arXiv:submit/7407070 [stat.ME] 25 Mar 2026

Amortized Inference for Correlated Discrete Choice Models via Equivariant Neural Networks

Easton Huch* Michael Keane†

March 2026

Abstract

Discrete choice models are fundamental tools in management science, economics, and marketing for understanding and predicting decision-making. Logit-based models are dominant in applied work, largely due to their convenient closed-form expressions for choice probabilities. However, these models entail restrictive assumptions on the stochastic utility component, constraining our ability to capture realistic and theoretically grounded choice behavior—most notably, substitution patterns. In this work, we propose an amortized inference approach using a neural network emulator to approximate choice probabilities for general error distributions, including those with correlated errors. Our proposal includes a specialized neural network architecture and accompanying training procedures designed to respect the invariance properties of discrete choice models. We provide group-theoretic foundations for the architecture, including a proof of universal approximation given a minimal set of invariant features. Once trained, the emulator enables rapid likelihood evaluation and gradient computation. We use Sobolev training, augmenting the likelihood loss with a gradient-matching penalty so that the emulator learns both choice probabilities and their derivatives. We show that emulator-based maximum likelihood estimators are consistent and asymptotically normal under mild approximation conditions, and we provide sandwich standard errors that remain valid even with imperfect likelihood approximation. Simulations show significant gains over the GHK simulator in accuracy and speed.

Keywords: amortized inference; DeepSet; discrete choice; invariant theory; multinomial probit; neural network emulator; permutation equivariance; Sobolev training

*Postdoctoral Fellow, Johns Hopkins Carey Business School, ehuch@jhu.edu.
†Carey Distinguished Professor, Johns Hopkins Carey Business School, mkeane14@jhu.edu.

1 Introduction

Discrete choice models are a widely used tool in management science, economics, marketing, and other fields for understanding how individuals and organizations make decisions among finite sets of alternatives (McFadden, 1974). These models provide a structural framework that delivers interpretable parameters—such as willingness-to-pay and demand elasticities—and that also enables predictions about the effects of hypothetical interventions, including pricing changes, product introductions, and policy modifications.

The dominant discrete choice model in applied work is the multinomial logit (MNL), which assumes that the utility consumer i obtains from alternative j takes the form $U_j = v_j + \epsilon_j$ where v_j is the deterministic component of utility, typically written as a linear function $\boldsymbol{x}_j^\top \boldsymbol{\beta}$ of alternative j's attributes $\boldsymbol{x}_j$, and ϵ_j is a Type I extreme value (Gumbel) error that is independent and identically distributed (iid) across alternatives $j = 1, \ldots, K$. This setup yields simple closed-form choice probabilities: the probability of choosing alternative j is given by the "softmax" function $\exp(v_j)/\sum_{k=1}^{K} \exp(v_k)$. The resulting computational convenience has made MNL the default choice in countless applications.

However, this convenience comes at a well-known cost. The iid Gumbel error assumption implies the restrictive independence of irrelevant alternatives (IIA) property: the odds ratio between any two alternatives is unaffected by the presence or attributes of other alternatives, which can lead to unrealistic substitution patterns. A popular generalization of MNL is the mixed logit model (mixed-MNL), in which the utility weight $\boldsymbol{\beta}$ is allowed to be heterogeneous across consumers. Estimation is straightforward via simulation methods (Train, 2009). The mixed-MNL relaxes IIA at the aggregate level, but IIA still holds at the level of consumer i's individual choices given the corresponding random utility weight, $\boldsymbol{\beta}_i$.

The multinomial probit (MNP) is a well-known alternative to MNL that relaxes IIA and allows for flexible substitution patterns by assuming $\epsilon_1, \ldots, \epsilon_K$ come from a multivariate normal distribution with cross-alternative correlations. Despite this advantage, MNP has seen relatively limited adoption, largely due to its computational demands. MNP choice probabilities have no closed form and require evaluating multivariate normal rectangle probabilities. Estimation relies on more sophisticated simulation methods than needed for mixed-MNL: the GHK simulator in a classical setting (Geweke, 1989; Hajivassiliou and McFadden, 1998; Keane, 1994) or MCMC in a Bayesian setting (McCulloch and Rossi, 1994).

We propose a fundamentally different approach: Rather than simulating choice probabilities anew for each likelihood evaluation, we train a neural network emulator to directly approx-

imate the choice probability function. This strategy—known as amortized inference—shifts the computational burden from inference time to a one-time training phase. Once trained, the emulator provides rapid deterministic approximations to choice probabilities via simple function calls. The amortized inference framework has proven highly successful in multiple scientific domains for approximating computationally intensive simulation models (Lueckmann et al., 2019; Cranmer et al., 2020). Here, we adapt it to discrete choice models, extending beyond MNL and MNP to general correlated error distributions.

In the econometrics literature, Norets (2012) employs a related strategy, approximating the expected value function in dynamic discrete choice models via a neural network. He applies this strategy in the context of a specific parametric model, so changes in the dynamic model require modifying and retraining the neural network. In contrast, our emulator operates on the generic choice problem given only the deterministic utilities and the parameters of the error distribution, enabling modifications to the parametric model (e.g., the functional form of the deterministic utilities) without altering or retraining the emulator.

Our methodology relies on several new contributions. First, we develop a neural network architecture specifically designed for choice models. It respects their fundamental invariance properties, which we formalize as symmetries (group actions) of the choice-probability map.[1] Choice probabilities are invariant to location shifts (adding a constant to all utilities) and scale transformations (multiplying utilities by a positive constant), and equivariant with respect to permutations of choice alternatives. Our architecture incorporates a preprocessing transformation that reduces the size of the feature space by enforcing location and scale invariance. The processed features are then passed to a per-alternative encoder module based on the DeepSet architecture (Zaheer et al., 2017). The output is then concatenated across alternatives and processed through equivariant layers that impose a sum-to-one constraint on the output probabilities while respecting the permutation equivariance property.

Second, we establish the theoretical foundations of our architecture. We prove that it can universally approximate choice probabilities on compact subsets of the parameter space, outside a measure-zero exceptional set. This result connects our architecture to the theory of orbit separation under group actions, extending the universal approximation results of Blum-Smith et al. (2025) on symmetric matrices to the joint space of utility vectors and centered covariance matrices. Blum-Smith et al. (2025) employ Galois theory to prove generic orbit separation, while we provide a direct proof based on an invariant reconstruction argument.

[1] If these properties are not embedded in the architecture, the NN must learn them during training. As a continuum of models generate equivalent choice probabilities, this would slow learning considerably.

Third, we establish the statistical properties of maximum likelihood estimators (MLEs) formed using an emulator approximation of the true discrete choice probabilities. We show that if the emulator approximates the true log likelihood sufficiently well—specifically, if the average approximation error is $o_p(n^{-1})$—then the emulator-based estimator inherits the consistency and asymptotic normality of the exact MLE. When this condition is not met, we show that valid inference can still be obtained via sandwich standard errors, treating the emulator as a working model in the quasi-maximum likelihood framework.

By careful design, our preprocessing transformations and architecture enforce smoothness of choice probabilities with respect to the inputs: the deterministic utilities and corresponding scale (covariance) matrix. We use Sobolev training (Czarnecki et al., 2017) to match emulator gradients to those of the log choice probabilities. Together, these design choices enable reliable use of automatic differentiation for downstream model-fitting and inference tasks.

With a pretrained emulator, generalizing from logit to probit (or any other error distribution) requires only the replacement of closed-form softmax probabilities with emulator evaluations of choice probabilities when forming the likelihood. In the probit case, given a fixed computational budget, we show via simulations that an ML estimator using our amortized inference procedure matches or exceeds the performance of one using the GHK algorithm in terms of estimation error and coverage rates. Furthermore, our approach can easily handle models other than MNP where an efficient simulation algorithm like GHK is unavailable.

For concreteness, much of our exposition focuses on MNP models. However, the emulator approach and supporting theory are largely agnostic to the assumed parametric form of the errors. Generalizing to other error distributions (e.g., correlated Gumbel) is straightforward, requiring only modest changes to the training data generation and emulator inputs.

The remainder of this paper is organized as follows. Section 2 summarizes related literature on choice modeling. Section 3 describes the MNP model and our inferential goals. Section 4 presents the emulator architecture and training procedure. Section 5 establishes the theoretical properties of the architecture and emulator-based estimators. Section 6 presents simulation results, and Section 7 concludes.

2 Related Literature

The literature on applying machine learning methods to discrete choice can be divided into two broad streams. The first stream maintains the classic random utility model (RUM) of choice, in which choice probabilities are generated by a population of rational consumer

types with different (but stable) preference orderings over the universe of choice objects, as explained by Bloch and Marschak (1960) and McFadden and Richter (1990).[2] MNL, MNP, and mixed-MNL are all members of the RUM class provided one maintains the RUM assumption that the utility of an option depends only on its own characteristics. Key papers in this strand start from this basic MNL structure and use neural networks to generalize the functional form of utility: Bentz and Merunka (2000), Sifringer et al. (2020), Wang et al. (2020), Han et al. (2022), and Singh et al. (2023). Two important recent papers extend this work: Aouad and Desir (2025) develop an architecture that implements the mixed logit model with a flexible distribution of taste heterogeneity (RUMnet), and Bagheri et al. (2025) develop another architecture that generalizes the Gumbel error assumption (RUM-NN). Both papers use softmax-smoothed sample averages to approximate choice probabilities within the loss function, resulting in increased computation time relative to pure logit-based models.

Aouad and Desir (2025) build on the generic approximation property of mixed-MNL models shown in McFadden and Train (2000). This result relies on a flexible basis expansion of the utility function and a potentially nonparametric mixing distribution. In practice, these model attributes are unknown to analysts. As emphasized in both Aouad and Desir (2025) and McFadden and Train (2000), this approximation property is not unique to Gumbel errors. Rather, it holds more generally for a larger class of hierarchical discrete choice models, including mixed-MNP models. McFadden and Train (2000) use this approximation property to justify adoption of mixed-MNL as a computationally convenient alternative to other models lacking closed-form choice probabilities, such as MNP. But this justification of mixed-MNL becomes less compelling given a practical, general-purpose alternative like the emulators we propose here.

The second stream dispenses with the RUM structure, often motivated by a desire to relax restrictive MNL assumptions like IIA and to allow more flexible substitution patterns. Some papers in this stream view discrete choice as a general classification problem that is amenable to machine learning methods. This is exemplified by Wang and Ross (2018), Lhéritier et al. (2019), Rosenfeld et al. (2020), Chen and Mišić (2022), and Chen et al. (2025). Others maintain an MNL structure at the top level—that is, choice probabilities are determined by a vector of alternative-specific utilities that enter a softmax function—but a neural net is used to construct the alternative-specific utility functions in flexible ways that deviate from RUM assumptions. For instance, the utility of alternative j is allowed to depend on

[2]Suppose there are $\bar{K}$ objects in the universe and a consumer is presented with a choice set (or "assortment") that contains $K \leq \bar{K}$ elements. A key implication of RUM is that the utility a consumer derives from product k is invariant to the choice set, ruling out context or assortment effects.

attributes of other alternatives to generate context effects. This is exemplified by Wang et al. (2021), Wong and Farooq (2021), Cai et al. (2022), Pfannschmidt et al. (2022), and Berbeglia and Venkataraman (2025).

Between these two streams, a fundamental tension emerges: While RUM models are favored for their interpretability and grounding in economic theory, the dominant RUM models in empirical work (linear MNL and mixed-MNL) enforce constraints on deterministic utilities and substitution patterns that may be too restrictive in some applications; in particular, they assume IIA at the level of individual choices. The first stream aims to address this challenge by adding additional flexibility to RUM models. However, many of these proposals maintain the assumption of independent logit errors, constraining substitution patterns and potentially biasing utility estimates. The exceptions—namely, RUMnet and RUM-NN—address substitution, but they do so at the expense of interpretability and/or computational efficiency due to their (a) (potentially) nonlinear utility functions and (b) expensive sample-average approximations of the likelihood function. The second stream abandons the RUM structure altogether, making it difficult or impossible to obtain reliable inferences for many economically meaningful quantities, such as consumer welfare, willingness-to-pay measures, demand elasticities, and substitution effects. In summary, this tension results in a tradeoff between modeling flexibility and interpretation, ranging from rigid interpretable models like MNL on one hand to flexible atheoretic classification methods on the other.

MNP models constitute a notable exception to the above tradeoff as they allow for flexible substitution patterns via *interpretable* covariance relationships. Moreover, if additional flexibility is desired, they can be extended to allow the deterministic utilities to follow the functional form of a neural network (Hruschka, 2007). Conversely, if researchers desire a more parsimonious or interpretable model, analysts can constrain the covariance structure via penalization methods as in Jiang et al. (2025) or via factor structures as we illustrate in Section 6. Despite these virtues, empirical applications of MNP are relatively sparse in the literature due, in large part, to the difficulty of evaluating MNP choice probabilities.

More broadly, MNP models are just one member of a larger class of RUM models featuring correlated error terms. Analogous to MNP, other members can be generated by assuming errors of the form $\boldsymbol{\epsilon} = \boldsymbol{\Sigma}^{1/2}\boldsymbol{\epsilon}^*$, where $\boldsymbol{\Sigma}^{1/2}$ is a square-root scale matrix and $\boldsymbol{\epsilon}^*$ is a vector of exchangeable errors. For example, $\epsilon_j^* \overset{iid}{\sim} \text{Gumbel}(0,1)$ results in a correlated Gumbel distribution with scale matrix $\boldsymbol{\Sigma}$. As another example, we could assume $\boldsymbol{\epsilon}^* \sim \text{Multivariate-}t(\mathbf{0}, \mathbf{I}, \nu)$ to capture heavy tail behavior via the degrees-of-freedom parameter $\nu > 0$ ($\mathbf{I}$ denotes an identity matrix). Bagheri et al. (2025) consider correlated error distributions formed in this

manner, resorting to expensive simulation methods to approximate the choice probabilities because an efficient algorithm (GHK) is available only in the Gaussian case.

The primary contribution of this work is an amortized inference framework that produces accurate, reusable, and computationally efficient estimates of choice probabilities for RUM models with general error distributions, including those featuring nontrivial correlation structures. The framework presents a potential resolution to the flexibility–interpretability tradeoff highlighted above, enabling practical estimation of flexible choice models without sacrificing the economically meaningful insights and parsimony of RUM models. The framework is supported by strong theoretical justification, including a universal approximation guarantee and asymptotic inference results under mild approximation conditions (see Section 5). Moreover, the framework is largely complementary to recent advances in machine learning methods, including the two streams discussed above. In many cases, these methods could be enhanced with emulator-based likelihood evaluations, resulting in flexible models with interpretable substitution patterns and manageable computational demands.

3 Problem Setup

This section summarizes the problem setup, including the invariance properties of discrete choice models and our inferential goals.

3.1 Discrete Choice Models

We consider a decision-maker choosing among K mutually exclusive alternatives. The decision-maker assigns a latent utility U_j to each alternative $j \in \{1, \dots, K\}$ and selects the alternative with the highest utility:

$$Y = \underset{j \in \{1,\dots,K\}}{\arg\max}\, U_j. \tag{1}$$

The latent utilities decompose into deterministic and stochastic components:

$$U_j = v_j + \epsilon_j, \quad j = 1, \dots, K, \tag{2}$$

where v_j is the deterministic (or systematic) utility that depends on observable characteristics, and ϵ_j is a centered random error capturing unobserved factors. As explained in Section 1, the deterministic utility typically takes a linear form $v_j = \boldsymbol{x}_j^\top \boldsymbol{\beta}$, where $\boldsymbol{x}_j$ is a vector of alternative-specific attributes and $\boldsymbol{\beta}$ is a parameter vector to be estimated.

Different distributional assumptions on the error vector $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_K)^\top$ yield different choice models. The MNL model assumes that the ϵ_j are independent and identically distributed with the Gumbel$(0, 1)$ distribution. In contrast, the MNP model assumes that $\boldsymbol{\epsilon}$ follows a multivariate normal distribution with mean zero and covariance matrix $\boldsymbol{\Sigma}$. Below, we consider correlated models in which $\boldsymbol{\epsilon} = \boldsymbol{\Sigma}^{1/2}\boldsymbol{\epsilon}^*$, where $\boldsymbol{\Sigma}^{1/2}$ is a square-root scale matrix as described in Section 2. We impose the following assumption on $\boldsymbol{\epsilon}^*$.

Assumption 1. The error vector $\boldsymbol{\epsilon}^*$ admits a density, $f_{\boldsymbol{\epsilon}^*} : \mathbb{R}^K \to [0, \infty)$, with respect to Lebesgue measure. Moreover, the elements of $\boldsymbol{\epsilon}^*$ are exchangeable so that $f_{\boldsymbol{\epsilon}^*}(\mathbf{P}_\pi \boldsymbol{\epsilon}') = f_{\boldsymbol{\epsilon}^*}(\boldsymbol{\epsilon}')$ for all $\boldsymbol{\epsilon}' \in \mathbb{R}^K$ and any permutation matrix, $\mathbf{P}_\pi$.

This setup and assumption include the MNP as a leading example with $f_{\boldsymbol{\epsilon}^*}$ equal to the PDF of the $\mathcal{N}(\mathbf{0}_K, \mathbf{I}_K)$ distribution. However, the setup is general enough to include many other possibilities, such as the correlated Gumbel and multivariate-t distributions described in Section 2. We defer treatment of additional regularity conditions until Section 5 when we develop the theoretical properties of our framework in the context of an assumed parametric model with parameter vector $\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^p$.

We denote the choice probability for alternative j as

$$P_j(\boldsymbol{v}, \boldsymbol{\Sigma}) = \Pr(U_j \geq U_k \text{ for all } k \neq j) = \Pr(\epsilon_k - \epsilon_j \leq v_j - v_k \text{ for all } k \neq j), \tag{3}$$

where $\boldsymbol{v} = (v_1, \ldots, v_K)^\top$. In practice, we may also choose to include additional parameters governing the distribution of $\boldsymbol{\epsilon}^*$, such as the degrees-of-freedom parameter, ν, described in Section 2, but these parameters are suppressed in the notation for simplicity. The choice probability can be expressed as the following integral:

$$P_j(\boldsymbol{v}, \boldsymbol{\Sigma}) = \int_{\mathcal{R}_j} f_{\boldsymbol{\epsilon}^*}(\boldsymbol{\epsilon}^*)\, d\boldsymbol{\epsilon}^*, \tag{4}$$

$$\mathcal{R}_j = \left\{\boldsymbol{\epsilon}^* \in \mathbb{R}^K : v_j + \left[\boldsymbol{\Sigma}^{1/2}\right]_{j\bullet}\boldsymbol{\epsilon}^* \geq v_k + \left[\boldsymbol{\Sigma}^{1/2}\right]_{k\bullet}\boldsymbol{\epsilon}^* \text{ for all } k \neq j\right\}, \tag{5}$$

where $\mathcal{R}_j$ is the region of the error space where alternative j is chosen and $\left[\boldsymbol{\Sigma}^{1/2}\right]_{j\bullet}$ denotes the jth row of $\boldsymbol{\Sigma}^{1/2}$ so that $\left[\boldsymbol{\Sigma}^{1/2}\right]_{j\bullet}\boldsymbol{\epsilon}^* = \epsilon_j$. In general, this integral is not analytically tractable. In MNP models, in particular, it does not have a closed-form solution for $K \geq 3$, necessitating numerical methods, such as the GHK algorithm, which simulates these probabilities via recursive conditioning. For each evaluation, GHK requires R simulation draws, with the approximation error decreasing at rate $O(R^{-1/2})$.

3.2 Invariance Properties

Choice probabilities satisfy three fundamental invariance properties.

Location invariance. Adding a constant to all utilities does not change choice probabilities:

$$P_j(\boldsymbol{v} + C\mathbf{1}_K, \boldsymbol{\Sigma}) = P_j(\boldsymbol{v}, \boldsymbol{\Sigma}) \tag{6}$$

where $\mathbf{1}_K$ is the K-vector of ones and $C \in \mathbb{R}$ may be random. This follows because choices depend only on utility differences.

Scale invariance. Scaling all utilities and the scale matrix preserves choice probabilities:

$$P_j(\alpha\boldsymbol{v}, \alpha^2\boldsymbol{\Sigma}) = P_j(\boldsymbol{v}, \boldsymbol{\Sigma}) \quad \text{for all } \alpha > 0. \tag{7}$$

This follows because the scaled model is equivalent to the original model with $\boldsymbol{v}$ and $\boldsymbol{\epsilon}$ replaced by $\alpha\boldsymbol{v}$ and $\alpha\boldsymbol{\epsilon}$, respectively.

Permutation equivariance. Relabeling alternatives permutes the choice probabilities correspondingly. For any permutation π of $\{1, \ldots, K\}$ with associated permutation matrix $\mathbf{P}_\pi$:

$$P_j(\boldsymbol{v}, \boldsymbol{\Sigma}) = P_{\pi^{-1}(j)}(\mathbf{P}_\pi\boldsymbol{v}, \mathbf{P}_\pi\boldsymbol{\Sigma}\mathbf{P}_\pi^\top). \tag{8}$$

A naive attempt at constructing an emulator may not respect these properties. Translating, rescaling, or permuting the alternatives could produce different emulator predictions, resulting in slow learning and poor generalization because the emulator would need to learn these properties manually. In Section 4, we propose a solution that automatically satisfies these properties based on two design components: a preprocessing step that fixes the location and scale, and an equivariant architecture that ensures permutation equivariance without sacrificing expressivity.

The above invariance properties imply that the choice model is not identified without normalization. We assume throughout that the model has been appropriately normalized so that the parameter vector $\boldsymbol{\theta}$ is identified. This is typically achieved in practice by setting the utility of one alternative to zero deterministically and fixing one of the nonzero diagonal elements in $\boldsymbol{\Sigma}$ (see Keane (1992)). Similarly, we assume that the data exhibit sufficient variation to identify the remaining parameters, which typically requires variation in alternative-specific covariates, such as prices.

3.3 Inferential Goals

Given observations $\{(y_i, \mathbf{X}_i)\}_{i=1}^n$, where $y_i \in \{1, \ldots, K\}$ is the observed choice and $\mathbf{X}_i$ represents the covariates, we seek to estimate the parameter vector, $\boldsymbol{\theta}$. This parameter vector includes both the coefficients affecting deterministic utilities and additional parameters governing the scale matrix.

The (averaged) log-likelihood function is

$$\ell_n(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \log P_{y_i} \left\{ \boldsymbol{v}_i(\mathbf{X}_i, \boldsymbol{\theta}), \boldsymbol{\Sigma}_i(\mathbf{X}_i, \boldsymbol{\theta}) \right\}, \tag{9}$$

where $\boldsymbol{v}_i(\mathbf{X}_i, \boldsymbol{\theta})$ and $\boldsymbol{\Sigma}_i(\mathbf{X}_i, \boldsymbol{\theta})$ denote the utility vector and scale matrix for observation i as functions of the covariates and parameters. Maximum likelihood estimation requires maximizing $\ell_n(\boldsymbol{\theta})$ over Θ, which in turn requires repeated evaluation of $\ell_n(\boldsymbol{\theta})$ and its gradient $\nabla_{\boldsymbol{\theta}} \log P_j(\boldsymbol{v}, \boldsymbol{\Sigma})$. Similarly, Bayesian methods such as Hamiltonian Monte Carlo require repeated evaluations of both $\ell_n(\boldsymbol{\theta})$ and $\nabla_{\boldsymbol{\theta}} \log P_j(\boldsymbol{v}, \boldsymbol{\Sigma})$, often thousands of times throughout the estimation process.

Our goal is to construct a neural network emulator $\hat{P}_j(\boldsymbol{v}, \boldsymbol{\Sigma}; \boldsymbol{\phi})$ that provides fast, accurate approximations of the true choice probabilities, along with analytic gradients via automatic differentiation. The emulator is trained once on simulated data spanning the relevant input space, after which it can be used for rapid inference on many datasets and models.

4 Emulator Design

This section outlines the emulator preprocessing transformations, neural network architecture, training procedure, and inference process.

4.1 Design Goals

The emulator must satisfy four desiderata:

1. **Respect invariance properties.** The emulator should respect the invariance properties described in Section 3.2: invariance with respect to location shifts and scale transformations, and equivariance with respect to permutations.

2. **Provide smooth approximations.** The emulator should be differentiable with respect to its inputs, enabling gradient-based optimization and automatic differentiation.

3. **Generalize across specifications.** A single trained emulator should work for diverse utility and covariance structures.

4. **Be computationally efficient.** Emulator evaluation should be fast enough to enable routine use in estimation.

We achieve these goals through a combination of preprocessing transformations and a carefully designed neural network architecture.

4.2 Preprocessing: Centering and Scaling

We address location and scale invariance through preprocessing transformations that project the inputs onto a canonical subspace.

Centering. Define the centering matrix

$$\mathbf{M} = \mathbf{I}_K - \frac{1}{K}\mathbf{1}_K\mathbf{1}_K^\top, \tag{10}$$

which we employ to project the vector of utilities, $\boldsymbol{U} \in \mathbb{R}^K$, onto the subspace orthogonal to $\mathbf{1}_K$, resulting in transformed utilities $\tilde{\boldsymbol{U}} = \mathbf{M}(\boldsymbol{v} + \boldsymbol{\epsilon}) = \boldsymbol{v} + \boldsymbol{\epsilon} - \mathbf{1}_K(\bar{v} + \bar{\epsilon})$, where $\bar{v} = \frac{1}{K}\sum_{j=1}^K v_j$ and $\bar{\epsilon} = \frac{1}{K}\sum_{j=1}^K \epsilon_j$. These transformed utilities sum to zero within each realization: $\sum_{j=1}^K \tilde{U}_j = 0$. In effect, this transformation results in modified emulator inputs:

$$\tilde{\boldsymbol{v}} = \mathbf{M}\boldsymbol{v} = \boldsymbol{v} - \mathbf{1}_K\bar{v}, \quad \tilde{\boldsymbol{\Sigma}} = \mathbf{M}\boldsymbol{\Sigma}\mathbf{M}^\top. \tag{11}$$

Scaling. We normalize by the trace of the centered scale matrix, producing transformed utilities $\boldsymbol{U}^* = \sqrt{K/\mathrm{tr}(\tilde{\boldsymbol{\Sigma}})}\tilde{\boldsymbol{U}}$. The final transformed parameters are then

$$\boldsymbol{v}^* = \sqrt{\frac{K}{\mathrm{tr}(\tilde{\boldsymbol{\Sigma}})}}\,\tilde{\boldsymbol{v}}, \quad \boldsymbol{\Sigma}^* = \frac{K}{\mathrm{tr}(\tilde{\boldsymbol{\Sigma}})}\,\tilde{\boldsymbol{\Sigma}}. \tag{12}$$

We can omit the case $\mathrm{tr}(\tilde{\boldsymbol{\Sigma}}) = 0$ as this results in known, deterministic choices. The complete transformation produces normalized inputs $(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*)$ satisfying:

1. $\sum_{j=1}^K v_j^* = 0$ (centered deterministic utilities),

2. $\boldsymbol{\Sigma}^*\mathbf{1}_K = \mathbf{0}$ (centered scale matrix),

3. $\text{tr}(\boldsymbol{\Sigma}^*) = K$ (scale-normalized),

4. $\boldsymbol{\Sigma}^*$ is positive semidefinite with rank at most $K - 1$.

This normalization reduces the size of the emulator's input space, accelerating learning and improving generalization. Importantly, this transformation commutes with permutation matrices: for any permutation matrix $\mathbf{P}_\pi$, applying the transformation to $(\mathbf{P}_\pi \boldsymbol{v}, \mathbf{P}_\pi \boldsymbol{\Sigma} \mathbf{P}_\pi^\top)$ yields $(\mathbf{P}_\pi \boldsymbol{v}^*, \mathbf{P}_\pi \boldsymbol{\Sigma}^* \mathbf{P}_\pi^\top)$. This ensures that the preprocessing preserves permutation equivariance.

4.3 Neural Network Architecture

After preprocessing, we construct a neural network that maps $(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*)$ to choice probabilities while maintaining permutation equivariance. The architecture consists of three components: a per-alternative encoder, permutation-equivariant layers, and an output layer. Below, we interpret $\boldsymbol{\Sigma}$ as a covariance matrix to simplify the exposition, referring to its diagonal and off-diagonal elements as variances and covariances, respectively. This interpretation is correct for MNP models. For other models, it is only illustrative as these parameters may not exactly coincide with the variances and covariances.

Per-alternative encoder. For each alternative $j \in \{1, \ldots, K\}$, we construct a representation $\boldsymbol{z}_j \in \mathbb{R}^{d_z}$ that captures how alternative j relates to all other alternatives. This representation is built from two complementary DeepSet networks—a diagonal DeepSet and an off-diagonal DeepSet—whose outputs are combined with alternative j's own features. The diagonal DeepSet processes pairwise relationships between j and each other alternative, while the off-diagonal DeepSet summarizes the covariance structure among alternatives other than j. In the descriptions below, the features labeled as "base inputs" are sufficient for universal approximation of choice probabilities (see Section 5.1). We include additional features to improve the expressivity of the emulator, as we find this improves its predictions.

Diagonal DeepSet. For each pair (j, k) with $k \neq j$, we construct a feature vector $\boldsymbol{d}_{jk}$ containing differentiable functions of the utilities and covariances associated with alternatives j and k. The base inputs are the utilities v_j^* and v_k^*, the variances Σ_{jj}^* and Σ_{kk}^*, and the covariance Σ_{jk}^*. From these, we derive additional feature transformations that facilitate learning, including standard deviations $\sigma_j = \sqrt{\Sigma_{jj}^*}$ and $\sigma_k = \sqrt{\Sigma_{kk}^*}$, the correlation $\rho_{jk} = \Sigma_{jk}^*/(\sigma_j \sigma_k)$, and the standardized utility difference (z-score):

$$z_{jk} = \frac{v_j^* - v_k^*}{\sqrt{\Sigma_{jj}^* + \Sigma_{kk}^* - 2\Sigma_{jk}^*}}. \tag{13}$$

The diagonal DeepSet processes these features using the architecture of Zaheer et al. (2017):

$$\boldsymbol{h}_j^{\text{diag}} = \rho_{\text{diag}}\left(\sum_{k\neq j}\phi_{\text{diag}}(\boldsymbol{d}_{jk})\right), \tag{14}$$

where ϕ_{diag} is a multi-layer perceptron (MLP) applied to each pair, the sum aggregates over all $k \neq j$, and ρ_{diag} is another MLP that produces a learned nonlinear representation of the sum. We use the notation ϕ_{diag} and ρ_{diag} following Zaheer et al. (2017); the MLPs ρ_{diag} and ρ_{off} (defined below) can be distinguished from the correlation ρ_{jk} by their subscripts.

Singh et al. (2023) also employ a variant of the DeepSet architecture. However, their method assumes iid errors and operates on problem-specific features, which requires their neural network to be modified and retrained to accommodate new data structures. In contrast, our architecture relies on generic RUM model inputs, such as deterministic utilities.

Off-diagonal DeepSet. For each pair (k, l) with $k < l$ and $k, l \neq j$, we construct a feature vector $\boldsymbol{o}_{kl}$ containing differentiable, symmetric functions comparing alternatives k and l; we require symmetry because the pairs do not possess a natural ordering. The base input is Σ^*_{kl}, and we include other derived features, such as the correlation ρ_{kl}, squared utility difference $(v^*_k - v^*_l)^2$, the squared z-score z^2_{kl}, and the variance sum $\Sigma^*_{kk} + \Sigma^*_{ll}$.

The off-diagonal DeepSet has an analogous structure to that of the diagonal DeepSet:

$$\boldsymbol{h}_j^{\text{off}} = \rho_{\text{off}}\left(\sum_{\substack{k<l\\k,l\neq j}}\phi_{\text{off}}(\boldsymbol{o}_{kl})\right). \tag{15}$$

Combining network. The outputs of both DeepSets are concatenated with additional "pass-through features" $\boldsymbol{s}_j$, such as v^*_j, σ_j, and summary statistics of alternative j's covariances with other alternatives. As with the additional features above, the pass-through features are not required for universal approximation, but we find that they improve performance empirically. The concatenated result is then processed by the combining MLP, ζ, to produce the per-alternative representation:

$$\boldsymbol{z}_j = \zeta\left(\boldsymbol{h}_j^{\text{diag}}, \boldsymbol{h}_j^{\text{off}}, \boldsymbol{s}_j\right). \tag{16}$$

Permutaition-equivariant layers. The per-alternative representations are stacked into a matrix $\mathbf{Z} \in \mathbb{R}^{K\times d_z}$ and processed through linear permutation-equivariant layers (Zaheer

et al., 2017), allowing information exchange across alternatives. These layers take the form

$$L(\mathbf{Z}) = \sigma\left(\mathbf{Z}\mathbf{A} + \frac{1}{K}\mathbf{1}_K\mathbf{1}_K^\top\mathbf{Z}\mathbf{B} + \mathbf{1}_K\boldsymbol{c}^\top\right), \tag{17}$$

where $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ and $\boldsymbol{c} \in \mathbb{R}^{d_{\text{out}}}$ are learnable parameters, and σ is a smooth activation function. The first term applies a per-alternative transformation, the second aggregates information across alternatives, and the third adds a shared bias (intercept).

Output layer. The final layer produces one logit per alternative:

$$\textbf{logit} = \mathbf{Z}^{(L)}\boldsymbol{a}, \tag{18}$$

where $\mathbf{Z}^{(L)}$ is the output of the last equivariant layer and $\boldsymbol{a} \in \mathbb{R}^{d_L}$ so that the output **logit** is a vector in $\mathbb{R}^K$. We omit the other terms because they produce a constant shift that does not affect the output probabilities.

Choice probabilities are obtained by applying the softmax function to the output layer:

$$\hat{P}_j = \frac{\exp(\text{logit}_j)}{\sum_{k=1}^K \exp(\text{logit}_k)}. \tag{19}$$

For any vector of valid choice probabilities $(P_1, \ldots, P_K)$ on the probability simplex Δ^{K-1}, there exists a corresponding vector of logits yielding $(P_1, \ldots, P_K)$ via softmax, so this parameterization entails no loss of generality. Figure 1 illustrates the overall architecture.

Activation functions. Throughout the architecture, we use smooth (infinitely differentiable) activation functions. This ensures that the emulator output is a smooth function of the inputs, enabling accurate gradient computation for downstream model fitting. We found that the Swish activation function $\text{Swish}(x) = x/(1+e^{-x})$ works well empirically (Ramachandran et al., 2018), though other smooth activation functions could also be applied, such as tanh or softplus (Dugas et al., 2000).

Computational complexity. A naive implementation of the per-alternative encoder would require $O(K^3)$ forward-pass operations due to the off-diagonal DeepSet computation. However, by precomputing the sum over all off-diagonal pairs and using subtraction to obtain the sum excluding pairs involving alternative j, we achieve $O(K^2)$ complexity. Because $\boldsymbol{\Sigma}$ has $O(K^2)$ unique elements influencing the choice probabilities, this scaling is optimal for general $\boldsymbol{\Sigma}$.

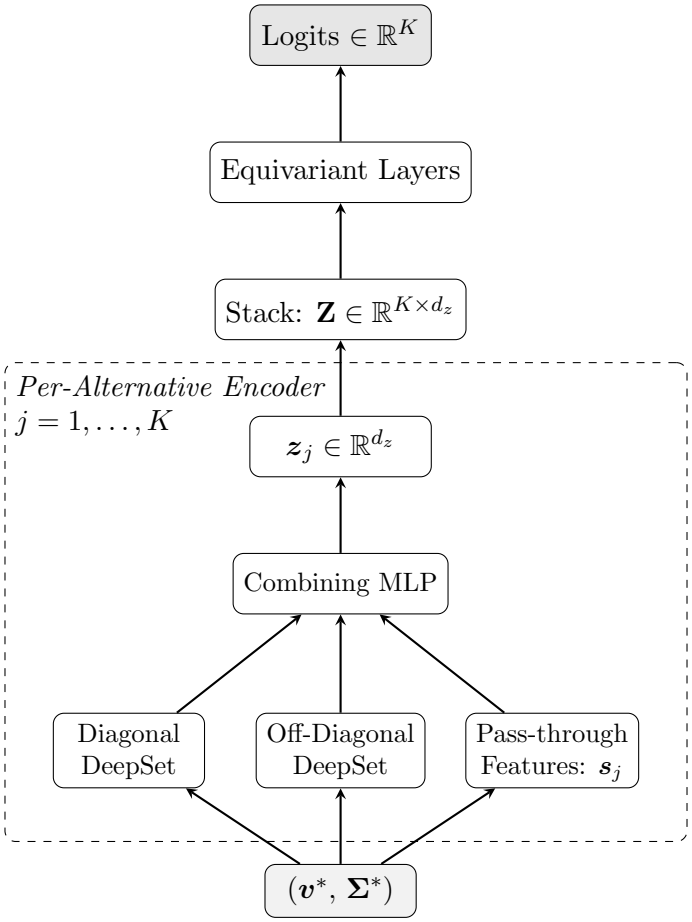


Figure 1: Architecture of the neural network emulator. For each alternative j, the per-alternative encoder processes diagonal features (relating j to each other alternative), off-diagonal features (pairwise features among alternatives other than j), and alternative j's own features. These are combined via a combining MLP to produce representation $\boldsymbol{z}_j$. Representations for all alternatives are stacked and processed through permutation-equivariant layers to produce the final logits.

Network size. The architecture can be compact. In our simulations, we use networks with 8–64 hidden units and 0–2 hidden layers per component. In general, larger values of K require larger networks for accurate approximation.

Relation to Blum-Smith et al. (2025). Our proposed architecture can be seen as an extension of the DS-CI architecture of Blum-Smith et al. (2025) with three key differences. First, whereas Blum-Smith et al. (2025) require a "linking feature" combining diagonal and off-diagonal elements, our centering procedure makes this feature redundant, simplifying the set of required inputs. Second, our problem includes an additional vector input ($\boldsymbol{v}$) not considered by Blum-Smith et al. (2025); this additional complexity requires us to couple deterministic utility values with the diagonal variance elements. Third, the equivariant nature of our problem requires us to encode each alternative separately, prompting the separation into a single distinguished alternative (j) and rival alternatives ($k \neq j$) in the per-alternative encoder.

4.4 Training Procedure

We train the emulator using simulated data spanning a diverse range of utility and covariance configurations, employing Sobolev training to ensure accurate approximation of both choice probabilities and their gradients.

Data generation. To generate each training example for our simulations in Section 6 involving the MNP model, we follow the procedure below:

1. **Error covariance.** Draw a covariance matrix $\boldsymbol{\Sigma}$ from a Wishart$(\mathbf{I}_K/(K+2), K+2)$ distribution.
2. **Raw deterministic utilities.** Draw independent utilities $\boldsymbol{v} \sim \mathcal{N}(\mathbf{0}, 4\mathbf{I}_K)$.
3. **Deterministic utility covariance.** Draw a second covariance matrix $\boldsymbol{\Sigma}'$ (independently from the same distribution as $\boldsymbol{\Sigma}$), draw $\omega \sim \text{Uniform}(0,1)$, and form the convex combination $\boldsymbol{\Sigma}'' = \omega\boldsymbol{\Sigma} + (1-\omega)\boldsymbol{\Sigma}'$.
4. **Transformed deterministic utilities.** Transform the utilities to have covariance structure correlated with $\boldsymbol{\Sigma}''$: $\boldsymbol{v}' = (\boldsymbol{\Sigma}'')^{1/2}\boldsymbol{v}$. This yields $\text{Cov}(\boldsymbol{v}') = \boldsymbol{\Sigma}''$.
5. **Preprocessing.** Apply the preprocessing transformation to obtain $(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*)$ from $(\boldsymbol{v}', \boldsymbol{\Sigma})$.
6. **Simulated choices.** Simulate 10^6 choices from the MNP model with utilities $\boldsymbol{v}^*$ and scale matrix $\boldsymbol{\Sigma}^*$ (or, equivalently, $\boldsymbol{v}'$ and $\boldsymbol{\Sigma}$).

Steps 3 and 4 ensure diversity in the relationship between deterministic utilities and error covariances. When $\omega = 1$, the utilities are perfectly aligned with the covariance structure; when $\omega = 0$, they are independent. Intermediate values produce partial correlation.

This procedure can be generalized to other error distributions by replacing the MNP model in the final step with another model with a different error distribution. Similarly, the other distributions listed and the number of choices (10^6) can be modified based on the problem domain and computational resources.

Loss function. We employ Sobolev training (Czarnecki et al., 2017), which augments the standard likelihood loss with a gradient-matching penalty. The total loss is:

$$\mathcal{L}(\boldsymbol{\phi}) = \mathcal{L}_{\text{CE}}(\boldsymbol{\phi}) + \lambda_{\text{grad}} \mathcal{L}_{\text{grad}}(\boldsymbol{\phi}), \tag{20}$$

where $\lambda_{\text{grad}} > 0$ controls the weight of the gradient-matching term.

The cross-entropy component is the multinomial negative log-likelihood (up to constants):

$$\mathcal{L}_{\text{CE}}(\boldsymbol{\phi}) = -\sum_{j=1}^{K} \hat{P}_j^{\text{sim}} \log \hat{P}_j(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*; \boldsymbol{\phi}), \tag{21}$$

where $\hat{P}_j^{\text{sim}} = n_j / \sum_{k=1}^{K} n_k$ is the simulated choice frequency for alternative j.

For the gradient-matching component, we follow the stochastic Sobolev training approach of Czarnecki et al. (2017), which avoids computing full Jacobian matrices by instead matching directional derivatives along random directions. Let $\boldsymbol{d} = (\boldsymbol{d}_v, \text{vech}(\mathbf{D}_\Sigma))$ denote a random direction in the joint space of utilities and covariances, where $\boldsymbol{d}_v \in \mathbb{R}^K$, $\mathbf{D}_\Sigma \in \mathbb{R}^{K \times K}$ is symmetric, and $\text{vech}(\cdot)$ denotes the half-vectorization (upper triangular elements including the diagonal). The gradient-matching loss is:

$$\mathcal{L}_{\text{grad}}(\boldsymbol{\phi}) = \sum_{j=1}^{K} \hat{P}_j^{\text{sim}} \left(\nabla_{\boldsymbol{d}} \log \hat{P}_j - \nabla_{\boldsymbol{d}} \log P_j^{\text{target}} \right)^2, \tag{22}$$

where $\nabla_{\boldsymbol{d}}$ denotes the directional derivative along $\boldsymbol{d}$. Weighting by the simulated choice frequencies $\hat{P}_j^{\text{sim}}$ focuses learning on alternatives that are frequently chosen, whose gradients most influence parameter estimation in downstream inference tasks.

Constraint-respecting directions. The canonical inputs $(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*)$ lie on a constrained

manifold:

$$\mathcal{X}_K = \left\{ (\boldsymbol{v}^*, \boldsymbol{\Sigma}^*) : \sum_{j=1}^{K} v_j^* = 0,\ \boldsymbol{\Sigma}^* \mathbf{1}_K = \mathbf{0},\ \mathrm{tr}(\boldsymbol{\Sigma}^*) = K,\ \boldsymbol{\Sigma}^* \succeq 0 \right\}. \tag{23}$$

Random directions must respect these constraints to remain in the tangent space of $\mathcal{X}_K$. We construct valid directions as follows:

1. Sample $\tilde{\boldsymbol{d}}_v \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_K)$ and center: $\boldsymbol{d}_v = \tilde{\boldsymbol{d}}_v - \bar{d}_v \mathbf{1}_K$, where $\bar{d}_v = \frac{1}{K}\sum_j \tilde{d}_{v,j}$.

2. Sample a symmetric matrix $\tilde{\mathbf{D}}_\Sigma$ with iid standard normal entries on and above the diagonal, then double-center:

$$\mathbf{D}'_\Sigma = \mathbf{M}\tilde{\mathbf{D}}_\Sigma\mathbf{M}, \tag{24}$$

   where $\mathbf{M} = \mathbf{I}_K - \frac{1}{K}\mathbf{1}_K\mathbf{1}_K^\top$ is the centering matrix from Section 4.

3. Remove the trace component to ensure $\mathrm{tr}(\mathbf{D}_\Sigma) = 0$:

$$\mathbf{D}_\Sigma = \mathbf{D}'_\Sigma - \frac{\mathrm{tr}(\mathbf{D}'_\Sigma)}{K-1}\mathbf{M}. \tag{25}$$

4. Set $\boldsymbol{d} = (\boldsymbol{d}_v, \mathrm{vech}(\mathbf{D}_\Sigma))$ and normalize: $\boldsymbol{d} \leftarrow \boldsymbol{d}/\|\boldsymbol{d}\|$.

Computing directional derivatives. We compute the emulator's directional derivatives via finite differences:

$$\nabla_{\boldsymbol{d}} \log \hat{P}_j \approx \frac{\log \hat{P}_j(\boldsymbol{v}^* + \epsilon\boldsymbol{d}_v, \boldsymbol{\Sigma}^* + \epsilon\boldsymbol{d}_\Sigma; \boldsymbol{\phi}) - \log \hat{P}_j(\boldsymbol{v}^* - \epsilon\boldsymbol{d}_v, \boldsymbol{\Sigma}^* - \epsilon\boldsymbol{d}_\Sigma; \boldsymbol{\phi})}{2\epsilon}, \tag{26}$$

where $\epsilon > 0$ is a small step size (we use $\epsilon = 10^{-5}$). This approach is computationally efficient, requiring only two additional forward passes per direction regardless of the input dimension. Alternatively, one could compute the directional derivative via automatic differentiation as in Czarnecki et al. (2017).

Target directional derivatives are computed from pre-stored Jacobians. During data generation, we compute and store the full Jacobians $\nabla_{\boldsymbol{\theta}} \log P_j^{\text{target}}$ with respect to $\boldsymbol{v}^*$ and $\boldsymbol{\Sigma}^*$. During training, the target directional derivative is obtained via the inner product:

$$\nabla_{\boldsymbol{d}} \log P_j^{\text{target}} = \left\langle \frac{\partial \log P_j^{\text{target}}}{\partial \boldsymbol{v}^*}, \boldsymbol{d}_v \right\rangle + \left\langle \frac{\partial \log P_j^{\text{target}}}{\partial \boldsymbol{\Sigma}^*}, \mathbf{D}_\Sigma \right\rangle_F, \tag{27}$$

where $\langle \cdot, \cdot \rangle_F$ denotes the Frobenius inner product.

Target gradient computation. Computing target gradients for Sobolev training requires differentiating through the choice simulation process. Since discrete choices are inherently non-differentiable, we employ a smooth relaxation using a temperature-scaled softmax. For each Monte Carlo draw r with utility vector $\boldsymbol{U}^{(r)} = \boldsymbol{v}^* + \boldsymbol{\epsilon}^{(r)}$, we approximate the hard choice indicator with:

$$\tilde{Y}_j^{(r)} = \frac{\exp(\tau U_j^{(r)})}{\sum_{k=1}^{K} \exp(\tau U_k^{(r)})}, \tag{28}$$

where $\tau > 0$ is a temperature parameter. As $\tau \to \infty$, this converges to the hard argmax; for finite τ, it provides a differentiable approximation. The target log-probability is then:

$$\log P_j^{\text{target}} = \log \left(\frac{1}{R} \sum_{r=1}^{R} \tilde{Y}_j^{(r)} \right), \tag{29}$$

and the target gradients $\partial \log P_j^{\text{target}} / \partial \boldsymbol{v}^*$ and $\partial \log P_j^{\text{target}} / \partial \boldsymbol{\Sigma}^*$ are obtained via automatic differentiation through this expression. This approach is closely related to the "Gumbel–softmax trick" frequently used for gradient estimation with neural networks (Jang et al., 2017; Maddison et al., 2017). We use $\tau = 3$ in our experiments, which provides a balance between gradient accuracy (low bias) and computational stability.

Training details. Our training procedure makes use of two *replay buffers*, a common technique in deep learning (Mnih et al., 2015). The first replay buffer stores precomputed training examples including inputs $(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*)$, simulated choice frequencies, and target Jacobians. The second stores random directions for the gradient-matching loss. At each training step, we sample random pairs from the replay buffers, compute the loss, and update the network parameters via stochastic gradient descent. The replay buffer accelerates training by allowing reuse of expensive high-precision simulations and gradient computations. We set the gradient penalty weight $\lambda_{\text{grad}} = 10^{-8}$ in our experiments; empirically, we find that this value provides sufficiently well-behaved derivatives.

Multi-K training. Remarkably, none of the weight vectors and matrices has a dimension depending on K. The dimension K enters the neural network only through the input data dimension; thus, we can design a single emulator capable of processing multiple values of K simultaneously. The training procedures described above can then be adapted to iterate over a range of K values, say $K \in \{3, 4, 5\}$, and the resulting emulator can then learn to approximate choice probabilities for a varying number of choices (see Section 6.3). Under this configuration, we recommend including K as an element of $\boldsymbol{s}_j$.

4.5 Inference with the Trained Emulator

Once trained, the emulator enables rapid inference. Given a new dataset $\{(y_i, \mathbf{X}_i)\}_{i=1}^n$ and candidate parameter vector $\boldsymbol{\theta}$:

1. Compute the deterministic utility vector $\boldsymbol{v}_i(\mathbf{X}_i, \boldsymbol{\theta})$ and scale matrix $\boldsymbol{\Sigma}_i(\mathbf{X}_i, \boldsymbol{\theta})$ for each observation.

2. Apply the preprocessing transformation to obtain $(\boldsymbol{v}_i^*, \boldsymbol{\Sigma}_i^*)$.

3. Evaluate the emulator to obtain estimated choice probabilities $\hat{P}_{ij}$ for all alternatives.

4. Compute the log likelihood $\hat{\ell}_n(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^n \log \hat{P}_{i,y_i}$.

As the preprocessing transformation and neural network are both smooth functions, the gradients of the emulator log likelihood with respect to $\boldsymbol{\theta}$ can be computed exactly via automatic differentiation. This enables the use of gradient-based optimizers for maximum likelihood estimation, as well as gradient-based sampling methods such as Hamiltonian Monte Carlo for Bayesian inference. The smoothness of the emulator also facilitates computation of standard errors, which can be estimated via outer products of gradients (see Section 5.2).

5 Theory

This section establishes the theoretical foundations for the emulator architecture and the statistical properties of emulator-based discrete choice estimators.

5.1 Universal Approximation

We first show that the per-alternative encoder architecture can universally approximate choice probabilities. The key insight is that the multisets processed by the encoder generically determine the input up to the relevant symmetry group. For each alternative j, define the invariant function $g_j : \mathcal{X}_K \to (\text{Multiset}, \text{Multiset})$ by:

$$\mathcal{T}_j(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*) = \{\{(v_k^*, \Sigma_{kk}^*, \Sigma_{jk}^*) : k \neq j\}\}, \tag{30}$$

$$\mathcal{O}_j(\boldsymbol{\Sigma}^*) = \{\{\Sigma_{kl}^* : 1 \leq k < l \leq K,\, k \neq j,\, l \neq j\}\}, \tag{31}$$

$$g_j(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*) = (\mathcal{T}_j(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*), \mathcal{O}_j(\boldsymbol{\Sigma}^*)). \tag{32}$$

The multiset $\mathcal{T}_j$ consists of $K-1$ triples, each coupling an alternative k's utility and variance with its covariance with alternative j. The multiset $\mathcal{O}_j$ consists of the off-diagonal covariances

among alternatives other than j. These are precisely the base inputs processed by the per-alternative encoder's diagonal and off-diagonal DeepSets.

Theorem 1 (Generic Separation)**.** *For each alternative $j \in \{1, \ldots, K\}$, there exists a closed, measure-zero set $\mathcal{B}_j \subset \mathcal{X}_K$ such that for any $(\boldsymbol{v}_1^*, \boldsymbol{\Sigma}_1^*), (\boldsymbol{v}_2^*, \boldsymbol{\Sigma}_2^*) \in \mathcal{X}_K \setminus \mathcal{B}_j$:*

$$g_j(\boldsymbol{v}_1^*, \boldsymbol{\Sigma}_1^*) = g_j(\boldsymbol{v}_2^*, \boldsymbol{\Sigma}_2^*) \implies (\boldsymbol{v}_1^*, \boldsymbol{\Sigma}_1^*) \text{ and } (\boldsymbol{v}_2^*, \boldsymbol{\Sigma}_2^*) \text{ are in the same } S_{K-1}^{(j)}\text{-orbit}, \tag{33}$$

where $S_{K-1}^{(j)}$ is the subgroup of permutations fixing alternative j.

The proof, given in Appendix A.8, proceeds in three steps:

Step 1: Genericity conditions. We define two genericity conditions on $(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*)$: (a) the triples $(v_k^*, \Sigma_{kk}^*, \Sigma_{jk}^*)$ for $k \neq j$ are pairwise distinct as elements of $\mathbb{R}^3$; and (b) for each $k \neq j$, there exists a unique subset of off-diagonal covariances that sums to $-\Sigma_{kk}^* - \Sigma_{jk}^*$. Condition (b) follows from the row-sum constraint $\boldsymbol{\Sigma}^* \mathbf{1}_K = \mathbf{0}$: the off-diagonal entries in row k (excluding Σ_{ki}) sum to $-\Sigma_{kk}^* - \Sigma_{jk}^*$.

Step 2: Recovery procedure. When both conditions hold, we can recover $(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*)$ from g_j up to relabeling of alternatives other than j. First, order the triples in $\mathcal{T}_j$ lexicographically to assign labels to alternatives. Then, for each alternative k, identify its row of off-diagonal entries as the unique subset in $\mathcal{O}_j$ with the required sum. Finally, recover shared entries Σ_{kl}^* as the singleton intersection of the subsets for rows k and l.

Step 3: Measure-zero exceptional set. The set $\mathcal{B}_j$ where either condition fails is a finite union of zero sets of nontrivial analytic functions on the affine subspace $\mathcal{X}_K$. By the principle that proper analytic subvarieties have measure zero (Mityagin, 2020), $\mathcal{B}_j$ has measure zero.

This separation result, combined with the Stone–Weierstrass theorem, yields universal approximation:

Theorem 2 (MLP Universal Approximation)**.** *Under Assumption 1, for each alternative j, the choice probability function P_j can be uniformly approximated on any compact subset of $\mathcal{X}_K \setminus \mathcal{B}_j$ by a multi-layer perceptron taking the components of g_j as input.*

Proof sketch. We apply the proof strategy of Blum-Smith et al. (2025). The choice probability P_j is continuous and $S_{K-1}^{(j)}$-invariant. By Theorem 1, the functions encoding g_j—sorted triples and sorted off-diagonal entries—separate points in the quotient space $(\mathcal{X}_K \setminus \mathcal{B}_j)/S_{K-1}^{(j)}$. Since $S_{K-1}^{(j)}$ is finite (hence compact), standard results imply that the quotient is Hausdorff

(Bredon, 1972). By the Stone–Weierstrass theorem, the algebra generated by these functions is dense in continuous functions on compact subsets of the quotient. Pulling back to $\mathcal{X}_K$ and applying the universal approximation theorem for MLPs (Cybenko, 1989; Hornik et al., 1989) yields the result. See Appendix A.8 for details. □

Corollary 1 (Architecture Universal Approximation)**.** *Let $\mathcal{B} = \cup_{j=1}^{K} \mathcal{B}_j$. Under Assumption 1, for each alternative j, the choice probability function P_j can be uniformly approximated on any compact subset of $\mathcal{X}_K \setminus \mathcal{B}_j$ by the per-alternative encoder. Further, the vector of choice probabilities $(P_1, \ldots, P_K)$ can be uniformly approximated on any compact subset of $\mathcal{X}_K \setminus \mathcal{B}$ by the neural network architecture of Section 4.3.*

Proof. Apply the proof of Theorem 2 in tandem with universal approximation results for DeepSet (Zaheer et al., 2017) and equivariant architectures (Segol and Lipman, 2020). □

The exceptional set $\mathcal{B}$ has measure zero, so it does not affect approximation in practice: for any probability distribution over $\mathcal{X}_K$ that is absolutely continuous with respect to Lebesgue measure, inputs lie in $\mathcal{B}$ with probability zero.

Corollary 1 guarantees that the per-alternative encoder alone is sufficient for universal approximation. The inclusion of the final equivariant layers ensures that the estimated probabilities sum to one and improves the expressivity of the neural network.

For large choice sets, we may opt for a simpler architecture that applies the per-alternative encoder to only the chosen alternative with no equivariant layers, resulting in a single output, P_j. This modification maintains the universal approximation property; however, it relaxes the sum-to-one constraint in return for computational efficiency.

Corollary 1 could also be extended to hold simultaneously for a finite set of K values under a suitable extension of the space $\mathcal{X}_K$ and its exceptional sets. Section 6.3 presents numerical results showing that the proposed architecture can provide accurate approximations for multiple values of K simultaneously.

5.2 Consistency and Asymptotic Normality

We now establish that emulator-based estimators inherit the asymptotic properties of exact maximum likelihood estimators under appropriate conditions.

Let $\{(y_i, \mathbf{X}_i)\}_{i=1}^{n}$ be iid observations from the true model with parameter $\boldsymbol{\theta}_0 \in \Theta$. Define the

true and emulator log-likelihood contributions as

$$m(y, \mathbf{X}; \boldsymbol{\theta}) = \log P_y \left\{ \boldsymbol{v}^*(\mathbf{X}, \boldsymbol{\theta}), \boldsymbol{\Sigma}^*(\mathbf{X}, \boldsymbol{\theta}) \right\}, \tag{34}$$

$$\hat{m}_n(y, \mathbf{X}; \boldsymbol{\theta}) = \log \hat{P}_y \left\{ \boldsymbol{v}^*(\mathbf{X}, \boldsymbol{\theta}), \boldsymbol{\Sigma}^*(\mathbf{X}, \boldsymbol{\theta}); \boldsymbol{\phi}_n \right\}, \tag{35}$$

with corresponding score functions

$$\psi(y, \mathbf{X}; \boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} m(y, \mathbf{X}; \boldsymbol{\theta}), \qquad \hat{\psi}_n(y, \mathbf{X}; \boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \hat{m}_n(y, \mathbf{X}; \boldsymbol{\theta}). \tag{36}$$

The true and emulator (averaged) log likelihoods are $\ell_n(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^n m(y_i, \mathbf{X}_i; \boldsymbol{\theta})$ and $\hat{\ell}_n(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^n \hat{m}_n(y_i, \mathbf{X}_i; \boldsymbol{\theta})$. Define the MLE and emulator-based estimator as

$$\tilde{\boldsymbol{\theta}}_n = \arg\max_{\boldsymbol{\theta} \in \Theta} \ell_n(\boldsymbol{\theta}), \qquad \hat{\boldsymbol{\theta}}_n = \arg\max_{\boldsymbol{\theta} \in \Theta} \hat{\ell}_n(\boldsymbol{\theta}). \tag{37}$$

We require that the emulator log likelihood converges to the true log likelihood.

Assumption 2 (Emulator Approximation Quality)**.** For some $\alpha \geq 0$, the emulator satisfies

$$\sup_{\boldsymbol{\theta} \in \Theta} \left| \hat{\ell}_n(\boldsymbol{\theta}) - \ell_n(\boldsymbol{\theta}) \right| = o_p(n^{-\alpha}). \tag{38}$$

The n referenced in Assumption 2 is the sample size of the observed data. Consequently, this assumption involves an asymptotic regime in which the emulator becomes increasingly accurate as $n \to \infty$, which is achievable by progressively (a) increasing the number of simulated training examples and (b) increasing the neural network complexity (i.e., the number of layers and hidden units). Appendix B.1 provides more primitive sufficient conditions for Assumption 2 with references to closely related work on convergence rates of neural networks and their usage in econometric theory. Notably, for models with analytic choice probabilities, we expect this assumption to require only $O(n^{\alpha+\delta})$ training examples for any $\delta > 0$.

We also require sufficient regularity of the true model.

Assumption 3 (Regularity)**.** The model satisfies:

(i) $\boldsymbol{\theta}_0$ is the unique maximizer of $\mathbb{E}\{m(y, \mathbf{X}; \boldsymbol{\theta})\}$ in the compact set Θ with $\boldsymbol{\theta}_0 \in \text{int}(\Theta)$ and $\mathbb{E}\{|m(y, \mathbf{X}; \boldsymbol{\theta}_0)|\} < \infty$;

(ii) $m(y, \mathbf{X}; \boldsymbol{\theta})$ is twice continuously differentiable in $\boldsymbol{\theta}$ for all $(y, \mathbf{X})$;

(iii) $\mathbb{E}\{\sup_{\boldsymbol{\theta} \in \Theta} \|\psi(y, \mathbf{X}; \boldsymbol{\theta})\|^2\} < \infty$;

(iv) $\mathbb{E}\left\{\sup_{\boldsymbol{\theta}\in\Theta}\|\nabla^2_{\boldsymbol{\theta}} m(y,\mathbf{X};\boldsymbol{\theta})\|\right\} < \infty$;

(v) the Fisher information $\mathbf{J}(\boldsymbol{\theta}_0) = \mathbb{E}\left\{\psi(y,\mathbf{X};\boldsymbol{\theta}_0)\psi(y,\mathbf{X};\boldsymbol{\theta}_0)^\top\right\}$ is positive definite.

Theorem 3 (Consistency)**.** *Under Assumptions 2 and 3,* $\hat{\boldsymbol{\theta}}_n \xrightarrow{p} \boldsymbol{\theta}_0$ *as* $n \to \infty$.

Proof. By Assumption 2 with $\alpha \geq 0$, $\sup_{\boldsymbol{\theta}} |\hat{\ell}_n(\boldsymbol{\theta}) - \ell_n(\boldsymbol{\theta})| \xrightarrow{p} 0$. Further, the regularity conditions of Assumption 3 are sufficient to apply the uniform law of large numbers (ULLN) so that $\sup_{\boldsymbol{\theta}} |\ell_n(\boldsymbol{\theta}) - \mathbb{E}\{m(y,\mathbf{X};\boldsymbol{\theta})\}| \xrightarrow{p} 0$. The triangle inequality gives uniform convergence of $\hat{\ell}_n$ to $\mathbb{E}\{m(y,\mathbf{X};\boldsymbol{\theta})\}$, which is uniquely maximized at $\boldsymbol{\theta}_0$ by Assumption 3(i). By the consistency theorem for extremum estimators (Newey and McFadden, 1994, Theorem 2.1), $\hat{\boldsymbol{\theta}}_n \xrightarrow{p} \boldsymbol{\theta}_0$. □

Under the stronger condition $\alpha \geq 1$, the emulator-based estimator is asymptotically normal.

Theorem 4 (Asymptotic Normality)**.** *Under Assumptions 2 and 3 with* $\alpha \geq 1$,

$$\sqrt{n}(\hat{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0) \xrightarrow{d} \mathcal{N}\left\{\mathbf{0}, \mathbf{J}(\boldsymbol{\theta}_0)^{-1}\right\}. \tag{39}$$

The proof, given in Appendix B.2, shows that $\hat{\boldsymbol{\theta}}_n$ is asymptotically equivalent to $\tilde{\boldsymbol{\theta}}_n$.

Remark 1 (Approximate Maximizers). If a maximum is not uniquely attained, one can work with approximate maximizers satisfying $\hat{\ell}_n(\hat{\boldsymbol{\theta}}_n) \geq \sup_{\boldsymbol{\theta}} \hat{\ell}_n(\boldsymbol{\theta}) - o_p(n^{-\alpha})$. The consistency and asymptotic normality results continue to hold; see Newey and McFadden (1994, Theorem 2.1) and Appendix B.2.

To estimate the Fisher information, define the true and emulator outer-product-of-scores estimators:

$$\mathbf{J}_n(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \psi(y_i,\mathbf{X}_i;\boldsymbol{\theta})\psi(y_i,\mathbf{X}_i;\boldsymbol{\theta})^\top, \tag{40}$$

$$\hat{\mathbf{J}}_n(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \hat{\psi}_n(y_i,\mathbf{X}_i;\boldsymbol{\theta})\hat{\psi}_n(y_i,\mathbf{X}_i;\boldsymbol{\theta})^\top. \tag{41}$$

Assumption 4 (Gradient Approximation Quality)**.** The emulator satisfies

$$\sup_{\boldsymbol{\theta}\in\Theta} \left\|\hat{\mathbf{J}}_n(\boldsymbol{\theta}) - \mathbf{J}_n(\boldsymbol{\theta})\right\| \xrightarrow{p} 0 \quad \text{as } n \to \infty. \tag{42}$$

Assumption 4 is justified by Sobolev training, which penalizes discrepancies between emulator and target gradients. The penalization is required because training on log-likelihood

values alone does not guarantee convergence of gradients. As with Assumption 2, see Appendix B.1 for more primitive sufficient conditions and additional discussion.

Proposition 1 (Consistent Estimation of Fisher Information)**.** *Under Assumptions 2, 3, and 4,* $\hat{\mathbf{J}}_n(\hat{\boldsymbol{\theta}}_n) \xrightarrow{p} \mathbf{J}(\boldsymbol{\theta}_0)$.

Proof. By the triangle inequality, $\|\hat{\mathbf{J}}_n(\hat{\boldsymbol{\theta}}_n) - \mathbf{J}(\boldsymbol{\theta}_0)\| \leq \|\hat{\mathbf{J}}_n(\hat{\boldsymbol{\theta}}_n) - \mathbf{J}_n(\hat{\boldsymbol{\theta}}_n)\| + \|\mathbf{J}_n(\hat{\boldsymbol{\theta}}_n) - \mathbf{J}(\hat{\boldsymbol{\theta}}_n)\| + \|\mathbf{J}(\hat{\boldsymbol{\theta}}_n) - \mathbf{J}(\boldsymbol{\theta}_0)\|$. The first term vanishes by Assumption 4. For the second term, the regularity conditions of Assumption 3 are sufficient to apply the ULLN to $\mathbf{J}_n$; thus, this term also converges in probability to zero. Finally, the third term vanishes by continuity of $\mathbf{J}$ and consistency of $\hat{\boldsymbol{\theta}}_n$. □

The emulator score $\hat{\psi}_n(y_i, \mathbf{X}_i; \boldsymbol{\theta})$ is computable via automatic differentiation through the smooth neural network. Standard errors for $\hat{\boldsymbol{\theta}}_n$ are obtained as the square roots of the diagonal elements of $\hat{\mathbf{J}}_n(\hat{\boldsymbol{\theta}}_n)^{-1}/n$.

An alternative estimator based on the Hessian, $\tilde{\mathbf{J}}_n = -\frac{1}{n}\sum_{i=1}^{n} \nabla^2_{\boldsymbol{\theta}} \hat{m}_n(y_i, \mathbf{X}_i; \hat{\boldsymbol{\theta}}_n)$, requires second-derivative accuracy not directly targeted by first-order Sobolev training. The score-based estimator $\hat{\mathbf{J}}_n(\hat{\boldsymbol{\theta}}_n)$ is therefore preferred.

5.3 Inference Under Misspecification

When the emulator does not perfectly approximate the true likelihood, we can still obtain valid inference by treating the emulator as a working model, following the quasi-maximum likelihood framework (White, 1982). In this section, we treat the neural network as fixed with n, dropping the n subscript from the observation log likelihood, $\hat{m}(y, \mathbf{X}; \boldsymbol{\theta})$, and score function, $\hat{\psi}(y, \mathbf{X}; \boldsymbol{\theta})$. Define the pseudo-true parameter as the population maximizer of the emulator log likelihood:

$$\boldsymbol{\theta}_0^{\dagger} = \arg\max_{\boldsymbol{\theta} \in \Theta} \mathbb{E}\left\{\hat{m}(y, \mathbf{X}; \boldsymbol{\theta})\right\}. \tag{43}$$

When the emulator closely approximates the true likelihood, $\boldsymbol{\theta}_0^{\dagger} \approx \boldsymbol{\theta}_0$. The asymptotic distribution depends on two matrices:

$$\mathbf{A}(\boldsymbol{\theta}) = -\mathbb{E}\left\{\nabla^2_{\boldsymbol{\theta}} \hat{m}(y, \mathbf{X}; \boldsymbol{\theta})\right\}, \qquad \mathbf{B}(\boldsymbol{\theta}) = \mathbb{E}\left\{\hat{\psi}(y, \mathbf{X}; \boldsymbol{\theta}) \hat{\psi}(y, \mathbf{X}; \boldsymbol{\theta})^{\top}\right\}. \tag{44}$$

Under correct specification (i.e., a perfect emulator), the information matrix equality gives $\mathbf{A}(\boldsymbol{\theta}_0) = \mathbf{B}(\boldsymbol{\theta}_0) = \mathbf{J}(\boldsymbol{\theta}_0)$. Under misspecification, these matrices differ, necessitating the sandwich covariance form. Under regularity conditions stated in Appendix B.3, we obtain

the following results.

Theorem 5 (Consistency Under Misspecification)**.** *Under Assumptions 5 and 6,* $\hat{\boldsymbol{\theta}}_n \xrightarrow{p} \boldsymbol{\theta}_0^\dagger$.

Theorem 6 (Asymptotic Normality Under Misspecification)**.** *Under Assumptions 5 and 6,*

$$\sqrt{n}(\hat{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0^\dagger) \xrightarrow{d} \mathcal{N}\left\{\mathbf{0}, \mathbf{A}(\boldsymbol{\theta}_0^\dagger)^{-1}\mathbf{B}(\boldsymbol{\theta}_0^\dagger)\mathbf{A}(\boldsymbol{\theta}_0^\dagger)^{-\top}\right\}. \tag{45}$$

The proofs follow standard M-estimator arguments; see Appendix B.4. The sandwich covariance $\mathbf{V} = \mathbf{A}(\boldsymbol{\theta}_0^\dagger)^{-1}\mathbf{B}(\boldsymbol{\theta}_0^\dagger)\mathbf{A}(\boldsymbol{\theta}_0^\dagger)^{-\top}$ can be consistently estimated by $\hat{\mathbf{V}}_n = \hat{\mathbf{A}}_n^{-1}\hat{\mathbf{B}}_n\hat{\mathbf{A}}_n^{-\top}$, where

$$\hat{\mathbf{A}}_n = -\frac{1}{n}\sum_{i=1}^{n} \nabla^2_{\boldsymbol{\theta}} \hat{m}(y_i, \mathbf{X}_i; \hat{\boldsymbol{\theta}}_n), \qquad \hat{\mathbf{B}}_n = \frac{1}{n}\sum_{i=1}^{n} \hat{\psi}(y_i, \mathbf{X}_i; \hat{\boldsymbol{\theta}}_n)\hat{\psi}(y_i, \mathbf{X}_i; \hat{\boldsymbol{\theta}}_n)^\top. \tag{46}$$

Both quantities are computable via automatic differentiation. Note that $\hat{\mathbf{B}}_n$ corresponds to $\hat{\mathbf{J}}_n(\hat{\boldsymbol{\theta}}_n)$ from the previous section under an asymptotic regime with a fixed emulator.

Theorem 6 provides valid inference for $\boldsymbol{\theta}_0^\dagger$, not necessarily $\boldsymbol{\theta}_0$. If we employ an asymptotic regime where the emulator varies with n, then the pseudo-true parameter also varies with n, so we denote it as $\boldsymbol{\theta}_n^\dagger$. When $\alpha \in [0, 1)$ in Assumption 2, we have $\boldsymbol{\theta}_n^\dagger \xrightarrow{p} \boldsymbol{\theta}_0$ as $n \to \infty$, but this convergence may be too slow relative to $\sqrt{n}$ sampling variability. The sandwich standard errors remain valid for $\boldsymbol{\theta}_n^\dagger$, but confidence intervals should not be interpreted as targeting $\boldsymbol{\theta}_0$. Even if $\mathbf{A}(\boldsymbol{\theta}_0^\dagger)$ and $\mathbf{B}(\boldsymbol{\theta}_0^\dagger)$ both approximate $\mathbf{J}(\boldsymbol{\theta}_0)$, the centering at $\boldsymbol{\theta}_n^\dagger \neq \boldsymbol{\theta}_0$ introduces asymptotic bias. Valid inference for $\boldsymbol{\theta}_0$ requires $\alpha \geq 1$ (Theorem 4).

Similarly, Proposition 1 provides conditions under which $\hat{\mathbf{B}}_n \xrightarrow{p} \mathbf{J}(\boldsymbol{\theta}_0)$. However, our first-order Sobolev training procedure provides no guarantee that the emulator Hessian will approximate the Hessian of the true model; thus, $\hat{\mathbf{A}}_n$ need not converge to any quantity from the true model. However, Theorem 6 requires only that $\mathbf{A}(\boldsymbol{\theta}_0^\dagger)$ be nonsingular, not that it match the true model. When the emulator additionally provides accurate second derivatives, the sandwich covariance matrix reduces to $\mathbf{J}(\boldsymbol{\theta}_0)^{-1}$.

6 Simulation Study

We evaluate the performance of emulator-based MNP estimators across a range of sample sizes, numbers of alternatives, and model specifications. We compare their statistical performance and computational requirements to estimators based on the GHK simulator.

6.1 Simulation Design

Model specification. We consider MNP models with $K \in \{3, 5, 10\}$ alternatives and $p = 2$ covariates. Alternative K serves as the reference with utility normalized to zero, representing an outside option. For each non-reference alternative $j \in \{1, \ldots, K-1\}$, the deterministic utility is $v_j = \boldsymbol{x}_j^\top \boldsymbol{\beta}$, where $\boldsymbol{x}_j \in \mathbb{R}^p$ is a vector of alternative-specific attributes and $\boldsymbol{\beta} = (0, 1)^\top$ is the true coefficient vector. We consider two specifications for $\boldsymbol{\Sigma}$, denoting them as Dense and Factor.

The Dense specification assumes a dense structure for $\boldsymbol{\Sigma}_{1:(K-1),1:(K-1)}$, the upper $(K-1) \times (K-1)$ submatrix of $\boldsymbol{\Sigma}$. For identification, we set $\Sigma_{11} = 1$ and $\boldsymbol{\Sigma}_{\bullet K} = \boldsymbol{\Sigma}_{K\bullet}^\top = \mathbf{0}_K$; the latter equation sets the variance and covariances of the zero-utility option to zero.

The Factor specification assumes a one-factor structure for $\boldsymbol{\Sigma}_{1:(K-1),1:(K-1)}$ of the form $\boldsymbol{\Sigma}_{1:(K-1),1:(K-1)} = \boldsymbol{\Psi} + \boldsymbol{\gamma}\boldsymbol{\gamma}^\top$, where $\boldsymbol{\gamma} \in \mathbb{R}^{K-1}$ and $\boldsymbol{\Psi} \in \mathbb{R}^{(K-1)\times(K-1)}$ is a diagonal matrix of uniquenesses. For identification, we impose an anchored normalization that fixes the first uniqueness and first factor loading:

$$\Psi_{11} = 0, \qquad \gamma_1 = 1. \tag{47}$$

As with the Dense specification, we set $\boldsymbol{\Sigma}_{\bullet K} = \boldsymbol{\Sigma}_{K\bullet}^\top = \mathbf{0}_K$. These constraints imply $\boldsymbol{\Sigma}_{11} = 1$, fixing the scale of the model, and $\boldsymbol{\Sigma}_{1j} = \gamma_j$ for $j \geq 2$, so that the factor loadings directly encode covariances with the first alternative. Given these constraints, we parameterize the model in terms of $K-2$ free uniquenesses $(\Psi_{22}, \ldots, \Psi_{K-1,K-1})$ and $K-2$ free loadings $(\gamma_2, \ldots, \gamma_{K-1})$. This anchored normalization is designed to eliminate flat directions in the likelihood surface.

Table 1 shows the number of nonredundant parameters by type for the Dense and Factor specifications. With the anchored normalization, the Factor specification is exactly identified for $K = 3$ and overidentified for $K \geq 4$. For $K = 10$, the Factor specification results in fewer than half as many parameters as the Dense specification (18 vs. 46).

Data generation. For each replication, we generate covariates $\boldsymbol{x}_{ij} \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{I}_p)$ for observations $i = 1, \ldots, n$ and alternatives $j = 1, \ldots, K-1$, with $\boldsymbol{x}_{iK} = \mathbf{0}$ for the reference alternative.

For the Dense specification, we draw the covariance matrix according to a Wishart$(\mathbf{I}_{K-1}, K+10)$ distribution and then normalize it so that $\Sigma_{11} = 1$, resulting in a probability density

Covariance ($\boldsymbol{\Sigma}$) Specification	K	Coefficients	Variances (Uniquenesses)	Covariances (Loadings)	Total
Dense	3	2	1	1	4
	5	2	3	6	11
	10	2	8	36	46
Factor	3	2	1	1	4
	5	2	3	3	8
	10	2	8	8	18

Table 1: Parameter counts by $\boldsymbol{\Sigma}$ specification (Dense or Factor) and K.

function (PDF) given by

$$\frac{\Gamma(s) \cdot |\boldsymbol{\Sigma}_{1:(K-1),1:(K-1)}|^5}{2^s \Gamma_{K-1}\left(\frac{K+10}{2}\right) \cdot \left\{\frac{1}{2}\text{tr}(\boldsymbol{\Sigma}_{1:(K-1),1:(K-1)})\right\}^s}, \tag{48}$$

where $s = (K+10)(K-1)/2$ and $\Gamma_{K-1}(\cdot)$ denotes the $(K-1)$-dimensional multivariate gamma function (see Appendix C.2).

For the Factor specification, we sample the free uniquenesses on the log scale, drawing $\log(\Psi_{jj}) \overset{iid}{\sim} \mathcal{N}(0, 0.1^2)$ for $j = 2, \ldots, K-1$, and exponentiate to obtain $\Psi_{jj} > 0$. For the factor loadings, we draw $\gamma_j \overset{iid}{\sim} \mathcal{N}(0, 0.2^2)$ for $j = 2, \ldots, K-1$. The fixed values $\Psi_{11} = 0$ and $\gamma_1 = 1$ complete the parameterization.

Given the covariates and true parameters, we compute deterministic utilities and simulate choices $y_i \in \{1, \ldots, K\}$ according to one of two data-generating processes (DGPs): Probit or Emulator. The Probit RUM generates choices from the exact MNP model, drawing errors $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ and selecting the utility-maximizing alternative. The Emulator RUM generates choices by sampling directly from the categorical distribution defined by the emulator's predicted choice probabilities. When coupled with the Emulator RUM, the emulator-based estimation method is correctly specified, enabling a comparison of emulator-based estimators under both correct specification (Emulator RUM) and approximate inference (Probit RUM).

Estimation. To ensure positive definiteness of $\boldsymbol{\Sigma}$, we estimate the Dense specification on the scale of the Cholesky factor, with the diagonal elements optimized on the log scale to enforce positivity. Similarly, we estimate the Factor specification with the free uniquenesses on the log scale, optimizing $\log(\Psi_{jj})$ for $j = 2, \ldots, K-1$.

We form estimators using two methods, Emulator and GHK, corresponding to whether we approximate the log-likelihood function using a pretrained emulator or the GHK simulator. We fit both methods to data generated by the Probit RUM; we also fit the Emulator method to data generated by the Emulator RUM to analyze its performance under correct specification. We apply the GHK method with $R \in \{10, 50, 250\}$ draws, representing low, moderate, and high computational effort.

Both methods, Emulator and GHK, perform regularized maximum likelihood estimation. The regularization consists of two components: a barrier penalty and a pseudo-prior. The barrier penalty is applied to $\log(\Psi_{jj})$ for the Factor specification and to the log-diagonal Cholesky parameters for the Dense specification. For a latent parameter η, the barrier penalty is $\lambda_{\text{reg}} \cdot \log(\eta + 8)/n$. The pseudo-prior terms take the form $\lambda_{\text{reg}} \cdot \log\{p(\boldsymbol{\theta})\}/n$, where $p(\boldsymbol{\theta})$ is the density function of $\boldsymbol{\theta}$ under the data-generating process. We add these penalties together to form a combined penalty of the form $\lambda_{\text{reg}} \cdot r(\boldsymbol{\theta})/n$, where $r(\boldsymbol{\theta})$ sums the barrier and log-prior penalties and λ_{reg} is a regularization hyperparameter.

We then estimate $\boldsymbol{\theta}$ as the maximizer of the sum of the (approximate) log likelihood and $\lambda_{\text{reg}} \cdot r(\boldsymbol{\theta})/n$. We take $\lambda_{\text{reg}} = 0.1$ so that the penalty is small and asymptotically negligible. In practice, we find that these penalties lead to more reliable convergence of the model-fitting algorithm and help address weak identification of the covariance parameters that arises when K is large and n is small. We account for these penalties in the standard error computation by estimating $\text{Cov}(\hat{\boldsymbol{\theta}}_n)$ as $\left\{n \cdot \hat{\mathbf{J}}_n(\hat{\boldsymbol{\theta}}_n) - \lambda_{\text{reg}} \cdot \nabla^2_{\boldsymbol{\theta}} r(\hat{\boldsymbol{\theta}}_n)\right\}^{-1}$. Again, the impact of the penalization is asymptotically negligible because the term $n \cdot \hat{\mathbf{J}}_n(\hat{\boldsymbol{\theta}}_n)$ dominates asymptotically.

For all methods, we maximize the (approximate) regularized log likelihood using L-BFGS optimization (Liu and Nocedal, 1989). In practice, we found it necessary to limit the minimum step size (we used 10^{-6}) and allow small decreases in the objective function to enable the optimizer to escape local minima.

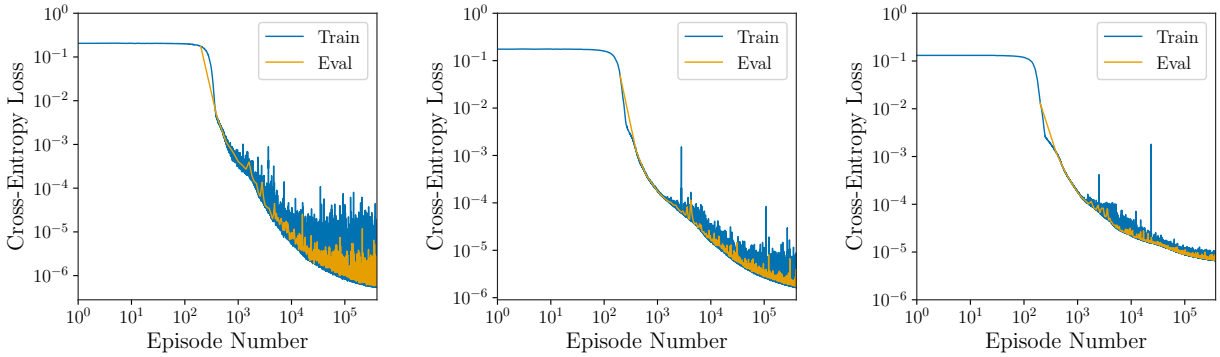
Experimental design. We consider sample sizes $n \in \{1{,}000, 10{,}000, 100{,}000\}$ and conduct 100 replications for each configuration. We compute performance metrics aggregated across all parameter types. We compute Monte Carlo standard errors for the coverage and average timings using scaled empirical standard deviations. For the remaining metrics, we apply bootstrap resampling (across repetitions) with 1,000 bootstrap samples to estimate the Monte Carlo standard errors.

Computational environment. All simulations were implemented in PyTorch (Paszke et al., 2019) and executed on a personal laptop with 48 GB RAM and an Apple M4 Pro

processor. We fit the emulators using the Apple Metal Performance Shaders (MPS). To obtain comparable estimation timings, however, we performed the rest of the simulation on CPU.

6.2 Simulation Results

We fit the emulators for 400,000 training episodes using 400,000 pregenerated training examples, each consisting of 10^6 choices. Details on the optimizer, learning rate schedule, and network architecture are provided in Appendix C.1. This model-fitting procedure completed in 2.1, 4.6, and 16.6 hours for $K = 3, 5$, and 10, respectively, representing a modest level of computational effort.



(a) $K = 3$ (b) $K = 5$ (c) $K = 10$

Figure 2: Training and evaluation loss over 400,000 training episodes for emulators with $K \in \{3, 5, 10\}$ alternatives. The training loss (blue) decreases consistently throughout training, and the evaluation loss (orange) closely tracks the training loss, indicating that the emulators do not overfit the simulated training data.

Figure 2 displays the training and evaluation loss for the emulators over time. At each value of $K \in \{3, 5, 10\}$, the training loss decreases consistently over time even at the end of training, indicating that further training would likely lead to additional improvements. The evaluation loss closely tracks the training loss, implying that the emulator generalizes well with minimal overfitting to the simulated training probabilities. We compute the evaluation loss every 200 episodes; these periodic calculations produce significantly less visible variation in the evaluation loss compared to the training loss.

Table 2 displays performance metrics for $K = 3$ comparing the performance of the emulator and GHK estimation methods. The performance metrics are the root mean squared error (RMSE), root mean squared bias (RMS Bias), ratio of estimated to empirical standard errors (SE Ratio), 95% confidence interval coverage rates (Coverage), and elapsed model-fitting time in seconds (Time (s)). The emulator methods perform well across all sample sizes,

n	DGP	Method	RMSE	RMS Bias	SE Ratio	Coverage	Time (s)
	Emulator	Emulator	0.159 (0.016)	0.010 (0.009)	1.00 (0.05)	0.94 (0.01)	0.11 (0.00)
	Probit	Emulator	0.206 (0.032)	0.049 (0.018)	0.95 (0.08)	0.95 (0.01)	0.10 (0.00)
1,000	Probit	GHK(10)	0.202 (0.031)	0.047 (0.019)	0.96 (0.08)	0.94 (0.02)	0.08 (0.00)
	Probit	GHK(50)	0.203 (0.032)	0.048 (0.018)	0.96 (0.09)	0.95 (0.01)	0.14 (0.00)
	Probit	GHK(250)	0.207 (0.033)	0.049 (0.019)	0.95 (0.08)	0.95 (0.01)	0.41 (0.01)
	Emulator	Emulator	0.052 (0.006)	0.007 (0.004)	0.94 (0.07)	0.94 (0.01)	0.36 (0.00)
	Probit	Emulator	0.053 (0.009)	0.008 (0.004)	0.99 (0.11)	0.94 (0.01)	0.36 (0.00)
10,000	Probit	GHK(10)	0.052 (0.008)	0.010 (0.003)	1.00 (0.09)	0.94 (0.01)	0.30 (0.00)
	Probit	GHK(50)	0.052 (0.009)	0.007 (0.004)	1.00 (0.10)	0.95 (0.01)	0.44 (0.01)
	Probit	GHK(250)	0.053 (0.009)	0.008 (0.005)	0.99 (0.11)	0.94 (0.01)	0.93 (0.01)
	Emulator	Emulator	0.015 (0.002)	0.001 (0.001)	0.97 (0.08)	0.94 (0.01)	1.80 (0.01)
	Probit	Emulator	0.016 (0.002)	0.001 (0.001)	1.06 (0.07)	0.98 (0.01)	1.82 (0.02)
100,000	Probit	GHK(10)	0.020 (0.002)	0.007 (0.001)	0.83 (0.06)	0.83 (0.02)	0.90 (0.01)
	Probit	GHK(50)	0.016 (0.002)	0.002 (0.001)	1.03 (0.08)	0.97 (0.01)	1.90 (0.02)
	Probit	GHK(250)	0.016 (0.002)	0.001 (0.001)	1.06 (0.07)	0.98 (0.01)	7.22 (0.09)

Table 2: Comparison of Emulator and GHK estimation methods for multinomial probit models with $K = 3$. The emulator performs competitively with GHK(50) and GHK(250) and requires computation time between that of GHK(10) and GHK(50). Performance metrics are averaged across all parameters (coefficients, variances, and covariances) with Monte Carlo standard errors shown in parentheses.

matching the performance of GHK(50) and GHK(250) in terms of statistical performance but at a lower computational cost. GHK(10) significantly undercovers at $n = 100{,}000$, exhibiting substantially higher RMS Bias than the other methods. This poor performance is likely an artifact of the GHK simulator providing biased estimates on the log scale due to Jensen's inequality; for sufficiently large n, this bias is eventually nonnegligible relative to $1/\sqrt{n}$ sampling error.

Table 3 displays similar results for $K = 5$. The emulator is competitive with GHK(50) and GHK(250) at all sample sizes. Compared to GHK(50), the compute time for the Emulator method is slightly lower for $n = 1{,}000$, approximately equal for $n = 10{,}000$, and slightly higher for $n = 100{,}000$.

Table 4 displays performance results for $K = 10$. The emulator performs comparably to GHK(250) with much lower computational requirements. At $n = 1{,}000$, the emulator fits the model in less than 3 seconds on average, outpacing even GHK(10). The computation time of the emulator falls between that of GHK(10) and GHK(50) at $n = 10{,}000$ and between GHK(50) and GHK(250) at $n = 100{,}000$. Compared to Tables 2 and 3, the standard error estimates are less reliable for the smaller sample sizes, presumably due to weak identification; the problem is amplified when we omit the penalty on $\boldsymbol{\Sigma}$. The standard errors improve

n	DGP	Method	RMSE	RMS Bias	SE Ratio	Coverage	Time (s)
	Emulator	Emulator	0.243 (0.017)	0.032 (0.012)	1.11 (0.06)	0.94 (0.01)	0.41 (0.01)
	Probit	Emulator	0.281 (0.020)	0.045 (0.016)	1.04 (0.04)	0.93 (0.01)	0.42 (0.00)
1,000	Probit	GHK(10)	0.278 (0.015)	0.074 (0.017)	0.95 (0.03)	0.89 (0.01)	0.29 (0.00)
	Probit	GHK(50)	0.280 (0.017)	0.051 (0.017)	1.02 (0.04)	0.92 (0.01)	0.62 (0.01)
	Probit	GHK(250)	0.284 (0.020)	0.046 (0.017)	1.03 (0.04)	0.93 (0.01)	2.02 (0.03)
	Emulator	Emulator	0.086 (0.006)	0.010 (0.004)	0.99 (0.05)	0.95 (0.01)	2.27 (0.02)
	Probit	Emulator	0.084 (0.005)	0.009 (0.004)	1.06 (0.04)	0.96 (0.01)	2.26 (0.02)
10,000	Probit	GHK(10)	0.095 (0.006)	0.032 (0.006)	0.88 (0.03)	0.89 (0.01)	0.89 (0.01)
	Probit	GHK(50)	0.086 (0.006)	0.013 (0.005)	1.02 (0.04)	0.95 (0.01)	2.27 (0.03)
	Probit	GHK(250)	0.084 (0.005)	0.009 (0.004)	1.06 (0.04)	0.96 (0.01)	4.95 (0.05)
	Emulator	Emulator	0.028 (0.004)	0.003 (0.001)	0.97 (0.09)	0.95 (0.01)	18.09 (0.15)
	Probit	Emulator	0.028 (0.002)	0.002 (0.001)	1.00 (0.06)	0.95 (0.01)	18.30 (0.17)
100,000	Probit	GHK(10)	0.056 (0.004)	0.024 (0.003)	0.49 (0.02)	0.60 (0.02)	3.75 (0.03)
	Probit	GHK(50)	0.030 (0.002)	0.004 (0.002)	0.93 (0.05)	0.92 (0.01)	9.45 (0.11)
	Probit	GHK(250)	0.028 (0.002)	0.002 (0.001)	1.00 (0.06)	0.95 (0.01)	42.96 (0.49)

Table 3: Comparison of Emulator and GHK estimation methods for multinomial probit models with $K = 5$. As in Table 2, the emulator performs competitively with GHK(50) and GHK(250) at a comparable or lower computational cost.

n	DGP	Method	RMSE	RMS Bias	SE Ratio	Coverage	Time (s)
	Emulator	Emulator	0.324 (0.014)	0.195 (0.011)	1.56 (0.14)	0.93 (0.01)	2.24 (0.02)
	Probit	Emulator	0.343 (0.016)	0.207 (0.012)	1.71 (0.24)	0.93 (0.01)	2.21 (0.02)
1,000	Probit	GHK(10)	0.388 (0.017)	0.217 (0.013)	0.35 (0.07)	0.22 (0.02)	5.86 (0.30)
	Probit	GHK(50)	0.363 (0.016)	0.215 (0.013)	0.70 (0.07)	0.54 (0.03)	14.16 (0.95)
	Probit	GHK(250)	0.351 (0.017)	0.214 (0.012)	1.43 (0.11)	0.91 (0.01)	14.87 (0.41)
	Emulator	Emulator	0.172 (0.010)	0.073 (0.007)	1.74 (0.19)	0.96 (0.01)	19.28 (0.26)
	Probit	Emulator	0.200 (0.011)	0.080 (0.010)	1.81 (0.20)	0.95 (0.01)	19.04 (0.19)
10,000	Probit	GHK(10)	0.236 (0.014)	0.123 (0.010)	1.57 (0.52)	0.79 (0.02)	8.95 (0.30)
	Probit	GHK(50)	0.203 (0.010)	0.090 (0.010)	1.29 (0.07)	0.91 (0.01)	33.54 (0.83)
	Probit	GHK(250)	0.198 (0.010)	0.079 (0.010)	1.41 (0.07)	0.95 (0.01)	50.22 (0.70)
	Emulator	Emulator	0.069 (0.004)	0.013 (0.003)	1.07 (0.05)	0.95 (0.01)	165.59 (1.15)
	Probit	Emulator	0.082 (0.005)	0.020 (0.004)	0.92 (0.03)	0.93 (0.01)	164.88 (1.49)
100,000	Probit	GHK(10)	0.139 (0.009)	0.073 (0.006)	0.48 (0.02)	0.60 (0.01)	31.48 (0.35)
	Probit	GHK(50)	0.085 (0.005)	0.027 (0.004)	0.89 (0.03)	0.90 (0.01)	93.23 (0.98)
	Probit	GHK(250)	0.081 (0.005)	0.020 (0.004)	0.97 (0.03)	0.93 (0.01)	404.22 (5.59)

Table 4: Comparison of Emulator and GHK estimation methods for multinomial probit models with $K = 10$. The emulator performs comparably to GHK(250) in terms of statistical performance but requires far less computation time. Compare to Tables 2 and 3.

substantially with the sample size (with or without penalization), reaching SE Ratios of 0.92–1.07 for the Emulator and GHK(250) methods at $n = 100{,}000$.

n	DGP	Method	RMSE	RMS Bias	SE Ratio	Coverage	Time (s)
1,000	Emulator	Emulator	0.089 (0.004)	0.022 (0.006)	1.18 (0.05)	0.95 (0.01)	0.11 (0.00)
	Probit	Emulator	0.095 (0.004)	0.007 (0.005)	1.10 (0.04)	0.96 (0.01)	0.10 (0.00)
	Probit	GHK(10)	0.096 (0.004)	0.011 (0.005)	1.08 (0.04)	0.96 (0.01)	0.08 (0.00)
	Probit	GHK(50)	0.094 (0.004)	0.007 (0.005)	1.11 (0.04)	0.96 (0.01)	0.15 (0.00)
	Probit	GHK(250)	0.095 (0.004)	0.007 (0.005)	1.10 (0.04)	0.96 (0.01)	0.42 (0.01)
10,000	Emulator	Emulator	0.038 (0.002)	0.003 (0.002)	0.99 (0.04)	0.93 (0.01)	0.39 (0.01)
	Probit	Emulator	0.038 (0.002)	0.007 (0.003)	1.00 (0.05)	0.96 (0.01)	0.38 (0.00)
	Probit	GHK(10)	0.040 (0.002)	0.013 (0.002)	0.99 (0.05)	0.95 (0.01)	0.33 (0.00)
	Probit	GHK(50)	0.038 (0.002)	0.008 (0.003)	1.00 (0.05)	0.96 (0.01)	0.48 (0.01)
	Probit	GHK(250)	0.038 (0.002)	0.007 (0.003)	1.00 (0.05)	0.95 (0.01)	0.96 (0.01)
100,000	Emulator	Emulator	0.011 (0.001)	0.001 (0.001)	1.08 (0.05)	0.96 (0.01)	1.97 (0.02)
	Probit	Emulator	0.011 (0.001)	0.001 (0.001)	1.04 (0.04)	0.96 (0.01)	1.91 (0.01)
	Probit	GHK(10)	0.016 (0.001)	0.009 (0.001)	0.88 (0.04)	0.82 (0.02)	0.98 (0.01)
	Probit	GHK(50)	0.012 (0.001)	0.002 (0.001)	1.04 (0.04)	0.95 (0.01)	2.07 (0.03)
	Probit	GHK(250)	0.011 (0.001)	0.001 (0.001)	1.04 (0.05)	0.96 (0.01)	7.52 (0.09)

Table 5: Comparison of Emulator and GHK estimation methods for factor-structured multinomial probit models with $K = 3$. Performance metrics are averaged over the factor parameters (coefficients, uniquenesses, and factor loadings). The Emulator's performance mirrors that of GHK(250) at a much lower computational cost.

Table 5 displays analogous results for the Factor specification with $K = 3$. Performance metrics are averaged over the nonredundant elements of $\boldsymbol{\beta}$, $\boldsymbol{\Psi}$, and $\boldsymbol{\gamma}$. The Emulator's performance mirrors that of GHK(250) but at a much lower computational cost.

Tables 6 and 7 display analogous results for 5 and 10 alternatives, respectively. In both cases, the Emulator performs comparably to GHK(50) and GHK(250) despite not being trained specifically for factor models. The Emulator is faster than GHK(50) for $n \in \{1{,}000, 10{,}000\}$. At $n = 100{,}000$, the Emulator's computation time falls between that of GHK(50) and GHK(250).

Across all scenarios, the Emulator method performs competitively with or significantly better than GHK. We expect that the relative performance of the emulator would increase with (a) additional training and (b) implementation on specialized deep-learning hardware, such as GPUs. The latter is likely to be especially advantageous for the emulator because its computations are trivially parallelizable and highly optimized on modern GPUs.

n	DGP	Method	RMSE	RMS Bias	SE Ratio	Coverage	Time (s)
	Emulator	Emulator	0.128 (0.004)	0.016 (0.005)	1.20 (0.04)	0.97 (0.01)	0.32 (0.01)
	Probit	Emulator	0.131 (0.003)	0.012 (0.004)	1.19 (0.03)	0.97 (0.01)	0.32 (0.00)
1,000	Probit	GHK(10)	0.134 (0.004)	0.017 (0.005)	1.12 (0.03)	0.95 (0.01)	0.23 (0.00)
	Probit	GHK(50)	0.131 (0.003)	0.012 (0.004)	1.19 (0.03)	0.98 (0.01)	0.49 (0.01)
	Probit	GHK(250)	0.132 (0.003)	0.012 (0.004)	1.19 (0.03)	0.97 (0.01)	1.61 (0.02)
	Emulator	Emulator	0.059 (0.002)	0.007 (0.003)	1.01 (0.04)	0.95 (0.01)	1.92 (0.02)
	Probit	Emulator	0.059 (0.003)	0.005 (0.002)	1.04 (0.04)	0.95 (0.01)	1.97 (0.02)
10,000	Probit	GHK(10)	0.062 (0.003)	0.011 (0.003)	0.93 (0.04)	0.93 (0.01)	0.77 (0.01)
	Probit	GHK(50)	0.059 (0.003)	0.004 (0.003)	1.02 (0.04)	0.95 (0.01)	2.02 (0.03)
	Probit	GHK(250)	0.059 (0.003)	0.007 (0.003)	1.04 (0.04)	0.95 (0.01)	4.67 (0.08)
	Emulator	Emulator	0.019 (0.001)	0.002 (0.001)	1.02 (0.03)	0.96 (0.01)	15.46 (0.18)
	Probit	Emulator	0.020 (0.001)	0.006 (0.001)	1.04 (0.03)	0.94 (0.01)	15.58 (0.18)
100,000	Probit	GHK(10)	0.029 (0.001)	0.017 (0.001)	0.75 (0.03)	0.75 (0.02)	3.21 (0.04)
	Probit	GHK(50)	0.019 (0.001)	0.004 (0.001)	1.02 (0.03)	0.95 (0.01)	8.47 (0.12)
	Probit	GHK(250)	0.019 (0.001)	0.002 (0.001)	1.04 (0.03)	0.96 (0.01)	37.30 (0.59)

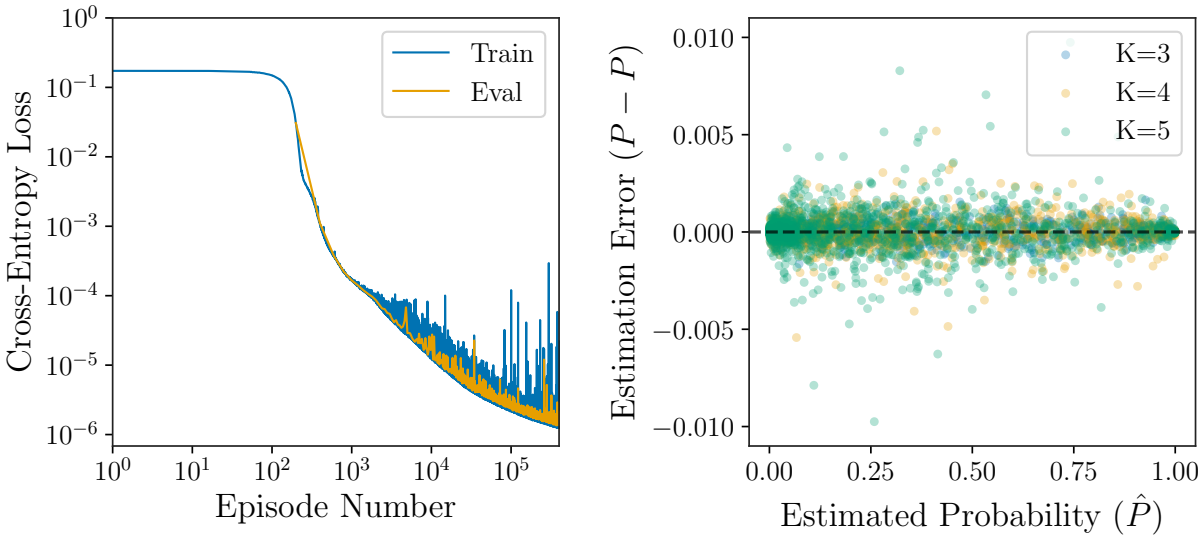
Table 6: Comparison of Emulator and GHK estimation methods for factor-structured multinomial probit models with $K = 5$. As in Table 5, the emulator performs competitively with GHK(50) and GHK(250) at a comparable or lower computational cost.

n	DGP	Method	RMSE	RMS Bias	SE Ratio	Coverage	Time (s)
	Emulator	Emulator	0.188 (0.004)	0.023 (0.004)	1.14 (0.02)	0.96 (0.00)	1.06 (0.01)
	Probit	Emulator	0.198 (0.005)	0.030 (0.005)	1.10 (0.02)	0.96 (0.01)	1.01 (0.01)
1,000	Probit	GHK(10)	0.220 (0.005)	0.059 (0.006)	0.92 (0.02)	0.90 (0.01)	0.77 (0.01)
	Probit	GHK(50)	0.205 (0.005)	0.039 (0.005)	1.06 (0.02)	0.95 (0.01)	2.44 (0.03)
	Probit	GHK(250)	0.199 (0.005)	0.030 (0.005)	1.11 (0.02)	0.96 (0.01)	5.93 (0.08)
	Emulator	Emulator	0.080 (0.002)	0.011 (0.003)	1.24 (0.04)	0.98 (0.00)	7.58 (0.06)
	Probit	Emulator	0.082 (0.002)	0.012 (0.003)	1.22 (0.04)	0.97 (0.01)	7.48 (0.06)
10,000	Probit	GHK(10)	0.103 (0.002)	0.049 (0.003)	0.95 (0.03)	0.88 (0.01)	4.03 (0.04)
	Probit	GHK(50)	0.087 (0.002)	0.019 (0.003)	1.14 (0.04)	0.96 (0.01)	14.83 (0.16)
	Probit	GHK(250)	0.084 (0.002)	0.012 (0.003)	1.20 (0.04)	0.97 (0.01)	24.85 (0.25)
	Emulator	Emulator	0.043 (0.002)	0.004 (0.002)	0.94 (0.04)	0.93 (0.01)	69.83 (0.57)
	Probit	Emulator	0.036 (0.001)	0.004 (0.001)	1.11 (0.04)	0.97 (0.01)	69.05 (0.66)
100,000	Probit	GHK(10)	0.071 (0.002)	0.047 (0.002)	0.65 (0.02)	0.60 (0.02)	19.73 (0.20)
	Probit	GHK(50)	0.039 (0.001)	0.012 (0.001)	1.04 (0.04)	0.93 (0.01)	55.94 (0.75)
	Probit	GHK(250)	0.036 (0.001)	0.004 (0.001)	1.12 (0.05)	0.96 (0.01)	218.97 (2.94)

Table 7: Comparison of Emulator and GHK estimation methods for factor-structured multinomial probit models with $K = 10$. The emulator performs comparably to GHK(250) in terms of statistical performance but requires far less computation time. Compare to Tables 5 and 6.

6.3 Multi-K Training Accuracy

We conclude this section by providing numerical results from training a neural network emulator to approximate choice probabilities for $K \in \{3, 4, 5\}$ simultaneously. We use the same neural network as we used in the previous section for $K = 5$. As above, we train for 400,000 episodes, but we loop over training examples with $K = 3$, 4, and 5 alternatives within each episode. The training process completes in 13.0 hours.



(a) Training Loss

(b) Estimation Accuracy

Figure 3: (a) Training and evaluation loss over 400,000 training episodes for emulators with $K \in \{3, 4, 5\}$ alternatives. The training loss (blue) decreases consistently throughout training, and the evaluation loss (orange) closely tracks the training loss, indicating minimal overfitting. (b) Emulator probability estimates compared to estimation errors. The emulator simultaneously learns the choice probabilities for $K = 3$, 4, 5 with small error.

Figure 3a shows the training and evaluation loss steadily decreasing throughout the training procedure, approaching levels similar to those of Figure 2. Figure 3b plots the estimated choice probabilities against estimation errors for $K \in \{3, 4, 5\}$ with 400 simulated examples each. The small errors indicate that the model accurately approximates the choice probabilities across all three values of K simultaneously.

7 Conclusion

This paper proposes an amortized inference approach for discrete choice models. The approach uses a novel neural network architecture to approximate choice probabilities under general error distributions, including those featuring nontrivial cross-alternative correlations,

such as MNP models. This generality enables flexible modeling of empirical choice behavior, including complex substitution patterns, without losing the theoretical grounding or interpretability of RUM models.

Because the emulator operates on the generic choice problem, it is agnostic to the assumed parametric form of the deterministic utilities or the accompanying scale (covariance) matrix. Given a pretrained emulator, this generality enables drop-in adoption without the need for custom architecture changes or additional training. In particular, generalizing from logit to probit requires only the replacement of closed-form softmax probabilities with emulator evaluations. Other correlated error distributions are similarly straightforward.

Our proposed architecture and training process feature a preprocessing transformation, DeepSet modules (Zaheer et al., 2017), equivariant layers, smooth activation functions, and Sobolev training (Czarnecki et al., 2017). Together, these design choices produce smooth approximations of choice probabilities that respect their invariance properties and accelerate learning via parameter sharing. We show that the architecture possesses a universal approximation property, extending recent theoretical results for symmetric matrices (Blum-Smith et al., 2025) to the joint space of utility vectors and centered scale (covariance) matrices.

In Monte Carlo simulations, we find that emulator-based estimators perform as well or better than GHK-based estimators in terms of both statistical performance and computational requirements when estimating MNP models. These findings support our theoretical results showing that emulator-based estimators inherit the properties of exact MLEs under mild approximation conditions. We focus on MNP models in the simulation study to enable comparison to the GHK simulator, which is directly applicable only to Gaussian errors. The computational benefits of the amortized inference approach are amplified by the fact that emulator evaluations are trivially parallelizable and highly optimized on modern deep learning hardware, such as GPUs.

Several limitations merit discussion. Practitioners must ensure sufficient overlap between the distribution of inputs in their application and the training distribution. Additionally, our asymptotic theory requires the emulator approximation error to vanish at an appropriate rate, meaning applications with very large samples may require more extensively trained emulators. Lastly, models with correlated errors require identifying restrictions not needed for independent logit-based models. Although our approach enables simple and efficient computation of choice probabilities for these models, the data requirements and modeling decisions to ensure proper identification remain a separate challenge.

The framework also allows numerous extensions. Our emulator architecture and training procedure could be specialized to specific settings, such as factor-structured scale matrices or mixture distributions. We expect these could be handled by reconfiguring the modules in our architecture and, in some cases, generalizing the preprocessing transformation. In particular, the two preceding examples would likely require an additional DeepSet operation over factors or mixture components. Our emulators could also be used in tandem with other modern choice modeling architectures, such as those described in Section 2. Previous work has applied neural-network-assisted estimation approaches to other econometric models with intractable likelihood functions, such as dynamic discrete choice (Norets, 2012) and consumer search (Wei and Jiang, 2025), in the context of specific parametric models. The amortized inference approach presented here could likely be generalized to many of these settings, enabling efficient amortized inference without modifying or retraining the neural network.

Acknowledgments

The authors acknowledge the use of AI tools to assist in the development of this manuscript. All content has been manually reviewed for accuracy.

References

Aouad, A. and Desir, A. (2025). Representing random utility choice models with nueral networks. *Management Science*, Articles in Advance.

Bagheri, N., Ghasri, M., and Barlow, M. (2025). A neural estimation framework for discrete choice models with arbitrary error distributions. *Journal of Choice Modelling*, 57:100583.

Bentz, Y. and Merunka, D. (2000). Neural networks and the multinomial logit for brand choice modelling: a hybrid approach. *Journal of Forecasting*, 19(3):177–200.

Berbeglia, G. and Venkataraman, A. (2025). The generalized stochastic preference choice model. *arXiv preprint arXiv:1803.04244*.

Bloch, H. and Marschak, J. (1960). Stochastic rationality and revealed stochastic preference. In *Contributions to probability and Statistics*. Stanford University Press.

Blum-Smith, B., Huang, N. T., Cuturi, M., and Villar, S. (2025). Functions on symmetric matrices and point clouds via lightweight invariant features from Galois theory. *SIAM Journal on Applied Algebra and Geometry*, 9(4):902–938.

Bredon, G. E. (1972). *Introduction to Compact Transformation Groups*, volume 46. Academic Press, New York.

Cai, Z., Wang, H., Talluri, K., and Li, X. (2022). Deep learning for choice modeling. *arXiv preprint arXiv:2208.09325*.

Chen, N., Gallego, G., and Tang, Z. (2025). The use of binary choice forests to model and estimate discrete choices. *arXiv preprint arXiv:1908.01109*.

Chen, Y.-C. and Mišić, V. V. (2022). Decision forest: A nonparametric approach to modeling irrational choice. *Management Science (INFORMS)*, 68(10):7090–7111.

Cocola, J. and Hand, P. (2020). Global convergence of Sobolev training for overparameterized neural networks. In *International Conference on Machine Learning, Optimization, and Data Science*, pages 574–586. Springer.

Cranmer, K., Brehmer, J., and Louppe, G. (2020). The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314.

Czarnecki, W. M., Osindero, S., Jaderberg, M., Swirszcz, G., and Pascanu, R. (2017). Sobolev training for neural networks. In *Advances in Neural Information Processing Systems*, volume 30, pages 4278–4287.

De Ryck, T., Lanthaler, S., and Mishra, S. (2021). On the approximation of functions by tanh neural networks. *Neural Networks*, 143:732–750.

Dugas, C., Bengio, Y., Bélisle, F., Nadeau, C., and Garcia, R. (2000). Incorporating second-order functional knowledge for better option pricing. In Leen, T., Dietterich, T., and Tresp, V., editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press.

E, W. and Wang, Q. (2018). Exponential convergence of the deep neural network approximation for analytic functions. *Science China Mathematics*, 61(10):1733–1740.

Farrell, M. H., Liang, T., and Misra, S. (2021). Deep neural networks for estimation and inference. *Econometrica*, 89(1):181–213.

Geweke, J. (1989). Bayesian inference in econometric models using Monte Carlo integration. *Econometrica*, 57(6):1317–1339.

Hajivassiliou, V. A. and McFadden, D. L. (1998). The method of simulated scores for the estimation of LDV models. *Econometrica*, 66(4):863–896.

Han, Y., Pereira, F. C., Ben-Akiva, M., and Zegras, C. (2022). A neural-embedded choice model: Learning taste representation with strengthened. *Transportation Research B*, 163:166–186.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

Hruschka, H. (2007). Using a heterogeneous multinomial probit model with a neural net extension to model brand choice. *Journal of Forecasting*, 26(2):113–127.

Jang, E., Gu, S., and Poole, B. (2017). Categorical reparameterization with Gumbel–softmax. In *International Conference on Learning Representations*.

Jiang, Z., Li, J., and Zhang, D. (2025). A high-dimensional choice model for online retailing. *Management Science (INFORMS)*, 71(4):3320–3339.

Keane, M. P. (1992). A note on identification in the multinomial probit model. *Journal of Business & Economic Statistics*, 10(2):193–200.

Keane, M. P. (1994). A computationally practical simulation estimator for panel data. *Econometrica*, 62(1):95–116.

Kingma, D. P. and Ba, J. L. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Lhéritier, A., Bocamazo, M., Delahaye, T., and Acuna-Agost, R. (2019). Airline itinerary choice modeling using machine learning. *Journal of Choice Modelling*, 31:198–209.

Liu, D. C. and Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528.

Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Lueckmann, J.-M., Bassetto, G., Karaletsos, T., and Macke, J. H. (2019). Likelihood-free inference with emulator networks. In Ruiz, F., Zhang, C., Liang, D., and Bui, T., editors, *Proceedings of The 1st Symposium on Advances in Approximate Bayesian Inference*, volume 96 of *Proceedings of Machine Learning Research*, pages 32–53. PMLR.

Maddison, C. J., Mnih, A., and Teh, Y. W. (2017). The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*.

McCulloch, R. and Rossi, P. E. (1994). An exact likelihood analysis of the multinomial probit model. *Journal of Econometrics*, 64(1):207–240.

McFadden, D. (1974). Conditional logit analysis of qualitative choice behavior. In Zarembka, P., editor, *Frontiers in Econometrics*, pages 105–142. Academic Press, New York.

McFadden, D. and Richter, M. (1990). Stochastic rationality and revealed stochastic preference. In *Preferences, Uncertainty, and Optimality: Essays in Honor of Leo Hurwicz*. Westview Press.

McFadden, D. and Train, K. (2000). Mixed mnl models for discrete response. *Journal of Applied Econometrics*, 15(5):447–470.

Mityagin, B. S. (2020). The zero set of a real analytic function. *Mathematical Notes*, 107(3):529–530.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Newey, W. K. and McFadden, D. (1994). Large sample estimation and hypothesis testing. In Engle, R. F. and McFadden, D. L., editors, *Handbook of Econometrics*, volume 4, pages 2111–2245. Elsevier.

Norets, A. (2012). Estimation of dynamic discrete choice models using artificial neural network approximations. *Econometric Reviews*, 31(1):84–106.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32.

Pfannschmidt, K., Gupta, P., Haddenhorst, B., and Hüllermeier, E. (2022). Learning context-dependent choice functions. *International Journal of Approximate Reasoning*, 140:116–155.

Ramachandran, P., Zoph, B., and Le, Q. V. (2018). Searching for activation functions. In *International Conference on Learning Representations*.

Rosenfeld, N., Oshiba, K., and Singer, Y. (2020). Predicting choice with set-dependent aggregation. In III, H. D. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8220–8229. PMLR.

Schmidt-Hieber, J. (2020). Nonparametric regression using deep neural networks with ReLU activation function. *The Annals of Statistics*, 48(4):1875–1897.

Segol, N. and Lipman, Y. (2020). On universal equivariant set networks. In *International Conference on Learning Representations*.

Shen, X., Jiang, C., Sakhanenko, L., and Lu, Q. (2023). Asymptotic properties of neural network sieve estimators. *Journal of Nonparametric Statistics*, 35(4):839–868. PMID: 38169985.

Sifringer, B., Lurkin, V., and Alahi, A. (2020). Enhancing discrete choice models with representation learning. *Transportation Research Part B: Methodological*, 140:236–261.

Singh, A., Liu, Y., and Yoganarasimhan, H. (2023). Choice models and permutation invariance: Demand estimation in differentiated products markets. *arXiv preprint arXiv:2307.07090*.

Train, K. E. (2009). *Discrete Choice Methods with Simulation*. Cambridge University Press, Cambridge, 2nd edition.

van der Vaart, A. W. (1998). *Asymptotic Statistics*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, Cambridge, UK.

Wang, F. and Ross, C. L. (2018). Machine learning travel mode choices: Comparing the performance of an extreme gradient boosting model with a multinomial logit model. *Transportation Research Record*, 2672(47):35–45.

Wang, S., Mo, B., and Zhao, J. (2020). Deep neural networks for choice analysis: Architecture design with alternative-specific utility functions. *Transportation Research Part C: Emerging Technologies*, 112:234–251.

Wang, S., Mo, B., and Zhao, J. (2021). Theory-based residual neural networks: A synergy of discrete choice models and deep neural networks. *Transportation Research Part B: Methodological*, 146:333–358.

Wei, Y. M. and Jiang, Z. (2025). Estimating parameters of structural models using neural networks. *Marketing Science*, 44(1):102–128.

White, H. (1982). Maximum likelihood estimation of misspecified models. *Econometrica*, 50(1):1–25.

Wong, M. and Farooq, B. (2021). Reslogit: A residual neural network logit model for data-driven choice modelling. *Transportation Research Part C: Emerging Technologies*, 126:103050.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. (2017). Deep sets. In *Advances in Neural Information Processing Systems*, volume 30.

Supplementary Materials for "Amortized Inference for Correlated Discrete Choice Models via Equivariant Neural Networks"

Easton Huch
Postdoctoral Fellow, Johns Hopkins Carey Business School

Michael Keane
Carey Distinguished Professor, Johns Hopkins Carey Business School

Appendix A presents results and proofs regarding generic separation and universal approximation. Section A.8 proves the results given in Section 5.1. Appendix B provides proofs of the estimation theory results in Section 5.2. Appendix C provides details on the simulation study.

A Generic Separation and Universal Approximation

This appendix develops the group-theoretic foundations underlying the per-alternative encoder architecture presented in the main text. We establish that certain invariants generically separate orbits under permutation groups, yielding universal approximation results via the Stone–Weierstrass theorem.

Let $\mathbb{S}_K$ denote the space of symmetric real-valued $K \times K$ matrices. We consider an extended group G_K acting on $\mathbb{S}_K$ by permutation, scaling, and translation by matrices in the null space of double-centering. Section A.1 introduces the necessary notation and establishes basic properties of the centering matrix and the normalized double-centering function f.

Sections A.2–A.4 establish that the composition $h = g \circ f$, which first applies normalized double-centering and then extracts multisets of diagonal and off-diagonal elements, generically separates G_K-orbits. The exceptional set where separation fails is closed, G_K-invariant, and has measure zero.

Section A.5 considers the restriction to the centered, trace-normalized subspace $\mathbb{S}_K^* = \mathrm{Im}(f)$, where g generically separates S_K-orbits and the Stone–Weierstrass theorem yields universal

approximation.

Section A.6 develops a variant for a distinguished index j, showing that an invariant g_j generically separates orbits under $S_{K-1}^{(j)}$, the subgroup fixing j.

Sections A.7–A.8 extend the theory to incorporate deterministic utilities, culminating in proofs of Theorem 1 and Theorem 2 from the main text. Throughout, $\mathbf{X}$ and $\mathbf{Y}$ represent scale matrices. As in Section 4.3, we sometimes refer to these matrices as covariance matrices although this interpretation is not strictly correct outside of MNP models.

A.1 Preliminaries

Throughout, let $K \geq 2$ be the number of alternatives. Let $\mathbf{I}_K$ denote the $K \times K$ identity matrix and $\mathbf{1}_K \in \mathbb{R}^K$ the vector of all ones.

Definition 1 (Centering Matrix)**.** The *centering matrix* $\mathbf{M} \in \mathbb{R}^{K \times K}$ is defined as $\mathbf{M} = \mathbf{I}_K - \frac{1}{K}\mathbf{1}_K\mathbf{1}_K^\top$.

Proposition 2 (Properties of the Centering Matrix)**.** *The centering matrix* $\mathbf{M}$ *is symmetric* ($\mathbf{M}^\top = \mathbf{M}$)*, idempotent* ($\mathbf{M}^2 = \mathbf{M}$)*, and annihilates constant vectors* ($\mathbf{M}\mathbf{1}_K = \mathbf{0}$)*.*

Proof. Symmetry is immediate. For idempotency, $\mathbf{M}^2 = \mathbf{I}_K - \frac{2}{K}\mathbf{1}_K\mathbf{1}_K^\top + \frac{1}{K^2}\mathbf{1}_K(\mathbf{1}_K^\top\mathbf{1}_K)\mathbf{1}_K^\top = \mathbf{I}_K - \frac{2}{K}\mathbf{1}_K\mathbf{1}_K^\top + \frac{1}{K}\mathbf{1}_K\mathbf{1}_K^\top = \mathbf{M}$. Finally, $\mathbf{M}\mathbf{1}_K = \mathbf{1}_K - \frac{1}{K}\mathbf{1}_K(\mathbf{1}_K^\top\mathbf{1}_K) = \mathbf{1}_K - \mathbf{1}_K = \mathbf{0}$. □

Definition 2 (Double-Centering Function)**.** Define the domain $\mathcal{U} = \{\mathbf{X} \in \mathbb{S}_K : \operatorname{tr}(\mathbf{MXM}) \neq 0\}$ and the function $f : \mathcal{U} \to \mathbb{S}_K$ by

$$f(\mathbf{X}) = \frac{K \cdot \mathbf{MXM}}{\operatorname{tr}(\mathbf{MXM})}.$$

Proposition 3 (Explicit Formula)**.** *For* $\mathbf{X} \in \mathbb{S}_K$*, let* $\bar{X}_{i\bullet} = \frac{1}{K}\sum_{k=1}^K X_{ik}$*,* $\bar{X}_{\bullet j} = \frac{1}{K}\sum_{k=1}^K X_{kj}$*, and* $\bar{X}_{\bullet\bullet} = \frac{1}{K^2}\sum_{i,j} X_{ij}$*. Then:*

(i) $[\mathbf{MXM}]_{ij} = X_{ij} - \bar{X}_{i\bullet} - \bar{X}_{\bullet j} + \bar{X}_{\bullet\bullet}$.

(ii) $\operatorname{tr}(\mathbf{MXM}) = \operatorname{tr}(\mathbf{X}) - K\bar{X}_{\bullet\bullet}$.

Proof. Part (i) follows by direct expansion. For part (ii), use idempotency and the cyclic property of trace: $\operatorname{tr}(\mathbf{MXM}) = \operatorname{tr}(\mathbf{M}^2\mathbf{X}) = \operatorname{tr}(\mathbf{MX}) = \operatorname{tr}(\mathbf{X}) - \frac{1}{K}\mathbf{1}_K^\top\mathbf{X}\mathbf{1}_K = \operatorname{tr}(\mathbf{X}) - K\bar{X}_{\bullet\bullet}$. □

Proposition 4 (Domain Properties)**.** *The set* $\mathbb{S}_K \setminus \mathcal{U} = \{\mathbf{X} \in \mathbb{S}_K : \operatorname{tr}(\mathbf{MXM}) = 0\}$ *is a hyperplane in* $\mathbb{S}_K$ *with Lebesgue measure zero. The domain* $\mathcal{U}$ *is open and* G_K*-invariant.*

Proof. By Proposition 3(ii), $\operatorname{tr}(\mathbf{MXM})$ is a nontrivial linear function, so its zero set is a measure-zero hyperplane. Openness follows from continuity. For G_K-invariance: scaling by $c \neq 0$ gives $\operatorname{tr}(\mathbf{M}(c\mathbf{X})\mathbf{M}) = c\operatorname{tr}(\mathbf{MXM}) \neq 0$; null-space translation by $\mathbf{N} = \boldsymbol{u}\mathbf{1}_K^\top + \mathbf{1}_K\boldsymbol{u}^\top$ satisfies $\mathbf{MNM} = \mathbf{0}$ since $\mathbf{M}\mathbf{1}_K = \mathbf{0}$; permutation preserves the trace by the cyclic property. □

Definition 3 (Extended Group Action)**.** Let S_K denote the symmetric group on K elements with associated permutation matrices $\mathbf{P}_\pi$, and let $\mathbb{R}^* = \mathbb{R} \setminus \{0\}$. Define the extended group $G_K = \mathbb{R}^K \rtimes (\mathbb{R}^* \times S_K)$, which acts on $\mathbb{S}_K$ by:

$$(c, \boldsymbol{u}, \pi) \cdot \mathbf{X} = c\,\mathbf{P}_\pi\mathbf{X}\mathbf{P}_\pi^\top + \boldsymbol{u}\mathbf{1}_K^\top + \mathbf{1}_K\boldsymbol{u}^\top.$$

Proposition 5 (f and G_K)**.** *The function* f *satisfies:*

(i) Equivariance under permutation: $f(\mathbf{P}_\pi\mathbf{X}\mathbf{P}_\pi^\top) = \mathbf{P}_\pi f(\mathbf{X})\mathbf{P}_\pi^\top$.

(ii) Invariance under scaling: $f(c\mathbf{X}) = f(\mathbf{X})$ *for* $c \neq 0$.

(iii) Invariance under null-space translation: $f(\mathbf{X} + \boldsymbol{u}\mathbf{1}_K^\top + \mathbf{1}_K\boldsymbol{u}^\top) = f(\mathbf{X})$.

Proof. For (i), since $\mathbf{P}_\pi\mathbf{1}_K = \mathbf{1}_K$, we have $\mathbf{P}_\pi\mathbf{M} = \mathbf{M}\mathbf{P}_\pi$, so $\mathbf{M}(\mathbf{P}_\pi\mathbf{X}\mathbf{P}_\pi^\top)\mathbf{M} = \mathbf{P}_\pi\mathbf{MXM}\mathbf{P}_\pi^\top$, and the trace is preserved. Part (ii) follows from $f(c\mathbf{X}) = \frac{cK\,\mathbf{MXM}}{c\cdot\operatorname{tr}(\mathbf{MXM})} = f(\mathbf{X})$. For (iii), $\mathbf{M}(\boldsymbol{u}\mathbf{1}_K^\top + \mathbf{1}_K\boldsymbol{u}^\top)\mathbf{M} = \mathbf{0}$ since $\mathbf{M}\mathbf{1}_K = \mathbf{0}$. □

Lemma 1 (Preimage Characterization)**.** *If* $f(\mathbf{X}) = f(\mathbf{Y})$ *for* $\mathbf{X}, \mathbf{Y} \in \mathcal{U}$*, then there exist* $c \in \mathbb{R}^*$ *and* $\boldsymbol{w} \in \mathbb{R}^K$ *such that* $\mathbf{X} = c\mathbf{Y} + \boldsymbol{w}\mathbf{1}_K^\top + \mathbf{1}_K\boldsymbol{w}^\top$.

Proof. Let $t_X = \operatorname{tr}(\mathbf{MXM})$ and $t_Y = \operatorname{tr}(\mathbf{MYM})$. The assumption $f(\mathbf{X}) = f(\mathbf{Y})$ implies $\mathbf{MXM} = c\,\mathbf{MYM}$ where $c = t_X/t_Y \in \mathbb{R}^*$.

For symmetric $\mathbf{Z}$, let $\boldsymbol{\mu}_\mathbf{Z} = \mathbf{Z}\mathbf{1}_K/K$ (row averages) and $m_\mathbf{Z} = \mathbf{1}_K^\top\mathbf{Z}\mathbf{1}_K/K^2$ (grand mean). Expanding $\mathbf{MZM} = \mathbf{Z} - \boldsymbol{\mu}_\mathbf{Z}\mathbf{1}_K^\top - \mathbf{1}_K\boldsymbol{\mu}_\mathbf{Z}^\top + m_\mathbf{Z}\mathbf{1}_K\mathbf{1}_K^\top$ and applying to both sides:

$$\mathbf{X} = c\mathbf{Y} + (\boldsymbol{\mu}_\mathbf{X} - c\boldsymbol{\mu}_\mathbf{Y})\mathbf{1}_K^\top + \mathbf{1}_K(\boldsymbol{\mu}_\mathbf{X} - c\boldsymbol{\mu}_\mathbf{Y})^\top - (m_\mathbf{X} - cm_\mathbf{Y})\mathbf{1}_K\mathbf{1}_K^\top.$$

Setting $\boldsymbol{w} = \boldsymbol{\mu}_\mathbf{X} - c\boldsymbol{\mu}_\mathbf{Y} - \frac{m_\mathbf{X} - cm_\mathbf{Y}}{2}\mathbf{1}_K$ gives the result. □

Definition 4 (Centered, Trace-Normalized Subspace)**.** Define $\mathbb{S}_K^* = \{\mathbf{Y} \in \mathbb{S}_K : \mathbf{Y}\mathbf{1}_K = \mathbf{0}, \operatorname{tr}(\mathbf{Y}) = K\}$.

Definition 5 (Invariant Function g)**.** For $\mathbf{Y} \in \mathbb{S}_K^*$, define $g(\mathbf{Y}) = (\mathcal{D}_K(\mathbf{Y}), \mathcal{O}_K(\mathbf{Y}))$ where $\mathcal{D}_K(\mathbf{Y}) = \{\{Y_{ii} : i = 1, \ldots, K\}\}$ is the multiset of diagonal entries and $\mathcal{O}_K(\mathbf{Y}) = \{\{Y_{ij} : 1 \leq i < j \leq K\}\}$ is the multiset of off-diagonal entries.

A.2 Generic Injectivity

We now define the genericity conditions under which g separates orbits.

Definition 6 (Genericity Conditions)**.** Let $\mathbf{Y} \in \mathbb{S}_K$. We say $\mathbf{Y}$ satisfies:

(a) *Condition (a)* if all entries are distinct: $Y_{ij} \neq Y_{kl}$ for $(i, j) \neq (k, l)$ with $i \leq j$, $k \leq l$.

(b) *Condition (b)* if for each i, there exists exactly one $(K-1)$-subset $\mathcal{S}$ of off-diagonal positions such that $\sum_{(k,l)\in\mathcal{S}} Y_{kl} = -Y_{ii}$.

Condition (b) exploits the row-sum constraint $\mathbf{Y}\mathbf{1}_K = \mathbf{0}$: the off-diagonal entries in row i sum to $-Y_{ii}$, so the canonical subset $\mathcal{S}_i^* = \{(i, j) : j \neq i\}$ always satisfies the equation. Condition (b) requires this subset to be unique among the $\binom{K(K-1)/2}{K-1}$ possible $(K-1)$-subsets.

Lemma 2 (Recovery Lemma)**.** *If $\mathbf{Y} \in \mathbb{S}_K^*$ satisfies conditions (a) and (b), then $g(\mathbf{Y})$ determines $\mathbf{Y}$ up to S_K-orbit.*

Proof. Given multisets $\mathcal{D}_K$ and $\mathcal{O}_K$, we reconstruct $\mathbf{Y}$ up to relabeling:

Step 1: Order the distinct diagonal elements as $Y_{(1)(1)} > \cdots > Y_{(K)(K)}$, assigning canonical labels to alternatives.

Step 2: For each (i), identify $\mathcal{R}_{(i)} \subset \mathcal{O}_K$ as the unique $(K-1)$-subset summing to $-Y_{(i)(i)}$ (exists by condition (b)).

Step 3: For $i \neq j$, recover $Y_{(i)(j)}$ as the unique element in $\mathcal{R}_{(i)} \cap \mathcal{R}_{(j)}$ (singleton by condition (a)).

Any $\mathbf{Y}'$ with $g(\mathbf{Y}') = g(\mathbf{Y})$ satisfying (a) and (b) yields the same reconstruction up to relabeling. □

A.3 The Exceptional Set

Definition 7 (Exceptional Set)**.** Define $\mathcal{E} = \mathcal{U}^C \cup \{\mathbf{X} \in \mathcal{U} : f(\mathbf{X})$ violates (a) or (b)$\}$.

Lemma 3 (Non-Triviality of Constraints)**.** *The following hold:*

(i) For distinct positions $(i,j) \neq (k,l)$, the function $f(\mathbf{X})_{ij} - f(\mathbf{X})_{kl}$ is not identically zero on $\mathcal{U}$.

(ii) For each index i and $(K-1)$-subset $\mathcal{S} \neq \mathcal{S}_i^*$, the function $\sum_{(k,l)\in\mathcal{S}} f(\mathbf{X})_{kl} + f(\mathbf{X})_{ii}$ is not identically zero on $\mathcal{U}$.

Proof. Consider diagonal $\mathbf{X} = \mathrm{diag}(\lambda_1, \ldots, \lambda_K)$ with distinct λ_j and $\sum_j \lambda_j \neq 0$. Then $[\mathbf{MXM}]_{ii} = \lambda_i(1 - 2/K) + \bar{\lambda}/K$ and $[\mathbf{MXM}]_{ij} = (\bar{\lambda} - \lambda_i - \lambda_j)/K$ for $i \neq j$. For generic λ_j, all entries of $f(\mathbf{X})$ are distinct, proving (i). For (ii), the linear combination for $\mathcal{S} \neq \mathcal{S}_i^*$ has nonzero coefficient on λ_i, so it is nontrivial. □

Proposition 6 (Properties of $\mathcal{E}$)**.** *The set $\mathcal{E}$ is closed, G_K-invariant, and has Lebesgue measure zero.*

Proof. **Closed:** Conditions (a) and (b) are open (strict inequalities and unique subset-sums are preserved under small perturbations), so $\mathcal{E}^C$ is open.

G_K-invariant: By Proposition 5, scaling and null-space translation leave $f(\mathbf{X})$ unchanged. Permutation conjugates $f(\mathbf{X})$, preserving distinctness and subset-sum structure.

Measure zero: Each entry $f(\mathbf{X})_{ij}$ is a rational (hence analytic) function on $\mathcal{U}$ by Proposition 3. By Lemma 3, the constraints defining violations of (a) and (b) are nontrivial analytic functions. Their zero sets have measure zero by Mityagin (2020), and $\mathcal{E}$ is a finite union of such sets. □

A.4 Basic Result for the Full Space

Theorem 7 (Main Theorem)**.** *For any $\mathbf{X}, \mathbf{Y} \in \mathbb{S}_K \setminus \mathcal{E}$: $g\{f(\mathbf{X})\} = g\{f(\mathbf{Y})\}$ implies $\mathbf{X}$ and $\mathbf{Y}$ are in the same G_K-orbit.*

Proof. By Lemma 2, $g\{f(\mathbf{X})\} = g\{f(\mathbf{Y})\}$ implies $f(\mathbf{Y}) = \mathbf{P}_\pi f(\mathbf{X})\mathbf{P}_\pi^\top$ for some $\pi \in S_K$. By Proposition 5(i), $f(\mathbf{P}_\pi \mathbf{X} \mathbf{P}_\pi^\top) = f(\mathbf{Y})$. By Lemma 1, $\mathbf{Y} = c\,\mathbf{P}_\pi \mathbf{X} \mathbf{P}_\pi^\top + \boldsymbol{w}\mathbf{1}_K^\top + \mathbf{1}_K \boldsymbol{w}^\top$ for some $c \in \mathbb{R}^*$, $\boldsymbol{w} \in \mathbb{R}^K$. □

A.5 Restriction to Centered, Trace-Normalized Matrices

Proposition 7 (Properties of $\mathbb{S}_K^*$)**.** *The subspace $\mathbb{S}_K^*$ is a nonempty, closed affine subspace of dimension $(K+1)(K-2)/2$, satisfies $\mathbb{S}_K^* = \mathrm{Im}(f)$, and is S_K-invariant.*

Proof. The dimension follows from $\dim(\mathbb{S}_K) - (K+1) = K(K+1)/2 - K - 1 = (K+1)(K-2)/2$. For $\mathbb{S}_K^* = \mathrm{Im}(f)$: if $\mathbf{X} \in \mathcal{U}$, then $f(\mathbf{X})\mathbf{1}_K = \mathbf{0}$ and $\mathrm{tr}(f(\mathbf{X})) = K$, so $\mathrm{Im}(f) \subseteq \mathbb{S}_K^*$; conversely, if $\mathbf{Y} \in \mathbb{S}_K^*$, then $\mathbf{MYM} = \mathbf{Y}$ and $f(\mathbf{Y}) = \mathbf{Y}$. For S_K-invariance: $(\mathbf{P}_\pi \mathbf{Y} \mathbf{P}_\pi^\top)\mathbf{1}_K = \mathbf{P}_\pi \mathbf{Y} \mathbf{1}_K = \mathbf{0}$ and the trace is preserved. □

Definition 8 (Restricted Exceptional Set)**.** Define $\mathcal{E}^* = \{\mathbf{Y} \in \mathbb{S}_K^* : \mathbf{Y} \text{ violates (a) or (b)}\}$.

Theorem 8 (Generic Separation on $\mathbb{S}_K^*$)**.** *The set $\mathcal{E}^*$ is closed, S_K-invariant, and has measure zero in $\mathbb{S}_K^*$. For any $\mathbf{Y}_1, \mathbf{Y}_2 \in \mathbb{S}_K^* \setminus \mathcal{E}^*$: $g(\mathbf{Y}_1) = g(\mathbf{Y}_2)$ implies $\mathbf{Y}_1$ and $\mathbf{Y}_2$ are in the same S_K-orbit.*

Proof. The properties of $\mathcal{E}^*$ follow from Proposition 6 restricted to $\mathbb{S}_K^*$, using Lemma 3 (which provides nontrivial constraints via $\mathbb{S}_K^* = \mathrm{Im}(f)$). Separation follows from Lemma 2. □

Corollary 2 (Universal Approximation on $\mathbb{S}_K^*$)**.** *Any continuous, S_K-invariant function on $\mathbb{S}_K^* \setminus \mathcal{E}^*$ can be uniformly approximated on compact subsets by an MLP taking the sorted diagonal and off-diagonal entries as input.*

Proof. Following Blum-Smith et al. (2025), encode g via continuous functions: $g_k^d(\mathbf{Y}) =$ the kth largest diagonal entry, and $g_\ell^o(\mathbf{Y}) =$ the ℓth largest off-diagonal entry. These are S_K-invariant and separate points in $(\mathbb{S}_K^* \setminus \mathcal{E}^*)/S_K$ by Theorem 8. Since S_K is finite (hence compact), the quotient is Hausdorff (Bredon, 1972). By Stone–Weierstrass, the algebra generated by these functions is dense. By the universal approximation theorem (Cybenko, 1989; Hornik et al., 1989), MLPs can approximate any continuous function of these inputs. □

A.6 Restriction to Submatrices

For a distinguished index j, we develop an invariant separating $S_{K-1}^{(j)}$-orbits, where $S_{K-1}^{(j)} = \{\pi \in S_K : \pi(j) = j\}$.

Definition 9 (Submatrix Invariant g_j)**.** For $\mathbf{Y} \in \mathbb{S}_K^*$ and index j, define $g_j(\mathbf{Y}) = (\mathcal{P}_j(\mathbf{Y}), \mathcal{O}_j(\mathbf{Y}))$

where:

$$\mathcal{P}_j(\mathbf{Y}) = \{\{(Y_{kk}, Y_{jk}) : k \neq j\}\},$$
$$\mathcal{O}_j(\mathbf{Y}) = \{\{Y_{kl} : 1 \leq k < l \leq K,\, k \neq j,\, l \neq j\}\}.$$

The multiset $\mathcal{P}_j$ couples each diagonal entry Y_{kk} (for $k \neq j$) with the covariance Y_{jk}. The multiset $\mathcal{O}_j$ contains off-diagonal entries of $\mathbf{Y}_{-j,-j}$. Note that $Y_{jj} = K - \sum_{k \neq j} Y_{kk}$ is implicitly recoverable.

Definition 10 (Genericity Conditions for Submatrices)**.** We say $\mathbf{Y} \in \mathbb{S}_K^*$ satisfies (a_j) if all entries except possibly Y_{jj} are distinct, and (b_j) if for each $k \neq j$, there is a unique $(K-2)$-subset $\mathcal{S}$ of off-diagonal positions in $\mathbf{Y}_{-j,-j}$ with $\sum_{(k,m)\in\mathcal{S}} Y_{km} = -Y_{kk} - Y_{jk}$.

The target sum $-Y_{kk} - Y_{jk}$ arises from the row-sum constraint: $Y_{kk} + Y_{kj} + \sum_{l \neq k, l \neq j} Y_{kl} = 0$.

Theorem 9 (Generic Separation via g_j)**.** *Define $\mathcal{E}_j = \{\mathbf{Y} \in \mathbb{S}_K^* : \mathbf{Y}$ violates (a_j) or $(b_j)\}$. Then $\mathcal{E}_j$ is closed, $S_{K-1}^{(j)}$-invariant, and has measure zero. For $\mathbf{Y}_1, \mathbf{Y}_2 \in \mathbb{S}_K^* \setminus \mathcal{E}_j$: $g_j(\mathbf{Y}_1) = g_j(\mathbf{Y}_2)$ implies $\mathbf{Y}_1, \mathbf{Y}_2$ are in the same $S_{K-1}^{(j)}$-orbit.*

Proof. The proof follows Theorem 8. For separation, the recovery procedure parallels Lemma 2: order pairs in $\mathcal{P}_i$ lexicographically, identify each row's off-diagonal entries via the unique subset-sum, and recover shared entries as singleton intersections. □

Corollary 3 (Universal Approximation via g_j)**.** *Any continuous, $S_{K-1}^{(j)}$-invariant function on $\mathbb{S}_K^* \setminus \mathcal{E}_j$ can be uniformly approximated on compact subsets by an MLP taking the sorted pairs from $\mathcal{P}_j$ and sorted entries from $\mathcal{O}_j$ as input.*

Proof. The proof is analogous to Corollary 2, with pairs replacing diagonal entries. □

A.7 Extension to Utility-Covariance Pairs: Global Invariants

We extend the theory to pairs $(\boldsymbol{v}, \mathbf{Y})$ consisting of a utility vector and covariance matrix.

Definition 11 (Centered Utility-Covariance Space)**.** Define $\mathcal{X}_K = \{(\boldsymbol{v}, \mathbf{Y}) \in \mathbb{R}^K \times \mathbb{S}_K : \sum_{i=1}^K v_i = 0,\, \mathbf{Y}\mathbf{1}_K = \mathbf{0},\, \mathrm{tr}(\mathbf{Y}) = K\}$.

This space has dimension $(K-1) + (K+1)(K-2)/2 = (K^2 + K - 4)/2$. It is closed, Hausdorff, and S_K-invariant under $\pi \cdot (\boldsymbol{v}, \mathbf{Y}) = (\mathbf{P}_\pi \boldsymbol{v}, \mathbf{P}_\pi \mathbf{Y} \mathbf{P}_\pi^\top)$.

Definition 12 (Global Joint Invariant)**.** For $(\boldsymbol{v}, \mathbf{Y}) \in \mathcal{X}_K$, define $\tilde{g}(\boldsymbol{v}, \mathbf{Y}) = (\mathcal{P}(\boldsymbol{v}, \mathbf{Y}), \mathcal{O}(\mathbf{Y}))$ where $\mathcal{P}(\boldsymbol{v}, \mathbf{Y}) = \{\{(v_i, Y_{ii}) : i = 1, \ldots, K\}\}$ and $\mathcal{O}(\mathbf{Y}) = \{\{Y_{ij} : 1 \leq i < j \leq K\}\}$.

The key extension is that $\mathcal{P}$ contains *pairs* (v_i, Y_{ii}) coupling utilities with variances.

Definition 13 (Genericity Conditions)**.** We say $(\boldsymbol{v}, \mathbf{Y}) \in \mathcal{X}_K$ satisfies ($\tilde{\text{a}}$) if the pairs (v_i, Y_{ii}) are pairwise distinct and the off-diagonal entries are pairwise distinct, and ($\tilde{\text{b}}$) if condition (b) holds for $\mathbf{Y}$.

Theorem 10 (Generic Separation on $\mathcal{X}_K$)**.** *Define* $\tilde{\mathcal{B}} = \{(\boldsymbol{v}, \mathbf{Y}) \in \mathcal{X}_K : (\tilde{a}) \text{ or } (\tilde{b}) \text{ fails}\}$. *Then* $\tilde{\mathcal{B}}$ *is closed,* S_K*-invariant, and has measure zero. For* $(\boldsymbol{v}_1, \mathbf{Y}_1), (\boldsymbol{v}_2, \mathbf{Y}_2) \in \mathcal{X}_K \setminus \tilde{\mathcal{B}}$*:* $\tilde{g}(\boldsymbol{v}_1, \mathbf{Y}_1) = \tilde{g}(\boldsymbol{v}_2, \mathbf{Y}_2)$ *implies the pairs are in the same* S_K*-orbit.*

Proof. The proof follows Theorem 8. The recovery procedure orders pairs lexicographically to assign labels, then recovers off-diagonal entries via subset-sum intersections. The measure-zero argument extends since $v_i - v_j$ is nontrivial on $\mathcal{X}_K$. □

A.8 Extension to Utility-Covariance Pairs: Per-Alternative Invariants

This section establishes Theorem 1 (Generic Separation) and Theorem 2 (MLP Universal Approximation) from the main text.

Definition 14 (Per-Alternative Invariant g_j)**.** For $(\boldsymbol{v}, \mathbf{Y}) \in \mathcal{X}_K$ and index j, define $g_j(\boldsymbol{v}, \mathbf{Y}) = (\mathcal{T}_j(\boldsymbol{v}, \mathbf{Y}), \mathcal{O}_j(\mathbf{Y}))$ where:

$$\mathcal{T}_j(\boldsymbol{v}, \mathbf{Y}) = \{\{(v_k, Y_{kk}, Y_{jk}) : k \neq j\}\},$$
$$\mathcal{O}_j(\mathbf{Y}) = \{\{Y_{kl} : 1 \leq k < l \leq K,\, k \neq j,\, l \neq j\}\}.$$

This combines extensions from Sections A.6 and A.7: $\mathcal{T}_j$ contains *triples* (v_k, Y_{kk}, Y_{jk}) coupling each alternative k's utility and variance with its covariance with j, while $\mathcal{O}_j$ contains off-diagonal entries of $\mathbf{Y}_{-j,-j}$. The values $v_j = -\sum_{k \neq j} v_k$ and $Y_{jj} = K - \sum_{k \neq j} Y_{kk}$ are implicitly recoverable.

Definition 15 (Genericity Conditions)**.** We say $(\boldsymbol{v}, \mathbf{Y}) \in \mathcal{X}_K$ satisfies ($\tilde{\text{a}}_j$) if the triples (v_k, Y_{kk}, Y_{jk}) for $k \neq j$ are pairwise distinct and the off-diagonal entries of $\mathbf{Y}_{-j,-j}$ are pairwise distinct, and ($\tilde{\text{b}}_j$) if condition (b_j) holds.

Definition 16 (Exceptional Set)**.** Define $\mathcal{B}_j = \{(\boldsymbol{v}, \mathbf{Y}) \in \mathcal{X}_K : (\tilde{\text{a}}_j) \text{ or } (\tilde{\text{b}}_j) \text{ fails}\}$.

Proposition 8 (Properties of $\mathcal{B}_j$)**.** *The set* $\mathcal{B}_j$ *is closed,* $S_{K-1}^{(j)}$*-invariant, and has measure zero in* $\mathcal{X}_K$*.*

Proof. The proof follows Proposition 6. Closedness holds because ($\tilde{\text{a}}_j$) and ($\tilde{\text{b}}_j$) are open conditions. Invariance under $S_{K-1}^{(j)}$ holds because permutations fixing j preserve distinctness

and subset-sum structure. For measure zero: coincidence of two triples defines the intersection of three hyperplanes $\{v_k = v_l\} \cap \{Y_{kk} = Y_{ll}\} \cap \{Y_{jk} = Y_{jl}\}$, each nontrivial; the subset-sum constraints are similarly nontrivial by arguments analogous to Lemma 3(ii). □

Lemma 4 (Recovery Lemma)**.** *If $(\boldsymbol{v}, \mathbf{Y}) \in \mathcal{X}_K$ satisfies $(\tilde{a}_j)$ and $(\tilde{b}_j)$, then $g_j(\boldsymbol{v}, \mathbf{Y})$ determines $(\boldsymbol{v}, \mathbf{Y})$ up to $S_{K-1}^{(j)}$-orbit.*

Proof. The proof extends Lemma 2 to triples. Order triples in $\mathcal{T}_j$ lexicographically to assign labels $(1), \ldots, (K-1)$. For each (k), identify $\mathcal{R}_{(k)} \subset \mathcal{O}_j$ as the unique $(K-2)$-subset summing to $-Y_{(k)(k)} - Y_{j(k)}$. Recover $Y_{(k)(l)}$ as the singleton $\mathcal{R}_{(k)} \cap \mathcal{R}_{(l)}$. Recover v_j and Y_{jj} from the constraints. □

We now prove the main results from Section 5.

Proof of Theorem 1. By Proposition 8, $\mathcal{B}_j$ is closed, $S_{K-1}^{(j)}$-invariant, and has measure zero. For $(\boldsymbol{v}_1, \mathbf{Y}_1), (\boldsymbol{v}_2, \mathbf{Y}_2) \in \mathcal{X}_K \setminus \mathcal{B}_j$ with $g_j(\boldsymbol{v}_1, \mathbf{Y}_1) = g_j(\boldsymbol{v}_2, \mathbf{Y}_2)$, Lemma 4 implies both pairs are determined up to $S_{K-1}^{(j)}$-orbit by this common value. □

Proof of Theorem 2. Encode g_j via continuous functions: sort triples lexicographically to obtain $3(K-1)$ functions $g_{k,1}^t = v_{(k)}$, $g_{k,2}^t = Y_{(k)(k)}$, $g_{k,3}^t = Y_{j(k)}$; sort off-diagonal entries to obtain $(K-1)(K-2)/2$ functions g_ℓ^o. These are continuous and $S_{K-1}^{(j)}$-invariant, and separate points on $(\mathcal{X}_K \setminus \mathcal{B}_j)/S_{K-1}^{(j)}$ by Theorem 1.

Since $S_{K-1}^{(j)}$ is finite, $\mathcal{X}_K / S_{K-1}^{(j)}$ is Hausdorff (Bredon, 1972). By Stone–Weierstrass, the algebra generated by these functions is dense on compact subsets. By the universal approximation theorem (Cybenko, 1989; Hornik et al., 1989), MLPs can approximate any continuous function of these inputs. Because the choice probabilities are defined according to the integral given in (4) under Assumption 1, they are continuous functions; thus, the result holds. □

B Estimation Theory

This appendix contains additional information on the estimation theory results in Sections 5.2 and 5.3. Section B.1 provides additional discussion of the approximation conditions: Assumptions 2 and 4. Section B.2 provides a proof of Theorem 4. Section B.3 contains the regularity conditions for Section 5.3. Section B.4 provides a proof of Theorem 6.

B.1 Emulator Approximation Assumptions

This section provides additional discussion of Assumptions 2 and 4, showing how they can be derived from more primitive conditions on the emulator and justifying these conditions through related theoretical results.

Primitive assumptions. We consider a sieve regime in which a sequence of emulators $\hat{P}_n$ is trained with increasing precision as the sample size n grows. This is achieved by progressively (a) increasing the number of simulated training examples $s = s_n$ and (b) increasing the neural network complexity $c = c_n$ (e.g., the number of layers or hidden units). We impose the following primitive conditions:

(P1) *Compact covariate support*: The covariate space $\mathcal{C}$ is compact.

(P2) *Smooth utility and covariance mappings*: The mappings $\boldsymbol{\theta} \mapsto \boldsymbol{v}_i^*(\mathbf{X}_i, \boldsymbol{\theta})$ and $\boldsymbol{\theta} \mapsto \boldsymbol{\Sigma}^*(\mathbf{X}_i, \boldsymbol{\theta})$ are continuously differentiable for all $\mathbf{X}_i \in \mathcal{C}$.

(P3) *Interior probability condition*: There exist a compact set $\bar{\mathcal{X}}_K \subset \mathcal{X}_K$ and $\underline{p} > 0$ such that $\inf_{(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*) \in \bar{\mathcal{X}}_K} \min_j P_j(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*) \geq \underline{p}$ and the image of $\mathcal{C} \times \Theta$ under the mapping $(\mathbf{X}, \boldsymbol{\theta}) \mapsto (\boldsymbol{v}^*(\mathbf{X}, \boldsymbol{\theta}), \boldsymbol{\Sigma}^*(\mathbf{X}, \boldsymbol{\theta}))$ is contained in $\bar{\mathcal{X}}_K$.

(P4) *Probability-scale approximation*: For some $\alpha \geq 0$, the emulator satisfies

$$\sup_{(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*) \in \bar{\mathcal{X}}_K} \max_{j=1,\ldots,K} |\hat{P}_{n,j}(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*) - P_j(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*)| = o_p(n^{-\alpha/2}). \tag{49}$$

(P5) *Gradient approximation*: The emulator gradients satisfy

$$\sup_{(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*) \in \bar{\mathcal{X}}_K} \max_{j=1,\ldots,K} \left\| \nabla_{(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*)} \log \hat{P}_{n,j}(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*) - \nabla_{(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*)} \log P_j(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*) \right\| = o_p(1). \tag{50}$$

We next show how these assumptions imply Assumptions 2 and 4. We then provide justification of assumptions (P4) and (P5) using related results in approximation theory.

From probability-scale errors to log-likelihood errors. We now show that conditions (P1)–(P4) imply Assumption 2. Write the log-likelihood error as

$$\hat{\ell}_n(\boldsymbol{\theta}) - \ell_n(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \xi_i, \quad \text{where} \quad \xi_i = \log \hat{P}_{n,y_i}(\boldsymbol{\theta}) - \log P_{y_i}(\boldsymbol{\theta}), \tag{51}$$

and $y_i \in \{1, \ldots, K\}$ denotes the observed choice. Conditional on covariates and the trained emulator, the ξ_i are independent. We decompose into bias and centered components:

$$\hat{\ell}_n(\boldsymbol{\theta}) - \ell_n(\boldsymbol{\theta}) = \underbrace{\frac{1}{n}\sum_{i=1}^{n} \mathbb{E}(\xi_i \mid \mathbf{X}_i)}_{\text{bias}} + \underbrace{\frac{1}{n}\sum_{i=1}^{n} \{\xi_i - \mathbb{E}(\xi_i \mid \mathbf{X}_i)\}}_{\text{centered term}}. \tag{52}$$

Let $\delta_n = \sup_{(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*) \in \bar{\mathcal{X}}_K} \max_j |\hat{P}_{n,j} - P_j| = o_p(n^{-\alpha/2})$ denote the probability-scale error from (P4). Taylor expanding $\log \hat{P}_{n,j}$ around P_j gives

$$\log \hat{P}_{n,j} - \log P_j = \frac{\hat{P}_{n,j} - P_j}{P_j} - \frac{(\hat{P}_{n,j} - P_j)^2}{2P_j^2} + O\left(\frac{\delta_n^3}{\underline{p}^3}\right). \tag{53}$$

The conditional expectation of ξ_i is

$$\mathbb{E}(\xi_i \mid \mathbf{X}_i) = \sum_{j=1}^{K} P_j (\log \hat{P}_{n,j} - \log P_j). \tag{54}$$

The first-order contribution is $\sum_j P_j \cdot (\hat{P}_{n,j} - P_j)/P_j = \sum_j (\hat{P}_{n,j} - P_j) = 0$, since both probability vectors sum to one. Therefore,

$$\mathbb{E}(\xi_i \mid \mathbf{X}_i) = -\sum_{j=1}^{K} \frac{(\hat{P}_{n,j} - P_j)^2}{2P_j} + O\left(\frac{\delta_n^3}{\underline{p}^3}\right) = O\left(\frac{\delta_n^2}{\underline{p}}\right). \tag{55}$$

For the centered term, the mean value theorem gives $|\log \hat{P}_{n,j} - \log P_j| \le |\hat{P}_{n,j} - P_j|/\underline{p} \le \delta_n/\underline{p}$, and hence $|\xi_i| \le \delta_n/\underline{p}$. The conditional variance satisfies $\mathrm{Var}(\xi_i \mid \mathbf{X}_i) \le \mathbb{E}(\xi_i^2 \mid \mathbf{X}_i) \le \delta_n^2/\underline{p}^2$. Since the centered terms $\xi_i - \mathbb{E}(\xi_i \mid \mathbf{X}_i)$ are conditionally independent with mean zero, the variance of their sample average is

$$\mathrm{Var}\left[\frac{1}{n}\sum_{i=1}^{n} \{\xi_i - \mathbb{E}(\xi_i \mid \mathbf{X}_i)\} \;\middle|\; \mathbf{X}_1, \ldots, \mathbf{X}_n\right] = \frac{1}{n^2}\sum_{i=1}^{n} \mathrm{Var}(\xi_i \mid \mathbf{X}_i) \le \frac{\delta_n^2}{n\underline{p}^2}. \tag{56}$$

By Chebyshev's inequality, the centered term is $O_p\left(\frac{\delta_n}{\underline{p}\sqrt{n}}\right)$.

Combining via the triangle inequality:

$$|\hat{\ell}_n(\boldsymbol{\theta}) - \ell_n(\boldsymbol{\theta})| = O\left(\frac{\delta_n^2}{\underline{p}}\right) + O_p\left(\frac{\delta_n}{\sqrt{n}\underline{p}}\right). \tag{57}$$

For $\alpha \in [0,1]$, the bias term $O(\delta_n^2/\underline{p}) = o(n^{-\alpha})$ dominates the centered term $O_p\left(\frac{\delta_n}{\underline{p}\sqrt{n}}\right) = o_p\left(n^{-(\alpha+1)/2}\right)$. Since conditions (P1) and (P2) ensure that the mapping from $(\mathbf{X}, \boldsymbol{\theta})$ to $(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*)$ is continuous on the compact set $\mathcal{C} \times \Theta$, the bound holds uniformly over $\boldsymbol{\theta} \in \Theta$:

$$\sup_{\boldsymbol{\theta}\in\Theta} |\hat{\ell}_n(\boldsymbol{\theta}) - \ell_n(\boldsymbol{\theta})| = o_p(n^{-\alpha}), \tag{58}$$

which is Assumption 2.

From gradient approximation to Fisher information consistency. We now show that conditions (P1), (P2), (P3), and (P5) imply Assumption 4. By the chain rule,

$$\nabla_{\boldsymbol{\theta}} \log \hat{P}_{n,j}(\boldsymbol{\theta}) = \{\nabla_{\boldsymbol{\theta}}(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*)\}^\top \nabla_{(\boldsymbol{v}^*,\boldsymbol{\Sigma}^*)} \log \hat{P}_{n,j}, \tag{59}$$

and similarly for $\nabla_{\boldsymbol{\theta}} \log P_j(\boldsymbol{\theta})$. By condition (P2), the Jacobian $\nabla_{\boldsymbol{\theta}}(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*)$ is continuous on the compact set $\mathcal{C} \times \Theta$ and hence uniformly bounded: $\sup_{(\mathbf{X},\boldsymbol{\theta})\in\mathcal{C}\times\Theta} \|\nabla_{\boldsymbol{\theta}}(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*)\| \le C$ for some constant $C < \infty$. Let

$$\varepsilon_n = \sup_{(\boldsymbol{v}^*,\boldsymbol{\Sigma}^*)\in\bar{\mathcal{X}}_K} \max_j \left\| \nabla_{(\boldsymbol{v}^*,\boldsymbol{\Sigma}^*)} \log \hat{P}_{n,j} - \nabla_{(\boldsymbol{v}^*,\boldsymbol{\Sigma}^*)} \log P_j \right\| = o_p(1) \tag{60}$$

denote the gradient approximation error from (P5). Then

$$\sup_{\boldsymbol{\theta}\in\Theta} \left\| \nabla_{\boldsymbol{\theta}} \log \hat{P}_{n,j}(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}} \log P_j(\boldsymbol{\theta}) \right\| \le C\varepsilon_n = o_p(1). \tag{61}$$

Recall the emulator and true outer-product-of-scores estimators are

$$\hat{\mathbf{J}}_n(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^n \nabla_{\boldsymbol{\theta}} \log \hat{P}_{n,y_i}(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log \hat{P}_{n,y_i}(\boldsymbol{\theta})^\top, \tag{62}$$

$$\mathbf{J}_n(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^n \nabla_{\boldsymbol{\theta}} \log P_{y_i}(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log P_{y_i}(\boldsymbol{\theta})^\top. \tag{63}$$

Let $\hat{\boldsymbol{s}}_i = \nabla_{\boldsymbol{\theta}} \log \hat{P}_{n,y_i}(\boldsymbol{\theta})$ and $\boldsymbol{s}_i = \nabla_{\boldsymbol{\theta}} \log P_{y_i}(\boldsymbol{\theta})$. The identity

$$\hat{\boldsymbol{s}}_i\hat{\boldsymbol{s}}_i^\top - \boldsymbol{s}_i\boldsymbol{s}_i^\top = (\hat{\boldsymbol{s}}_i - \boldsymbol{s}_i)\hat{\boldsymbol{s}}_i^\top + \boldsymbol{s}_i(\hat{\boldsymbol{s}}_i - \boldsymbol{s}_i)^\top \tag{64}$$

implies

$$\|\hat{\boldsymbol{s}}_i\hat{\boldsymbol{s}}_i^\top - \boldsymbol{s}_i\boldsymbol{s}_i^\top\| \le \|\hat{\boldsymbol{s}}_i - \boldsymbol{s}_i\| \cdot (\|\hat{\boldsymbol{s}}_i\| + \|\boldsymbol{s}_i\|). \tag{65}$$

The true scores $\boldsymbol{s}_i$ are uniformly bounded: (P3) ensures that all relevant inputs $(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*)$ lie in $\bar{\mathcal{X}}_K$ where $P_j \geq \underline{p} > 0$, guaranteeing bounded gradients $\nabla_{(\boldsymbol{v}^*,\boldsymbol{\Sigma}^*)} \log P_j$, and (P2) ensures the Jacobian $\nabla_{\boldsymbol{\theta}}(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*)$ is continuous and hence bounded on $\mathcal{C} \times \Theta$. Let $M = \sup_{\boldsymbol{\theta}\in\Theta} \max_j \|\nabla_{\boldsymbol{\theta}} \log P_j(\boldsymbol{\theta})\| < \infty$, so that $\|\boldsymbol{s}_i\| \leq M$ for all i. Since $\|\hat{\boldsymbol{s}}_i - \boldsymbol{s}_i\| \leq C\varepsilon_n = o_p(1)$, the emulator scores satisfy $\|\hat{\boldsymbol{s}}_i\| \leq \|\boldsymbol{s}_i\| + C\varepsilon_n \leq M + o_p(1)$. Therefore,

$$\begin{aligned}\sup_{\boldsymbol{\theta}\in\Theta} \|\hat{\mathbf{J}}_n(\boldsymbol{\theta}) - \mathbf{J}_n(\boldsymbol{\theta})\| &\leq \frac{1}{n}\sum_{i=1}^{n} \|\hat{\boldsymbol{s}}_i - \boldsymbol{s}_i\| \cdot (\|\hat{\boldsymbol{s}}_i\| + \|\boldsymbol{s}_i\|) \\ &\leq \frac{1}{n}\sum_{i=1}^{n} C\varepsilon_n \cdot \{2M + o_p(1)\} = o_p(1), \end{aligned} \tag{66}$$

which is Assumption 4.

Justification from approximation theory. The primitive conditions (P4) and (P5) are supported by related theoretical results on neural network approximation and estimation, though a complete theory tailored to our specific setting remains an area for future work.

The total emulator error can be decomposed into approximation error and estimation error:

$$\sup_{(\boldsymbol{v}^*,\boldsymbol{\Sigma}^*)\in\bar{\mathcal{X}}_K} \max_{j=1,\ldots,K} |\hat{P}_{n,j}(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*) - P_j(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*)| \leq \underbrace{a(c_n)}_{\text{approximation}} + \underbrace{e(s_n, c_n)}_{\text{estimation}}, \tag{67}$$

where $a(c)$ is the distance from the true choice probability function to the best approximator in the network class of complexity c, and $e(s, c)$ is the deviation of the trained network from this best-in-class approximator due to training on s simulated examples.

For the approximation error, provided the choice probabilities are real-analytic functions of $(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*)$ on the interior of the parameter space, neural networks with smooth activations can approximate them with exponentially small error. De Ryck et al. (2021) establish that tanh networks with two hidden layers achieve approximation error decaying as $O\left\{N^{k/(d+1)} \exp(-C \cdot N^{1/(d+1)} \log N)\right\}$ in Sobolev norms $W^{k,\infty}$, where N is the network width, d is the input dimension, and $C > 0$ depends on the analyticity parameters of the target function. Earlier work established similar exponential rates for deep ReLU networks (E and Wang, 2018), though under the stronger assumption that the target function admits an absolutely convergent power series expansion. Given these exponential convergence results, achieving approximation error ε requires network complexity scaling only polylogarithmically in $1/\varepsilon$.

For the estimation error, standard results from empirical process theory yield bounds of order $O\left\{\sqrt{c\log(sc)/s}\right\}$ in L^2 norms, where c is a complexity parameter (Schmidt-Hieber, 2020; Farrell et al., 2021; Shen et al., 2023). For our setting, however, we require an L^∞ bound. The results below assume minimally that $e(s,c) = O\left\{c^k \text{ polylog}(s)/\sqrt{s}\right\}$ for some $k < \infty$. Then, with network complexity growing polylogarithmically in the target accuracy, the dependence on c is negligible, and the estimation error is effectively of order $O\left(1/\sqrt{s}\right)$ up to logarithmic factors.

Achieving the rate given in (P4) requires that both $a(c_n) = o(n^{-\alpha/2})$ and $e(s_n, c_n) = o_p(n^{-\alpha/2})$; note that the approximation error is nonstochastic because it concerns pure function approximation. The exponential approximation rates for analytic functions imply that the first condition is satisfied with polylogarithmic growth in c_n. The second condition requires s_n to grow faster than n^α up to logarithmic factors. In practice, since simulation from the MNP model is computationally inexpensive, generating a training set with $s_n \gg n$ is feasible.

The cases $\alpha = 0$ and $\alpha = 1$ correspond to different inferential goals. When $\alpha = 0$, Assumption 2 requires only that the log-likelihood error vanish in probability, which suffices for consistency of the emulator-based estimator (Theorem 3). When $\alpha = 1$, the stronger requirement that the error vanish faster than n^{-1} ensures that the emulator-based estimator is asymptotically equivalent to the true MLE, inheriting its $\sqrt{n}$-consistency and asymptotic normality (Theorem 4). The latter requires a larger training set: s_n growing faster than n versus s_n growing faster than $n^0 = 1$.

For the gradient approximation in (P5), Czarnecki et al. (2017) show that neural networks are universal approximators in Sobolev norms, and the bounds of De Ryck et al. (2021) hold in $W^{k,\infty}$ norms, establishing that smooth networks can simultaneously approximate both function values and derivatives. Additionally, Cocola and Hand (2020) prove global convergence of gradient flow for Sobolev training with overparameterized two-layer networks, ensuring that the trained network achieves small Sobolev loss under appropriate conditions. These results support the plausibility of (P5), though the precise rates and conditions for our setting merit further investigation.

B.2 Proof of Theorem 4

Proof. We prove the result for approximate maximizers satisfying $\hat{\ell}_n(\hat{\boldsymbol{\theta}}_n) \geq \sup_{\boldsymbol{\theta}\in\Theta} \hat{\ell}_n(\boldsymbol{\theta}) - \eta_n$ where $\eta_n = o_p(n^{-1})$. Exact maximizers correspond to $\eta_n = 0$. We proceed in three

steps: first bounding the difference in log likelihoods, then showing that the estimators are asymptotically equivalent, and finally appealing to the asymptotic normality of the exact MLE.

Step 1: Bounding the difference in log likelihoods.

Define $\delta_n = \sup_{\boldsymbol{\theta}\in\Theta} |\hat{\ell}_n(\boldsymbol{\theta}) - \ell_n(\boldsymbol{\theta})| = o_p(n^{-1})$ by Assumption 2 with $\alpha \geq 1$. By the triangle inequality,

$$|\ell_n(\hat{\boldsymbol{\theta}}_n) - \ell_n(\tilde{\boldsymbol{\theta}}_n)| \leq |\ell_n(\hat{\boldsymbol{\theta}}_n) - \hat{\ell}_n(\hat{\boldsymbol{\theta}}_n)| + |\hat{\ell}_n(\hat{\boldsymbol{\theta}}_n) - \ell_n(\tilde{\boldsymbol{\theta}}_n)|. \tag{68}$$

By Assumption 2, the first term is bounded by δ_n. For the second term, we establish upper and lower bounds on $\hat{\ell}_n(\hat{\boldsymbol{\theta}}_n)$. Since $\tilde{\boldsymbol{\theta}}_n$ maximizes ℓ_n:

$$\hat{\ell}_n(\hat{\boldsymbol{\theta}}_n) \leq \ell_n(\hat{\boldsymbol{\theta}}_n) + \delta_n \leq \ell_n(\tilde{\boldsymbol{\theta}}_n) + \delta_n. \tag{69}$$

For the lower bound, by the approximate maximizer property and the fact that $\tilde{\boldsymbol{\theta}}_n \in \Theta$:

$$\hat{\ell}_n(\hat{\boldsymbol{\theta}}_n) \geq \sup_{\boldsymbol{\theta}\in\Theta} \hat{\ell}_n(\boldsymbol{\theta}) - \eta_n \geq \hat{\ell}_n(\tilde{\boldsymbol{\theta}}_n) - \eta_n \geq \ell_n(\tilde{\boldsymbol{\theta}}_n) - \delta_n - \eta_n. \tag{70}$$

Combining these bounds, $|\hat{\ell}_n(\hat{\boldsymbol{\theta}}_n) - \ell_n(\tilde{\boldsymbol{\theta}}_n)| \leq \delta_n + \eta_n$. Therefore,

$$|\ell_n(\hat{\boldsymbol{\theta}}_n) - \ell_n(\tilde{\boldsymbol{\theta}}_n)| \leq 2\delta_n + \eta_n = o_p(n^{-1}), \tag{71}$$

which implies $n|\ell_n(\hat{\boldsymbol{\theta}}_n) - \ell_n(\tilde{\boldsymbol{\theta}}_n)| = o_p(1)$.

Step 2: Asymptotic equivalence of the estimators.

We show that $\sqrt{n}(\hat{\boldsymbol{\theta}}_n - \tilde{\boldsymbol{\theta}}_n) \xrightarrow{p} \mathbf{0}$. Fix $\epsilon > 0$ and define the neighborhood

$$B_{n,\epsilon} = \left\{ \boldsymbol{\theta} \in \Theta : \sqrt{n}\|\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}}_n\| \leq \epsilon \right\}. \tag{72}$$

We show that $\hat{\boldsymbol{\theta}}_n \in B_{n,\epsilon}$ with probability approaching one.

For any $\boldsymbol{\theta} \notin B_{n,\epsilon}$, we have $\|\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}}_n\| > \epsilon/\sqrt{n}$. Since $\tilde{\boldsymbol{\theta}}_n$ maximizes ℓ_n, the score vanishes: $\frac{1}{n}\sum_{i=1}^n \psi(y_i, \mathbf{X}_i; \tilde{\boldsymbol{\theta}}_n) = \mathbf{0}$. By Taylor expansion around $\tilde{\boldsymbol{\theta}}_n$,

$$\ell_n(\boldsymbol{\theta}) = \ell_n(\tilde{\boldsymbol{\theta}}_n) + \frac{1}{2}(\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}}_n)^\top \nabla^2_{\boldsymbol{\theta}} \ell_n(\bar{\boldsymbol{\theta}})(\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}}_n), \tag{73}$$

where $\bar{\boldsymbol{\theta}}$ lies on the segment between $\boldsymbol{\theta}$ and $\tilde{\boldsymbol{\theta}}_n$. By consistency, both $\hat{\boldsymbol{\theta}}_n \xrightarrow{p} \boldsymbol{\theta}_0$ and $\tilde{\boldsymbol{\theta}}_n \xrightarrow{p} \boldsymbol{\theta}_0$,

so $\bar{\boldsymbol{\theta}} \xrightarrow{p} \boldsymbol{\theta}_0$. By the uniform law of large numbers and Assumption 3 (ii) and (iv),

$$\nabla^2_{\boldsymbol{\theta}} \ell_n(\bar{\boldsymbol{\theta}}) \xrightarrow{p} -\mathbf{J}(\boldsymbol{\theta}_0). \tag{74}$$

Let $\lambda_{\min} > 0$ denote the minimum eigenvalue of $\mathbf{J}(\boldsymbol{\theta}_0)$, which is positive by Assumption 3(v). For sufficiently large n, with probability approaching one, the Hessian $\nabla^2_{\boldsymbol{\theta}} \ell_n(\bar{\boldsymbol{\theta}})$ has all eigenvalues bounded above by $-\lambda_{\min}/2$. Thus, for $\boldsymbol{\theta} \notin B_{n,\epsilon}$,

$$n\left\{\ell_n(\tilde{\boldsymbol{\theta}}_n) - \ell_n(\boldsymbol{\theta})\right\} \geq \frac{\lambda_{\min}}{4} n\|\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}}_n\|^2 > \frac{\lambda_{\min}}{4} \cdot \epsilon^2 = \kappa > 0 \tag{75}$$

with probability approaching one as $n \to \infty$.

Now suppose $\Pr(\hat{\boldsymbol{\theta}}_n \notin B_{n,\epsilon}) \not\to 0$. On the event that both $\hat{\boldsymbol{\theta}}_n \notin B_{n,\epsilon}$ and the Hessian bound holds, we have $n(\ell_n(\tilde{\boldsymbol{\theta}}_n) - \ell_n(\hat{\boldsymbol{\theta}}_n)) > \kappa$. Since the Hessian bound holds with probability approaching one, this event has probability bounded away from zero, which implies that $\Pr\left[\left\{n(\ell_n(\tilde{\boldsymbol{\theta}}_n) - \ell_n(\hat{\boldsymbol{\theta}}_n)\right\} > \kappa\right] \not\to 0$. But this contradicts $n|\ell_n(\hat{\boldsymbol{\theta}}_n) - \ell_n(\tilde{\boldsymbol{\theta}}_n)| = o_p(1)$ from Step 1. Therefore, $\Pr(\hat{\boldsymbol{\theta}}_n \in B_{n,\epsilon}) \to 1$.

Since $\epsilon > 0$ was arbitrary, we conclude that $\sqrt{n}(\hat{\boldsymbol{\theta}}_n - \tilde{\boldsymbol{\theta}}_n) \xrightarrow{p} \mathbf{0}$.

Step 3: Asymptotic normality.

By standard maximum likelihood theory (Theorem 5.39 and Lemma 7.6 of van der Vaart (1998)), the regularity conditions in Assumption 3 ensure that the model is differentiable in quadratic mean at $\boldsymbol{\theta}_0$ with nonsingular Fisher information. Since $\tilde{\boldsymbol{\theta}}_n$ is consistent, $\sqrt{n}(\tilde{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0) \xrightarrow{d} \mathcal{N}\left\{\mathbf{0}, \mathbf{J}(\boldsymbol{\theta}_0)^{-1}\right\}$. Since $\sqrt{n}(\hat{\boldsymbol{\theta}}_n - \tilde{\boldsymbol{\theta}}_n) \xrightarrow{p} \mathbf{0}$, Slutsky's theorem gives

$$\sqrt{n}(\hat{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0) = \sqrt{n}(\hat{\boldsymbol{\theta}}_n - \tilde{\boldsymbol{\theta}}_n) + \sqrt{n}(\tilde{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0) \xrightarrow{d} \mathcal{N}\left\{\mathbf{0}, \mathbf{J}(\boldsymbol{\theta}_0)^{-1}\right\}. \tag{76}$$

□

B.3 Regularity Conditions for Inference Under Misspecification

This appendix states the regularity conditions required for the results in Section 5.3. These conditions parallel standard assumptions for M-estimators (van der Vaart, 1998, Chapter 5) and are mild for neural network emulators with smooth activation functions. Recall that in Section 5.3, we treat the neural network as fixed (not varying with n), so we use $\hat{m}$ and $\hat{\psi}$ without n subscripts.

Assumption 5 (Identification and Uniform Convergence)**.** The following conditions hold:

(i) The parameter space Θ is compact.

(ii) The pseudo-true parameter $\boldsymbol{\theta}_0^\dagger = \arg\max_{\boldsymbol{\theta}\in\Theta} \mathbb{E}\left\{\hat{m}(y, \mathbf{X}; \boldsymbol{\theta})\right\}$ exists uniquely and lies in the interior of Θ.

(iii) The pseudo-true parameter is well-separated: for every $\epsilon > 0$,

$$\sup_{\boldsymbol{\theta}:\|\boldsymbol{\theta}-\boldsymbol{\theta}_0^\dagger\|\geq\epsilon} \mathbb{E}\left\{\hat{m}(y, \mathbf{X}; \boldsymbol{\theta})\right\} < \mathbb{E}\left\{\hat{m}(y, \mathbf{X}; \boldsymbol{\theta}_0^\dagger)\right\}. \tag{77}$$

(iv) The sample criterion function converges uniformly to its expectation:

$$\sup_{\boldsymbol{\theta}\in\Theta} \left| \frac{1}{n}\sum_{i=1}^{n} \hat{m}(y_i, \mathbf{X}_i; \boldsymbol{\theta}) - \mathbb{E}[\hat{m}(y, \mathbf{X}; \boldsymbol{\theta})] \right| \xrightarrow{p} 0. \tag{78}$$

(v) The estimator $\hat{\boldsymbol{\theta}}_n$ is an approximate maximizer of the sample criterion function:

$$\frac{1}{n}\sum_{i=1}^{n} \hat{m}(y_i, \mathbf{X}_i; \hat{\boldsymbol{\theta}}_n) \geq \sup_{\boldsymbol{\theta}\in\Theta} \frac{1}{n}\sum_{i=1}^{n} \hat{m}(y_i, \mathbf{X}_i; \boldsymbol{\theta}) - o_p(n^{-1}). \tag{79}$$

Assumption 6 (Regularity for Asymptotic Normality)**.** The following conditions hold:

(i) (Differentiability) For each $(y, \mathbf{X})$, the map $\boldsymbol{\theta} \mapsto \hat{m}(y, \mathbf{X}; \boldsymbol{\theta})$ is differentiable on Θ with gradient $\hat{\psi}(y, \mathbf{X}; \boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \hat{m}(y, \mathbf{X}; \boldsymbol{\theta})$.

(ii) (Local dominance) There exists $\delta > 0$ and a measurable function $\dot{m}(y, \mathbf{X})$ having $\mathbb{E}[\dot{m}(y, \mathbf{X})^2] < \infty$ such that

$$\sup_{\boldsymbol{\theta}:\|\boldsymbol{\theta}-\boldsymbol{\theta}_0^\dagger\|\leq\delta} \|\hat{\psi}(y, \mathbf{X}; \boldsymbol{\theta})\| \leq \dot{m}(y, \mathbf{X}). \tag{80}$$

(iii) (Second-order expansion) The map $\boldsymbol{\theta} \mapsto \mathbb{E}\left\{\hat{m}(y, \mathbf{X}; \boldsymbol{\theta})\right\}$ admits a second-order Taylor expansion at $\boldsymbol{\theta}_0^\dagger$:

$$\mathbb{E}\left\{\hat{m}(y, \mathbf{X}; \boldsymbol{\theta})\right\} = \mathbb{E}\left\{\hat{m}(y, \mathbf{X}; \boldsymbol{\theta}_0^\dagger)\right\} - \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_0^\dagger)^\top \mathbf{A}(\boldsymbol{\theta}_0^\dagger)(\boldsymbol{\theta} - \boldsymbol{\theta}_0^\dagger) + o(\|\boldsymbol{\theta} - \boldsymbol{\theta}_0^\dagger\|^2), \tag{81}$$

where $\mathbf{A}(\boldsymbol{\theta}_0^\dagger) = -\mathbb{E}\left\{\nabla^2_{\boldsymbol{\theta}} \hat{m}(y, \mathbf{X}; \boldsymbol{\theta}_0^\dagger)\right\}$ is positive definite.

B.4 Proof of Theorems 5 and 6

Proof of Theorem 5. Define the sample and population criterion functions

$$M_n(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \hat{m}(y_i, \mathbf{X}_i; \boldsymbol{\theta}), \qquad M(\boldsymbol{\theta}) = \mathbb{E}\left\{\hat{m}(y, \mathbf{X}; \boldsymbol{\theta})\right\}. \tag{82}$$

By Assumption 5(iv), $\sup_{\boldsymbol{\theta}\in\Theta} |M_n(\boldsymbol{\theta}) - M(\boldsymbol{\theta})| \xrightarrow{p} 0$. By Assumption 5(ii)–(iii), $\boldsymbol{\theta}_0^\dagger$ is the unique maximizer of $M(\boldsymbol{\theta})$ and is well-separated in the sense that $\sup_{\boldsymbol{\theta}:\|\boldsymbol{\theta}-\boldsymbol{\theta}_0^\dagger\|\geq\epsilon} M(\boldsymbol{\theta}) < M(\boldsymbol{\theta}_0^\dagger)$ for every $\epsilon > 0$. By Assumption 5(v), $M_n(\hat{\boldsymbol{\theta}}_n) \geq \sup_{\boldsymbol{\theta}\in\Theta} M_n(\boldsymbol{\theta}) - o_p(n^{-1})$, which implies $M_n(\hat{\boldsymbol{\theta}}_n) \geq \sup_{\boldsymbol{\theta}\in\Theta} M_n(\boldsymbol{\theta}) - o_p(1)$. The result $\hat{\boldsymbol{\theta}}_n \xrightarrow{p} \boldsymbol{\theta}_0^\dagger$ now follows from Theorem 5.7 of van der Vaart (1998). □

Proof of Theorem 6. We verify the conditions of Theorem 5.23 of van der Vaart (1998) and apply it to obtain the result.

By Assumption 6(i), the map $\boldsymbol{\theta} \mapsto \hat{m}(y, \mathbf{X}; \boldsymbol{\theta})$ is differentiable at $\boldsymbol{\theta}_0^\dagger$ for every $(y, \mathbf{X})$, with derivative $\hat{\psi}(y, \mathbf{X}; \boldsymbol{\theta}_0^\dagger)$. By Assumption 6(ii), the score is locally dominated by a square-integrable function, which implies the Lipschitz condition

$$|\hat{m}(y, \mathbf{X}; \boldsymbol{\theta}_1) - \hat{m}(y, \mathbf{X}; \boldsymbol{\theta}_2)| \leq \dot{m}(y, \mathbf{X})\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\| \tag{83}$$

for $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$ in a neighborhood of $\boldsymbol{\theta}_0^\dagger$ via the mean value theorem. Assumption 6(iii) provides the required second-order Taylor expansion of $\boldsymbol{\theta} \mapsto \mathbb{E}\left\{\hat{m}(y, \mathbf{X}; \boldsymbol{\theta})\right\}$ at the point of maximum $\boldsymbol{\theta}_0^\dagger$, with nonsingular second derivative matrix $-\mathbf{A}(\boldsymbol{\theta}_0^\dagger)$. By Theorem 5, $\hat{\boldsymbol{\theta}}_n \xrightarrow{p} \boldsymbol{\theta}_0^\dagger$. By Assumption 5(v), the near-maximization condition $M_n(\hat{\boldsymbol{\theta}}_n) \geq \sup_{\boldsymbol{\theta}\in\Theta} M_n(\boldsymbol{\theta}) - o_p(n^{-1})$ is satisfied.

Applying Theorem 5.23 of van der Vaart (1998):

$$\sqrt{n}(\hat{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0^\dagger) = \mathbf{A}(\boldsymbol{\theta}_0^\dagger)^{-1}\frac{1}{\sqrt{n}}\sum_{i=1}^{n} \hat{\psi}(y_i, \mathbf{X}_i; \boldsymbol{\theta}_0^\dagger) + o_p(1). \tag{84}$$

By Assumption 6(ii), $\mathbb{E}\left\{\|\hat{\psi}(y, \mathbf{X}; \boldsymbol{\theta}_0^\dagger)\|^2\right\} \leq \mathbb{E}\left\{\dot{m}(y, \mathbf{X})^2\right\} < \infty$. The central limit theorem and Slutsky's lemma yield

$$\sqrt{n}(\hat{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0^\dagger) \xrightarrow{d} \mathcal{N}\left\{\mathbf{0}, \mathbf{A}(\boldsymbol{\theta}_0^\dagger)^{-1}\mathbf{B}(\boldsymbol{\theta}_0^\dagger)\mathbf{A}(\boldsymbol{\theta}_0^\dagger)^{-\top}\right\}, \tag{85}$$

where $\mathbf{B}(\boldsymbol{\theta}_0^\dagger) = \mathbb{E}\left\{\hat{\psi}(y, \mathbf{X}; \boldsymbol{\theta}_0^\dagger)\hat{\psi}(y, \mathbf{X}; \boldsymbol{\theta}_0^\dagger)^\top\right\}$. □

C Simulation Study Details

This appendix provides additional details on the factor model parameterization, emulator training procedure, and the scaled Wishart distribution used in the simulation study.

C.1 Emulator Training

We train the emulator using the Adam optimizer (Kingma and Ba, 2015) with the AdamW weight decay variant (Loshchilov and Hutter, 2019). The base learning rate is 0.003 with momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$, numerical stability constant $\epsilon = 10^{-8}$, and weight decay 3×10^{-6}.

Learning rate schedule. We use a learning rate schedule with linear warmup followed by inverse-square-root decay. During the first 5,000 steps, the learning rate increases linearly from zero to the base rate. After warmup, the learning rate decays as $\sqrt{t_{\text{warmup}}/t}$, where t is the current step and $t_{\text{warmup}} = 5{,}000$.

Dropout schedule. We apply dropout during training to encourage diversity among learned representations. We set the dropout rate to $1/(10t)$, where t indexes the training episodes from 1 to 400,000.

Training procedure. We pregenerate 400,000 training examples, each consisting of a canonicalized input $(\boldsymbol{v}^*, \boldsymbol{\Sigma}^*)$, simulated choice frequencies based on 10^6 Monte Carlo draws, and target gradients computed using the soft relaxation described in Section 4. During training, we randomly pair these examples with pre-computed tangent space directions for the directional derivative loss. Gradients are clipped to a maximum norm of 0.01. We train for 400,000 optimizer steps with a batch size of 10,000.

Network architecture. Table 8 summarizes the neural network architecture for each value of K. All networks use the Swish activation function. Larger values of K require larger networks to maintain approximation accuracy.

C.2 Derivation of Scaled Wishart PDF

This section derives the PDF of a scaled Wishart random matrix as described in Section 6.1.

Let $\mathbf{W} \sim W_d(\mathbf{V}, n)$ denote a $d \times d$ Wishart random matrix with scale matrix $\mathbf{V}$ and $n \geq d$ degrees of freedom. We derive the distribution of $\mathbf{\Sigma} = \mathbf{W}/W_{jj}$, where W_{jj} is the (j, j) element of $\mathbf{W}$ for some $j \in \{1, \ldots, d\}$.

The Wishart density is

$$f_{\mathbf{W}}(\mathbf{W}) = \frac{|\mathbf{W}|^{(n-d-1)/2} \exp\left\{-\frac{1}{2}\text{tr}(\mathbf{V}^{-1}\mathbf{W})\right\}}{2^{nd/2}|\mathbf{V}|^{n/2}\Gamma_d(n/2)} \tag{86}$$

where $\Gamma_d(\cdot)$ is the multivariate gamma function.

The inverse transformation is $\mathbf{W} = W_{jj}\mathbf{\Sigma}$. Note that $\mathbf{\Sigma}$ is positive definite with $\Sigma_{jj} = 1$, so it has $d(d+1)/2 - 1$ free parameters. Together with W_{jj}, this matches the $d(d+1)/2$ unique elements of $\mathbf{W}$. The Jacobian determinant of this transformation is $w_{jj}^{d(d+1)/2-1}$, where w_{jj} denotes the value of W_{jj}.

Substituting $\mathbf{W} = W_{jj}\mathbf{\Sigma}$ into the Wishart density and applying the Jacobian yields the joint density

$$f_{W_{jj},\mathbf{\Sigma}}(w_{jj}, \mathbf{\Sigma}) = \frac{|\mathbf{\Sigma}|^{(n-d-1)/2}}{2^{nd/2}|\mathbf{V}|^{n/2}\Gamma_d(n/2)} \cdot w_{jj}^{nd/2-1} \exp\left\{-\frac{w_{jj}}{2}\text{tr}(\mathbf{V}^{-1}\mathbf{\Sigma})\right\}. \tag{87}$$

Integrating over $w_{jj} > 0$ using the gamma integral gives the marginal density of $\mathbf{\Sigma}$:

$$f_{\mathbf{\Sigma}}(\mathbf{\Sigma}) = \frac{\Gamma(nd/2)}{2^{nd/2}|\mathbf{V}|^{n/2}\Gamma_d(n/2)} \cdot \frac{|\mathbf{\Sigma}|^{(n-d-1)/2}}{\left\{\frac{1}{2}\text{tr}(\mathbf{V}^{-1}\mathbf{\Sigma})\right\}^{nd/2}} \tag{88}$$

for symmetric, positive definite $\mathbf{\Sigma}$ with $\Sigma_{jj} = 1$ for some fixed $j \in \{1, \ldots, d\}$.

In the simulation study, we use $d = K - 1$, $n = K + 10$, $\mathbf{V} = \mathbf{I}_{K-1}$, and $j = 1$. Substituting these values and simplifying yields the density given in Section 6.1.

Component	$K = 3$	$K = 5$	$K = 10$
Diagonal DeepSet			
ϕ output dim	8	24	32
ϕ hidden layers	0	1	1
ϕ hidden dim	N/A	24	32
ρ output dim	8	24	32
ρ hidden layers	0	0	0
Parameters	152	1,440	2,432
Off-diagonal DeepSet			
ϕ output dim	8	24	32
ϕ hidden layers	0	1	1
ϕ hidden dim	N/A	24	32
ρ output dim	8	24	32
ρ hidden layers	0	0	0
Parameters	128	1,368	2,336
Combining MLP			
Output dim	8	24	32
Hidden dim	12	48	64
Hidden layers	2	2	2
Parameters	572	6,312	10,976
Equivariant layers			
Hidden dim	4	12	16
Hidden layers	1	1	1
Parameters	76	612	1,072
Total parameters	928	9,732	16,816

Table 8: Neural network architecture by number of alternatives K. All configurations use an initial dropout rate of 0.1 and Swish activations. When the number of hidden layers is 0, the hidden dimension is not applicable (N/A).