsequent layers to $\Theta^l \in \mathbb{R}^{J'\times J'}$ for $l \geq 2$. After the final layer, we project y^L back to dimension J using a linear transformation $W^L \in \mathbb{R}^{J\times J'}$. By choosing $J' > J$, we effectively increase model capacity. We refer to J' as the model width. In essence, this over-parametrizes lower-order representations to better facilitate the modeling of higher-order effects. We refer the readers to more details about this in Appendix A.2.2.

B.2 Feature-based Models

Non-Linear Embedding χ In our implementation, the non-linear embedding function χ is defined as a three-layer multilayer perceptron (MLP) with ReLU activations after each hidden layer and a final layer normalization. The network maps the input $x \in \mathbb{R}^{d_x}$ to an embedding space of dimension d:

$$\chi(x) = \text{LayerNorm}\left(W_3 \cdot \text{ReLU}\left(W_2 \cdot \text{ReLU}\left(W_1 x + b_1\right) + b_2\right) + b_3\right), \quad x \in \mathbb{R}^{d_x} \tag{15}$$

where $W_1 \in \mathbb{R}^{d\times d_x}$, $W_2, W_3 \in \mathbb{R}^{d\times d}$, and $b_1, b_2, b_3 \in \mathbb{R}^d$. This embedding function is shared by all items in the choice set.

Head-specific Nonlinear Transformations ϕ_h^l We implement the head-specific non-linear transformation ϕ_h^l as a two-layer MLP. The first layer is specific to each head h, while the second (output) layer is shared across all heads. A LayerNorm is applied after the second layer to stabilize training.

$$\phi_h^l(z_j^0) = \text{LayerNorm}\left(W_{\text{shared}}^l \cdot \text{ReLU}(W_h^l z_j^0 + b_h^l) + b_{\text{shared}}^l\right), \quad z_j^0 \in \mathbb{R}^d \tag{16}$$

where $z_j^0 = \chi(x_j) \in \mathbb{R}^d$, $W_h^l \in \mathbb{R}^{d\times d}$, $W_{\text{shared}}^l \in \mathbb{R}^{d\times d}$, and $b_h^l, b_{\text{shared}}^l \in \mathbb{R}^d$.

Dummy Items To enable parallel computation and efficient model training, all choice sets are padded to a uniform size. We pad dummy items using zero vectors in the input space, i.e., $\mathbf{0}^{d_x}$, and explicitly enforce their latent representations to remain inactive throughout the network. Specifically, after each residual layer, we reset $Z_j^l = \mathbf{0}^d$ for any dummy item j, ensuring that dummy items do not accumulate or propagate any signal across layers. Since σ is defined as an element-wise polynomial activation function, it satisfies $\sigma(\mathbf{0}) = \mathbf{0}$, ensuring that the aggregation process Equation 6 remains unaffected by dummy items. In the final softmax mapping, we set all dummy items' utility value to be $-inf$. This mechanism guarantees that dummy items do not affect contextual aggregation or final predictions, thereby preserving the integrity of the learned representations.

C Empirical Study Details

C.1 Featureless Experiments

C.1.1 Synthetic Data Experiments

Data Generation In this experiment, we directly sample the choice probability vector uniformly from the 15-dimensional probability simplex $\mathbb{R}^{15}$, rather than sampling each individual halo effect from a distribution—such as the standard normal—and then computing the corresponding choice probabilities. While the latter approach may seem more intuitive or realistic, it would require evaluating the choice probabilities exponentially many times due to the complex structure of the choice sets, making it computationally inefficient. Importantly, any choice data can be represented by an appropriate system of halo effects [Batsell and Polking, 1985b], implying that it is not necessary to explicitly specify the underlying halo effects. This justifies the feasibility and practicality of our direct sampling approach.

Experiments Setup We conduct two sets of experiments under parameter budgets of 200k and 500k, respectively. For each budget, we vary the model depth from 3 to 7. The batch size and learning rate are fixed to be 1024 and 1×10^{-4}. All models are trained for 500 epochs. The detailed configurations and corresponding training results are summarized below (see Table 3). All experiments, including hypothetical data experiments, are conducted on a single Google Colab T4 GPU with Adam optimizer.

Table 3: Model configurations with depth, width, parameter count, and in-sample training RMSE.

Group	Depth	Width	Param Count	Training RMSE
200k	3	306	200.736k	0.0434
	4	251	200.298k	0.0243
	5	218	200.124k	0.0202
	6	195	199.290k	0.0188
	7	179	200.838k	0.0183
500k	3	489	499.758k	0.0419
	4	401	500.448k	0.0156
	5	348	500.424k	0.0140
	6	312	501.384k	0.0131
	7	285	501.030k	0.0130

In these experiments, we use the mean squared error (MSE) between the predicted choice probabilities and the ground-truth one-hot choice vectors as the training objective. This allows us to use training RMSE as a proxy for evaluating the model's approximation error. Note that the training RMSE reported in Table 3 and Figure 2 is not computed directly from the loss. Instead, for each choice set, we first compute the empirical choice frequency vector and then measure the RMSE between this vector and the model's predicted choice probabilities. A perfect fit would result in an RMSE of zero. This evaluation method highlights the expressiveness of the model, as it directly reflects how well the predicted distributions capture the underlying variability in observed choices, beyond merely fitting to individual samples.

C.1.2 Real Dataset Experiments

Hotel From our experimental results, the contextual effects are more pronounced in the second hotel, suggesting that user choices in this setting are more strongly influenced by the composition of the choice set. As a result, we select the second hotel as a case study to compare our method against baseline models, in order to better evaluate their ability to capture such contextual dependencies.

SFOwork and SFOshop The **SFOwork** dataset includes six transportation modes: driving alone, shared ride (2), shared ride (3+), bike, transit, and walk. The **SFOshop** dataset comprises eight transportation modes: SharedRide (2+) and DriveAlone, SharedRide (2/3+), DriveAlone, SharedRide (3+), walk, transit, SharedRide (2), and bike. These datasets originate from Koppelman and Bhat [2006], and we use the preprocessed versions provided by Seshadri et al. [2019].

Table 4: DeepHalo hyper-parameters and training settings for experiments on Hotel, SFOshop, and SFOwork datasets.

Hyper-parameter	**Hotel**	**SFOshop**	**SFOwork**
Model Width J'	3	20	20
Layers L	4	5	5
Polynomial Activation σ	Quadratic	Quadratic	Quadratic
Batch size	Full Batch	256	256
Learning Rate	1×10^{-3}	1×10^{-4}	1×10^{-4}

Experiments Setup For the HOTEL dataset, due to its small size and the absence of a dedicated validation split, we train the model using a full-batch setting for 300 epochs without early stopping. To mitigate overfitting, we adopt a compact model configuration with reduced width and fewer parameters. In contrast, for experiments on the SFO datasets (SFOSHOP and SFOWORK), we tune the number of layers $L \in \{4, 5\}$ and the intermediate width $J' \in \{10, 20\}$ based on validation performance. Early stopping is applied with a patience of 10 epochs to prevent overfitting. All experiments are conducted on a single Google Colab T4 GPU with Adam optimizer.

Reproducibility Remark We observe that across all three datasets, the experimental results are highly stable: the standard deviations of both NLL and accuracy over 5 runs are consistently close to zero. Moreover, top-1 accuracy, while intuitive, shows limited sensitivity in reflecting model performance differences due to the relative simplicity of the tasks. Therefore, for brevity and clarity, we omit the full multi-run result tables from the appendix. We have, however, ensured that all reported findings are representative and reproducible.

C.2 Feature-based Experiments

C.2.1 Datasets Details

Expedia Dataset For data preprocessing, we generally follow the procedures outlined in Aouad et al. [2021]. Specifically, we one-hot encode the categorical features `site_id`, `visitor_location_country_id`, `prop_country_id`, and `srch_destination_id`, grouping all categories with fewer than 1,000 occurrences into a special category labeled `-1`. Continuous features such as `price_usd` and `srch_booking_window` are filtered to remove unrealistic values: we retain only searches with hotel prices between \$10 and \$1,000 and booking windows shorter than one year. A logarithmic transformation is applied to both features to reduce skewness. Missing values across the dataset are imputed using the placeholder value `-1`.

We further filter out all records where no hotel was chosen, as such cases are not informative for modeling discrete choice and constitute a large portion of the raw data. To handle dummy items introduced for padding, we assign all input features of these items to zero vectors. After preprocessing, we obtain a dataset consisting of 275,609 transaction observations, each described by 35 item-specific features and 56 shared features.

LPMC Dataset The LPMC dataset is preprocessed by constructing both item-specific and item-shared features for the four available transportation modes: walk, cycle, public transit, and drive. Each alternative is represented by a 4-dimensional item-specific feature vector that includes duration

(e.g., walking time, cycling time, or transit access/rail/bus/interchange time for public transit), cost (such as fuel prices, transit fare, or congestion charges), number of interchanges (for transit), and a congestion level indicator (for driving).

In addition, we incorporate a set of shared features that are common across all items within a choice set. These include numerical variables such as straight-line distance, user age, gender (binary), driver's license status (binary), and the number of cars owned. Categorical variables like day of the week and trip purpose are encoded using one-hot representations. After preprocessing, the final dataset includes 8 item-specific features and 17 shared features for each sample.

C.2.2 Experiments Details

Experiments Setup We tune the hyperparameters of DeepHalo based on validation performance. Specifically, we search the number of layers $L \in \{4, 5\}$, embedding dimensions $\mathrm{d} \in \{32, 64\}$, and hidden dimensions $H \in \{8, 16\}$. For all configurations, we adopt early stopping with a patience of 10 epochs to prevent overfitting. On the LPMC dataset, we further employ a learning rate scheduling strategy: training is first conducted with a learning rate of 10^{-3}, and then fine-tuned using a smaller rate of 10^{-4}. This two-phase training helps improve convergence stability and final model performance. More details are shown in Table.5. All experiments are conducted on a single Google Colab T4 GPU with Adam optimizer.

Table 5: DeepHalo hyper-parameters and training settings on Expedia and LPMC Experiments.

Hyper-parameter	**Expedia**	**LPMC**
Embedding dimension d	32	32
Hidden size H	8	8
Layers L	5	4
Polynomial Activation σ	Hadamard product	Hadamard product
Batch size	256	256
Learning Rate	1×10^{-3}	$1 \times 10^{-3}/1 \times 10^{-4}$

Baseline Information We summarize below the key configurations and model architectures for each baseline used in our experiments.

TCNet adopts a single-layer Transformer encoder-decoder architecture with multi-head self-attention and feedforward layers. Both the source and target inputs are linear projections of item features. For the LPMC dataset, we use an embedding dimension of $d = 36$ and $K = 6$ attention heads; for the Expedia dataset, we set $d = 64$ and $K = 8$. The decoder output for each item is passed through a final linear layer to obtain utility scores.

RUMnet models sequential utility construction using a GRU that processes items in their presented order within each choice set. At each timestep, the GRU updates a latent preference state, from which the utility for the current item is computed using a linear transformation. Additionally, an item-specific bias term, learned from raw features, is added to the final score. The hidden size of the GRU is set to 96 for LPMC and 128 for Expedia to accommodate the respective feature complexities.

TasteNet implements a latent factor model with learned item and user representations. Both the item encoder and the user encoder are two-layer multilayer perceptrons (MLPs) with a hidden size of 128 and output dimension 128. The user embedding is obtained by averaging valid item feature vectors within a choice set. Final utilities are computed as dot products between item and user embeddings, with an item-specific bias added.

FATEnet uses a DeepSet architecture where the embedding dimension matches the input size d_x. Both the embedding network and the pairwise utility network are implemented as five-layer MLPs with 64 hidden units per layer and ReLU activations. The pairwise utility network outputs a scalar utility score for each item-context pair.

MLP is a simple three-layer MLP with ReLU activations and a default hidden width of $d_{\text{embed}} = 128$. It maps each item feature vector to a scalar utility, followed by a masked softmax normalization over available alternatives. The model does not include any residual connections or context aggregation mechanisms.

ResLogit encodes each item through a three-layer MLP with embedding dimension $d = 32$, using ReLU activations and dropout. The encoded features are passed through a layer normalization layer and then linearly projected via a learned coefficient vector $\beta \in \mathbb{R}^d$. The resulting utility scores are refined through a 10-layer residual network consisting of nonlinear residual blocks based on softplus activations, followed by masked softmax.

DLCL is a context-aware linear model in which item utilities are computed via a learned matrix B modulated by context-dependent adjustments from a second matrix A, based on the average feature vector of the entire choice set. The outputs from different feature dimensions are combined using a set of learnable mixture weights to produce the final utility distribution.

Detailed Results We run each experiment with different random seeds and report the mean and standard deviation of the evaluation metrics from 5 repetitions. This improves statistical robustness and accounts for variance due to initialization and stochastic training dynamics. We evaluate model performance using negative log-likelihood (NLL) and top-1 accuracy, where accuracy is defined as the proportion of cases where the model assigns the highest predicted probability to the actually chosen item. Unless otherwise specified, the main results reported in the paper correspond to a single representative run. The summary of detailed results is shown in Table 6 and Table 7.

Table 6: Expedia Detailed Results

Model	Train		Validation		Test	
	Loss	Acc	Loss	Acc	Loss	Acc
DeepHalo	2.5086 ± 0.0051	0.2501 ± 0.0011	**2.5216 ± 0.0025**	**0.2474 ± 0.0007**	**2.5288 ± 0.0026**	**0.2442 ± 0.0009**
TCnet	**2.4965 ± 0.0125**	**0.2519 ± 0.0034**	2.5260 ± 0.0051	0.2476 ± 0.0006	2.5343 ± 0.0041	0.2423 ± 0.0005
MLP	2.5693 ± 0.0030	0.2362 ± 0.0008	2.5695 ± 0.0011	0.2371 ± 0.0009	2.5805 ± 0.0015	0.2304 ± 0.0011
FATEnet	2.5460 ± 0.0067	0.2407 ± 0.0015	2.5440 ± 0.0046	0.2421 ± 0.0008	2.5548 ± 0.0057	0.2365 ± 0.0008
ResLogit	2.5540 ± 0.0044	0.2401 ± 0.0015	2.5577 ± 0.0015	0.2396 ± 0.0011	2.5692 ± 0.0028	0.2342 ± 0.0008
RUMnet	2.5565 ± 0.0041	0.2397 ± 0.0016	2.5685 ± 0.0011	0.2358 ± 0.0009	2.5782 ± 0.0026	0.2308 ± 0.0021
TasteNet	2.5623 ± 0.0081	0.2383 ± 0.0016	2.5675 ± 0.0036	0.2374 ± 0.0007	2.5766 ± 0.0040	0.2326 ± 0.0008
DLCL	2.5624 ± 0.0009	0.2357 ± 0.0002	2.5546 ± 0.0010	0.2391 ± 0.0005	2.5621 ± 0.0010	0.2307 ± 0.0002
MNL	2.6272 ± 0.0001	0.2260 ± 0.0002	2.6160 ± 0.0001	0.2275 ± 0.0002	2.6245 ± 0.0001	0.2210 ± 0.0003

Table 7: LPMC Detailed Results

Model	Train		Validation		Test	
	Loss	Acc	Loss	Acc	Loss	Acc
DeepHalo	**0.6427 ± 0.0122**	**0.7527 ± 0.0037**	**0.6412 ± 0.0053**	**0.7576 ± 0.0028**	**0.6407 ± 0.0045**	**0.7552 ± 0.0022**
TCnet	0.6744 ± 0.0098	0.7422 ± 0.0036	0.6581 ± 0.0096	0.7500 ± 0.0038	0.6577 ± 0.0076	0.7468 ± 0.0023
MLP	0.7036 ± 0.0034	0.7333 ± 0.0019	0.6855 ± 0.0029	0.7402 ± 0.0019	0.6870 ± 0.0049	0.7367 ± 0.0030
FATEnet	0.6765 ± 0.0042	0.7413 ± 0.0015	0.6583 ± 0.0046	0.7508 ± 0.0018	0.6619 ± 0.0043	0.7457 ± 0.0029
ResLogit	0.7080 ± 0.0047	0.7293 ± 0.0025	0.6926 ± 0.0057	0.7348 ± 0.0022	0.6915 ± 0.0052	0.7370 ± 0.0017
RUMnet	0.6923 ± 0.0074	0.7378 ± 0.0017	0.6754 ± 0.0079	0.7460 ± 0.0025	0.6779 ± 0.0067	0.7411 ± 0.0020
TasteNet	0.7140 ± 0.0056	0.7292 ± 0.0031	0.6976 ± 0.0056	0.7356 ± 0.0032	0.6963 ± 0.0068	0.7348 ± 0.0035
DLCL	0.7148 ± 0.0027	0.7151 ± 0.0013	0.6978 ± 0.0016	0.7266 ± 0.0018	0.6988 ± 0.0026	0.7218 ± 0.0011
MNL	0.8813 ± 0.0000	0.6312 ± 0.0002	0.8621 ± 0.0000	0.6426 ± 0.0003	0.8637 ± 0.0001	0.6441 ± 0.0002