Mix It Up: Improving Performance in Travel Choice Modeling

Apostolos Avranas[1,*], Moaad Maaroufi[1], Alix Lheritier[1], Rodrigo Acuna Agost[1] and Eoin Thomas[1]

[1]*Amadeus, 821 Av. Jack Kilby, 06270 Villeneuve-Loubet*

Abstract

Discrete choice models are pivotal in describing, explaining, and predicting choices between two or more discrete alternatives, such as which mode of transport to take. These models often leverage features of the alternatives and contextual information about the decision-making process. In this work, we propose a novel process for improving choice modeling through feature engineering and an advanced data augmentation strategy known as mixup, which has not been previously applied in this domain. Our results show that the unconventional combination of two different sessions using mixup boosts the performance of choice models. We first introduce our process using a carefully designed Transformer model on a dataset focused on flight choices. Then to ensure the robustness of our process, we apply the process *unchanged* with different sized transformer models, as well as to previously proposed neural network architectures. We also verify the effectiveness of the process on another well-known public dataset for hotel room bookings.

Keywords

Discrete Choice Models, Transformer, Feature Engineering, Mixup

1. Introduction

Discrete choice modeling is the process of replicating the decision-making process made by a person when choosing among a set of distinct alternatives. These models are then used to better understand the factors that lead to decisions, or to predict individual decisions.

One application of such models is in the travel and transport domain. For example, discrete choice models were trained on survey data to predict and inform modes of transport, such as new rail lines [1]. Choice models are integral parts of many revenue management systems, performing tasks such as demand modeling and assortment optimization [2] and are widely used in travel choice modeling and travel offer pricing [3]. In a retrospective review [4], it was shown that machine learning models, including recommender systems and choice models, create clear business value and are implemented at various stages of the online travel booking flow.

One of the oldest choice modeling approaches is the multinomial logit (MNL) model, originally introduced by Luce [5]. It relies on the simplifying assumption of the *independence of irrelevant alternatives* (IIA), which states that the relative probabilities between two alternatives are not affected by the presence of additional alternatives. [6] later showed that the resulting model is a particular case of Random Utility Models (RUM), where decision-makers choose alternatives that maximize the obtained utility, which is considered to be a linear function of observable features of the alternatives plus a stochastic noise term. Despite its popularity, the MNL suffers from notable limitations—including its inability to model correlation between alternatives and its implication of proportional substitution across alternatives (e.g. an improvement in the attributes of an alternative reduces the probabilities for all the other alternatives by the same percentage)—which often make it unrealistic for complex, real-world decision-making scenarios [7].

To address these shortcomings, the field has increasingly turned toward machine learning and, more recently, deep learning approaches. These methods offer enhanced flexibility, scalability, and

Workshop on Recommenders in Tourism (RecTour 2025), September 22, 2025, co-located with the 19th ACM Conference on Recommender Systems, Prague, Czech Republic.

apostolos.avranas@amadeus.com (A. Avranas); alix.lheritier@amadeus.com (A. Lheritier); rodrigo.acunaagost@amadeus.com (R. A. Agost); eoin.thomas@amadeus.com (E. Thomas)

© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

CEUR-WS.org/Vol-4052/paper1.pdf

performance, as well as greater control over model design elements such as objective functions and user representations [8]. Notable efforts include the use of deep neural networks to parameterize flexible models such as Pairwise Choice Markov Chains (PCMC), which relax traditional assumptions like IIA and stochastic transitivity [9, 10]. It should be noted that despite the significant added complexity of these models, explainable AI tooling has also progressed significantly allowing for insights into these models as shown in [11].

In parallel, in 2017 the attention-based architecture called Transformers [12] was introduced, and since then they have transformed multiple domains, including natural language processing (text completion, translation and code generation among others) [13], vision [14], and signal processing [15]. Their ability to contextualize and compare elements/tokens with an attention mechanism, makes them particularly appealing for discrete choice tasks, where alternatives must be evaluated relative to one another—naturally overcoming the IIA limitation. However, best practices for applying Transformers to choice modeling remain underexplored.

Crucially, model architecture is only one dimension of success. Techniques such as feature engineering, regularization, and data augmentation are essential for improving generalization and robustness. In choice modeling, feature engineering through the design or transformation of input features—such as characteristics of alternatives or contextual variables—can significantly improve the model's ability to capture choice behaviour [16]. Regularization methods like dropout [17] and weight decay [18] are widely adopted to reduce overfitting. Data augmentation has been effectively used to generate synthetic data for mitigating data scarcity and imbalances [19]. In this work, we introduce mixup [20]—a data augmentation technique that linearly interpolates pairs of training samples—for the first time in the context of choice modeling. Perhaps surprisingly we show that combining features from unrelated choice sessions, improves model performance, particularly in ranking the true choice among top alternatives.

In this paper, we explore Transformer-based and other deep learning models for discrete choice modeling in the travel domain. Our study focuses on an internal Airline Itinerary Dataset (AID), where the goal is to predict which itinerary was booked from a set of alternatives.[1]. We also evaluate our methods on the public Expedia dataset, which contains hotel search sessions and bookings. The prediction task in both datasets is challenging due to the large number of alternatives per assortment (typically more than 30 on average), the high similarity between alternatives, and the prevalence of singletons (i.e., alternatives that appear only once in the entire dataset).

We first optimize a Transformer architecture and design a training procedure. We then demonstrate the transferability of our training procedure to other neural network models and to the Expedia dataset for hotel booking prediction. Our key contributions are:

- We perform an architectural analysis of Transformers for choice modeling and find that, unlike in natural language processing (NLP) tasks, single-head attention offers better performance.
- We confirm the importance of feature engineering, demonstrating that adding session-level features improves predictive accuracy.
- Contrary to the common practice of treating different sessions individually, we show that they can be combined. We introduce mixup to choice modeling and show that combining unrelated choice sessions enhances performance, especially in terms of top-k ranking metric.
- We demonstrate that our training methodology generalizes across model classes and datasets, requiring no additional tuning to yield strong results on the Expedia dataset.

[1]Code and dataset will appear at https://github.com/AmadeusITGroup/ChoiceTransformer

2. Related Work

2.1. Choice models, Machine learning and Recommender Systems

Choice modeling seeks to understand and predict individual decisions among competing alternatives, with applications in transportation [21, 22, 23], travel demand analysis [24, 11], and online marketplaces [25]. A defining characteristic of many real-world choice problems is that each decision instance—i.e., choice session—involves a distinct, session-specific set of alternatives. For example, in airline choice modeling, a session may correspond to a trip request from city A to city B on a given date, with alternatives representing the available itineraries. Machine learning classification techniques can be used to predict the booked itinerary, but the problem differs fundamentally from standard multiclass classification tasks (e.g., image recognition), where the set of classes is fixed and globally defined across all instances. In choice modeling, by contrast, each session has its own context and alternatives, and these alternatives are valid only within that session (e.g., a specific flight itinerary is meaningful only for a given origin, destination, and date). Thus, alternatives cannot be treated as global "classes" in the standard machine learning sense, since they rarely, if ever, reappear across sessions [9, 7].

Choice modeling setting also differs from traditional recommender systems. In recommender systems, predictions are typically informed by user-specific histories—such as past purchases, ratings, or clicks—which enable personalized recommendations [26, 27]. In many choice modeling settings, however, such user data are unavailable or even prohibited (e.g., privacy regulations may prevent tracking passengers' booking histories). As a result, predictions must rely exclusively on alternative/item attributes and the aggregate behavior of users, rather than repeated interactions with the same individual. This makes the problem similar to a "cold-start" setting in recommender systems [27], but with the additional challenge that both users and items are often unique to each session.

While the classical Multinomial Logit (MNL) model has been widely adopted, its limitations—such as the independence of irrelevant alternatives (IIA) assumption and limited capacity to model complex interactions—have motivated researchers to explore more flexible machine learning approaches. In this context, single-choice modeling can be framed as a classification task, where a model assigns probabilities to the different available alternatives. Embedding methods [28] or random forests [29] were applied to improve prediction accuracy and overcome some limitations of the MNL model. Decision trees have also been used to automatically partition customers into hierarchical segments and capture non-linear interactions between features of alternatives and characteristics of the decision maker [16].

More advanced models have been proposed to approximate the structure of random utility models (RUM). RUMnets [30] were designed to approximate arbitrarily closely the class of RUM discrete choice models. Similarly, AssortNet [31] proposes residual neural network architectures to predict choice probabilities for use in assortment optimization problems. In [32], the authors propose a recurrent neural network (RNN) architecture with an attention mechanism that learns to point, within a sequence of alternatives, to the chosen one. PCMC-Net [9] combines representation learning with the Pairwise Choice Markov Chains (PCMC) framework [10] to relax strong assumptions like IIA and obtain a linear system whose solution is the choice distribution.

A commonly raised concern with deep learning-based choice models is their interpretability. While such models often outperform traditional methods in terms of predictive accuracy, their "black-box" nature can obscure the decision logic. However, experiments in [11] demonstrate how the probabilities provided by such a choice model can be interpreted as market shares at equilibrium and be used as demand quantities. Furthermore, the recent release of the open-source library Choice-Learn [33] provides a modular and extensible framework for building and evaluating machine learning-based choice models.

2.2. Transformers as discrete choice models

The Transformer architecture was originally introduced for machine translation [12] and has rapidly evolved as the main neural network architecture of choice not only for natural language processing tasks

(encoding text, text and code generation, etc.), but also in computer vision [34] (image classification, image generation, object detection, etc.), in audio processing [35], and numerous other tasks.

Its application to choice modeling is more recent, but growing. In [36], a Transformer-based architecture is proposed to address different types of choice modeling problems: single choice, sequential choice and multi-choice. Transformer networks were considered especially suitable for this task as they take into account not only the features of the customer and the items, but also the context, which could be the assortment, as well as past choices of the customer.

The Transformer is employed in [37] both as a choice model and as a simulator to generate data to reflect different consumer purchasing behaviours. The simulator generates data which allows to contrast the differences between the baseline MNL and DeepFM models. An attention-based choice model was proposed by [38] and shown to be a low-rank generalization of the Halo Multinomial Logit (Halo-MNL) model. The proposed method was shown empirically to outperform other methods on a hotel dataset and shopping cart dataset.

A comprehensive study of Transformer-based choice modeling is given in [39]. In particular, Transformer-based architectures are compared against many other different models on both the IRI dataset and Expedia Hotels dataset. Notably, their results showed that even compact Transformer models with as few as 2,448 parameters outperformed larger RUMnet models, and that larger Transformer variants further improved performance. This findings indicate that the architecture itself is well suited to choice modeling, and scales well, as observed in other domains.

Based on the literature above, we consider the Transformer to be the state of the art model, upon which major part of our experiments is carried out. In the following sections, we detail feature processing and data augmentation techniques which can further improve the performance of a Transformer model.

3. Datasets and input features

Before describing the datasets, we first clarify some key definitions used in choice modeling. A *session* (or decision instance) refers to a situation where a user makes a choice. The set of available alternatives in that session (e.g., all flight itineraries for a given origin, destination, and date) is called the *assortment*. Each *alternative* is a specific option within the assortment (e.g., a particular flight itinerary) and is characterized by its own alternative features (e.g. price, duration, or airline). The *context*, in contrast, captures session-level attributes that apply to all alternatives within the assortment (e.g. the requested origin, destination, travel date).

3.1. Airline Itinerary Dataset

A passenger name record (PNR) contains relevant data regarding travel bookings, such as flight information of each segment of a journey and information about the individual. Additionally, it has information about ancillary services and special service requests. In order to obtain a full choice set, data from PNRs are matched with search log activity. Those logs shows all available options presented to the traveller prior to booking.

We have created the Airline Itinerary Dataset (AID): a dataset consisting of flight booking sessions on a set of European origins and destinations. Each choice session contains up to 50 different proposed itineraries, one of which has been booked by the customer. There are 815 559 distinct alternatives among which 84% are singletons and 99% are observed at most seven times. In total, there are 33 951 choice sessions. For the experiments in section 4, we split the sessions into 27 160 for training and to 6791 for testing. The dataset has both numerical and categorical features, shown in Table 1. Finally there is the target variable which is binary and indicates whether an alternative was chosen within its session.

Within the set of features of a given session, there are certain elements which are common across all alternatives. We refer to those as the context. Examples of context features are the origin and destination of the trip, or whether the trip is domestic. The rest of the features are specific to each alternative, which we refer to as the alternative features. These include the price of the flight or the

Table 1
Features of the airline itinerary choice dataset.

	Type	Feature	Range/Cardinality
Context	Cat.	Origin-Destination (OD)	98
		Search Office	12
	Num.	Departure weekday	[0, 6]
		Stay Saturday	[0, 1]
		Continental Trip	[0, 1]
		Domestic Trip	[0, 1]
		Days to departure	[0, 343]
Alternative features	Cat.	Airline (of first flight)	64
	Num.	Price	[77.15, 16781.50]
		Stay duration (minutes)	[121, 434000]
		Trip duration (minutes)	[105, 4314]
		Number connections	[2, 6]
		Number airlines	[1, 4]
		Outbound departure time (in s)	[0, 84000]
		Outbound arrival time (in s)	[0, 84000]

number of connections. Note that within an assortment, alternatives may have common features such as the same airline, but at least one alternative feature is always distinct from the other alternatives.

3.2. Expedia Travel Dataset

In this study we use the AID dataset to develop a training procedure and subsequently evaluate its robustness on the Expedia hotel search dataset. The Expedia dataset was released as a part of a competition [40]. It contains hotel search sessions conducted on the Expedia website. It comprises of 399 344 hotel search sessions with a total of 9 917 530 alternatives, representing 136 886 distinct hotels. A single hotel may appear multiple times in a session, offering different room types or prices. Each session includes between 5 and 38 alternatives.

In this dataset the context features include the length of stay, whether the stay includes a Saturday night, the number of adults and children, the number of rooms requested, the date and time of the search, and the booking window (i.e., the time between the search and the stay). Additional context variables are location related indexes representing the geographical location from which the search is done, as well as the location of the accommodation. Alternative features include the current price, historical average price and whether the alternative is a promoted offer. Other features include the hotel's star rating, average customer review scores, and scores reflecting how well-located the hotel is, and whether it belongs to a major hotel chain.

Since our focus is on predicting which alternative was booked, we exclude sessions where no booking occurred. In accordance to the preprocessing used in [39], we also discard features that are only observable at or after the time of search. Specifically, we omit the click indicator and the final transaction price. We retain only alternatives with strictly positive prices below 1000$ and booking windows shorter than one year. For categorical variables, all categories with fewer than 1000 occurrences are grouped together into a single class with a value equal -1 and the missing values are imputed with zeros. After the preprocessing, the final dataset contains 276 211 sessions and 6 909 971 alternatives.

We note that despite following the preprocessing of [39], we obtain a dataset whose size is not exactly the same as reported in the original study, meaning that our results are similar but not directly comparable to [39]. Furthermore, [30] proposed a similar data preprocessing but with the inclusion of a "clicked" related feature. Therefore, their results correspond to a different shopping phase, where there is the knowledge of whether the user interacted with the alternatives. This knowledge is provided as an input feature to the model, simplifying the prediction task and yielding higher top-1 and top-5

accuracies.

3.3. Initial Processing of data features

As shown in table 1, the dataset contains both numerical and categorical variables. Examples of categorical variables include the origin and destination airport names, or booking office which are strings. Since neural networks require numerical input, for each categorical variable we use an embedding layer to transform the variable into a numerical one. The embedding layer takes as input a categorical feature with cardinality c_i (e.g., $c_i = 98$ for airport codes representing origin and destination), and maps it to a dense vector of dimension d_i. Following the rule of thumb proposed in [9], we initially set $d_i := min(\lceil c_i/2 \rceil, 50)$, although later we show that d_i can be greatly decreased. To avoid overfitting, after the embedding stage, dropout is applied on the embedded categorical input. In contrast, numerical variables—such as trip duration or price—are directly given as inputs to the network without transformation.

After embedding the categorical variables, two structures are created: a numerical vector of size $d_{context}$ representing the context of the entire session and a matrix of size $[n, d_{alternative}]$ containing the features for each of the n alternatives. Since the context contains relevant information across all alternatives, we concatenate the context vector with the features of each alternative, resulting in an input matrix of size $[n, d_{data}]$ with $d_{data} = d_{context} + d_{alternative}$. Please note that the embeddings described above are all learned jointly with the model's parameters using standard backpropagation.

In the study, we primarily use a BERT-like Transformer architecture [41]. The model accepts an input matrix of size $[n, d_{model}]$, with n being the number of alternatives per session. Analogously, for Transformers used as Large Language Model (LLM) n would be the number of input tokens and d_{model} the dimensionality of each input embedding. Each alternative is represented by a d_{model}-dimensional feature embedding. To map the data matrix of size $[n, d_{data}]$ to the input size of $[n, d_{model}]$ expected by the Transformer, we use a linear layer followed by a Gelu activation function. The Gelu was added to avoid having two consecutive linear transformations, as the Transformer's multi-head attention module begins with a linear transformation. Hence, this non-linear activation function avoids the composition of two linear layers which is equivalent to a single linear layer.

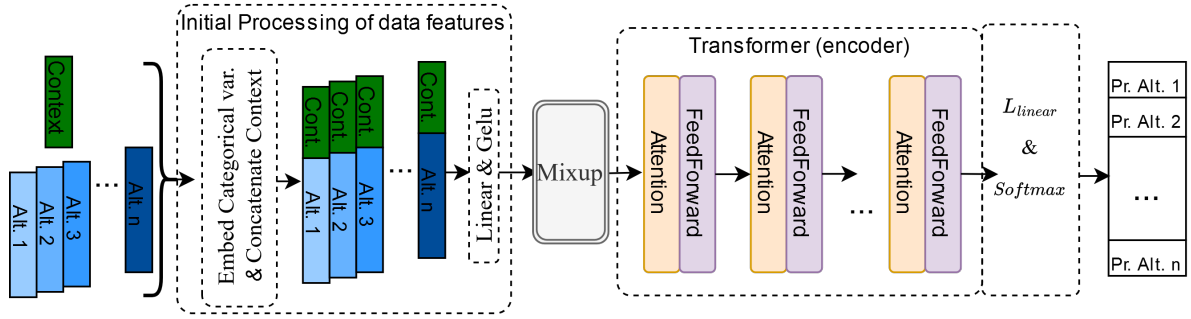

Figure 1: Proposed Transformer pipeline for choice modeling

4. Methodology

Focusing on the AID dataset, we first optimize the Transformer architecture and then design a training procedure to enhance its performance. The overall pipeline is illustrated in Figure 1. The experimental steps are as follows:

Step 1: We begin by optimizing the Transformer architecture, tuning a number of hyperparameters that determine the size of the model. Based on the results of this step, we select an architecture

offering a good trade-off between number of parameters and performance. We name that model *ChoiceTransformer*.

Step 2: With the ChoiceTransformer architecture fixed, we proceed into investigating the processing of the input features. We show that the embedding dimensions for categorical inputs can be significantly reduced without sacrificing accuracy. Additionally, we improve the performance by introducing new features, defined as ratios comparing numerical feature values of each alternative to the session's best corresponding feature value.

Step 3: Finally, we test a data augmentation strategy using the best feature setup from Step 2 and the ChoiceTransformer architecture. Specifically, we introduce mixup [20] which applies a convex combination of the inputs of pairs of sessions. We show this method leads to further gains in accuracy. To the best of our knowledge, this is the first application of mixup in the context of choice modeling.

Steps 2 and 3 were optimized using the ChoiceTransformer as the model. However, our goal is to provide a general recipe for feature processing and augmentation that is effective across architectures and datasets. Therefore in Section 5 we test the proposed recipe using a larger Transformer variant (ChoiceTransformer-L), a different class of neural network models (PCMC-net [9]), and a another dataset (Expedia).

4.1. Step 1: Optimizing Transformer architecture

The Transformer architecture [12] was initially conceived for natural language processing, but since then it has become extremely popular and tested in many applications of machine learning. The encoder Transformer architecture can be seen as a function $T : \mathbb{R}^{n \times d_{model}} \rightarrow \mathbb{R}^{n \times d_{model}}$ generating high-level representations per alternative. Intrinsically, the Transformer architecture is permutation equivariant meaning that if a random permutation σ of the rows of the input is applied, the output will present the same permutation, i.e. $T(\sigma(x)) = \sigma(T(x))$. In natural language, the order of the words/tokens is important, so to break this property, positional embeddings are added. In contrast, the choice modeling problem is mainly a permutation equivariant problem, meaning that changing the order of the alternatives should generally not impact the user's preference ranking of the alternatives. Therefore, similarly to previous works [39, 36, 37] we avoid the use of positional embeddings which deteriorates the performance in choice modeling tasks.

The architecture is a stack of N layers where each layer is comprised of a standard multi-head attention part and a feed-forward network. The multi-head attention has h heads. As in [12] each head has key, queries and values of dimensions $\frac{d_{model}}{h}$ per alternative (in the NLP domain it is per token) and so the total number of parameters in the model is independent of the number of heads h. We use a BERT-like architecture [41] where the feed-forward network has a standard form consisting of layer normalization [42] followed by a linear layer expanding four-fold the dimensions (i.e $d_{hidden} = 4d_{model}$), then a Gelu activation function and finally a second linear layer bringing the dimensions back to d_{model}. Finally, as shown in Figure 1, we apply a linear transformation per alternative $L_{linear} : \mathbb{R}^{n \times d_{model}} \rightarrow \mathbb{R}^{n \times 1}$ on the output of the Transformer model and a softmax function providing the probability with which each alternative might be selected. Note that the linear layer L_{linear} preserves also the permutation equivariance since it is applied identically to the features of each alternative.

In table 2 a grid search is performed over different architecture hyperparameters. We vary the number of layers $N \in \{1, 2, 4\}$, the number of heads $h \in \{1, 2, 4\}$ and embedding dimensions $d_{model} \in \{16, 32, 64\}$. We split our dataset into 80% training and 20% test set and run each configuration with 6 different seeds and report the mean test performance over those runs and the standard deviation (std). The optimization algorithm is AdamW [43] with the default pytorch hyperparameters (i.e. learning rate 0.001, weight decay 0.01, and betas = (0.9, 0.999)) and batch size equal to 32. The reported number of parameters does not include the parameters used in the processing of the input data, which in total are for $d_{model} = \{16, 32, 64\}$ equal to $\{8380, 9964, 13132\}$ respectively.

Table 2 presents the top 12 configurations. Surprisingly, we note that, in general, the lower number of heads the better, with an average accuracy over all configurations for $h = 1$ being 29.10%, with $h = 2$ being 29.03% and with $h = 4$ being 28.91%. Also there is the trend of configurations with more parameters yielding better results, but the difference may be marginal. The best configuration is (4,1,64), and it has the most parameters. However, the configuration (2,1,32) has around 8 times less parameters and only 0.04% lower accuracy. For this reason, we adopt it as our base model, naming it *ChoiceTransformer*. The larger configuration is retained for testing the robustness of our proposed method and referred to as *ChoiceTransformer-L* in Section 5.

Table 2
Impact of Transformer hyperparameter scaling with respect to the number of parameters and Accuracy

Layers N	Heads h	d_{model}	Transformer Parameters	Mean Top-1 Accuracy	Std Top-1 Accuracy
4	1	64	199361	**29.35**%	0.53
2	1	64	99777	29.32%	0.23
2	1	32	25313	29.31%	0.28
4	2	32	50529	29.24%	0.26
1	1	32	12705	29.22%	0.34
4	4	64	199361	29.15%	0.31
4	2	64	199361	29.12%	0.21
2	2	64	99777	29.08%	0.17
2	2	32	25313	29.06%	0.36
2	2	16	6513	29.05%	0.50
1	1	16	3281	29.02%	0.44
4	2	16	12977	29.02%	0.29

4.2. Step 2: Feature engineering and training strategies

Here, we fix the architecture of the model to ChoiceTransformer with configuration (2,1,32) and proceed to optimizing the processing of the input features. Using the implementation of [9] for transforming the categorical variables to numerical vectors, we note that the final dimensions of the input for the AID dataset are $d_{data} = 98$ and in total $d_{categ} = 86$ elements are devoted to represent the categorical variables. We first reduce d_{categ} by changing the function used to define the embedding dimensions for each categorical variable from $d_i := min(\lceil c_i/2 \rceil, 50)$ to $d_i := min(\lceil c_i/10 \rceil, 4)$. This results in $d_{data} = 22$ and $d_{categ} = 10$. Dropout with a probability $p_{drop} = 0.5$ was also used in [9] to avoid overfitting, however it is reasonable, since d_{categ} is reduced, to also reduce the regularization applied on those embeddings. In Table 3, decreasing the probability to $p_{drop} = 0.4$ seems optimal and so we adopt it for the next results. We emphasize that without any loss in performance, the reduction of d_{data} reduced the total number of parameters of ChoiceTransformer (including the ones in the processing of the data) by 25% (i.e from 35 277 to 26 713).

Table 3
Top-1 accuracy, mean and standard deviation across 6 runs, as a function of dropout rate.

Dropout rate	Mean Accuracy	Std. Accuracy
0.1	28.98	0.37
0.3	29.32	0.22
0.4	**29.39**	0.11
0.5	29.27	0.24

Choice models tasks based on machine learning models can benefit from some engineered features that give a ratio or comparison between a numerical feature of each alternative and the rest of all alternatives in the session [16]. In this work, we incorporate such comparative information by introducing the

following ratio for the applicable numerical features:

$$\frac{\text{alternative's feature value}}{\text{best feature value across session}}. \tag{1}$$

This transformation is applied to features that are directly comparable across alternatives, specifically: price, trip duration, and stay duration. These new ratio features allow each alternative to be contextualized relative to others in the same session. For instance, the price ratio quantifies how much more expensive an option is compared to the cheapest one.

The inclusion of these three features improves top-1 accuracy from 29.39% (std = 0.11) to 29.57% (std = 0.35).

4.3. Step 3: Data augmentation - mixup

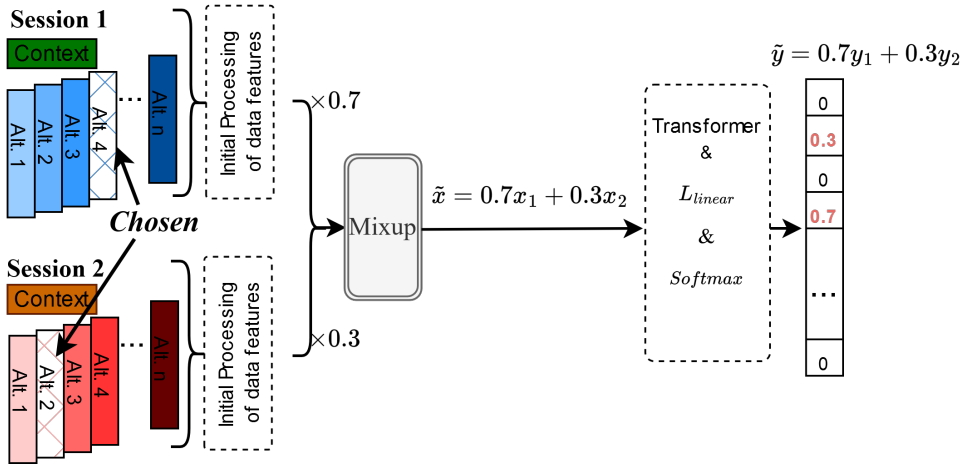

Figure 2: Mixup example visualisation for choice modeling

With the previous steps, the performance of the choice model has improved and it is not trivial to improve it further. Here, we introduce mixup [20] which is a well established data augmentation technique in image classification tasks. Mixup combines two different samples, to synthetically generate a new sample. Specifically, it linearly interpolates a pair of samples and their corresponding labels. Mathematically, given two samples (x_i, y_i) and (x_j, y_j), where x represents the input to the model and y the label in one-hot encoding format, the mixup does:

$$\begin{aligned} \tilde{x} &= \lambda x_i + (1-\lambda)x_j \\ \tilde{y} &= \lambda y_i + (1-\lambda)y_j \end{aligned} \tag{2}$$

with $\lambda \in [0, 1]$ being a mixing coefficient sampled from a Beta distribution:

$$\lambda \sim \text{Beta}(\alpha, \alpha)$$

where $\alpha > 0$ is a hyperparameter controlling the strength of interpolation. As α approaches 0 the effect mixup has on the training procedure vanishes. With $\alpha = 1$ the distribution coincides to a uniform one. As $\alpha \to \infty$ it converges to a Dirac distribution located at 0.5. As shown in Figure 2, we apply mixup on the input of the Transformer model.

There are multiple reasons why mixup can lead to overall performance improvements. In general, mixup has been shown to act as a regularization technique [44] preventing overfitting. The model has to distinguish two inputs that have been linearly combined, making it difficult for the model to memorize those inputs. Thus, it reduces the memorization effect [20] which is a source of deterioration of the generalization performance of neural networks.

In a choice modeling task, applying mixup may seem rather unconventional. Our intuition for testing this technique is that mixup encourages the model to predict softer output distributions. Without mixup the model is trained to always predict the single alternative that was the preferable in that session. However, user preferences are inherently stochastic: two users with identical contexts may make different choices due to personal constraints or preferences. For example, two users might both want to travel the same day from the same departure airport to the same destination one, but end up choosing different itineraries due to different personal time schedule constraints.

Using mixup the model learns to assign some probability to more than one alternative. Therefore it is reasonable to believe that mixup will improve not just the top-1 accuracy, but especially the top-k accuracy with $k > 1$. In Table 4, we show that our intuition is confirmed and mixup improves in general the top-k accuracy. This is important for choice models since it identifies a subset of the many available alternatives that might be appealing to a user. For the AID dataset, this means presenting to the user a desirable set of itineraries for traveling to their destination at the top of their shopping list.

There are some important implementation details to note. There is the possibility that 2 sessions have unequal number of alternatives which will result in $x_i \in \mathbb{R}^{n_i \times d_{model}}$ and $x_j \in \mathbb{R}^{n_j \times d_{model}}$ with $n_i \neq n_j$. Assuming $n_i < n_j$, x_i is padded with zeros to match the dimensions of x_j. Up until now the ordering of the alternatives in the data did not matter due to permutation equivariance. However, when mixup is used, some precautions are needed. In the dataset, the alternatives are sorted according to their price, with the cheapest alternative being the first in the list. This results in the label pointing to the first alternative in most of sessions, as users tend to book the least expensive alternative. Therefore, when picking random pairs to combine, it becomes very likely that $y_j = y_i = (1, 0, 0, ..., 0)$ and so $\tilde{y} = y_i = y_j$. To avoid these scenarios and ensure mixup functions as desired, we permute randomly the alternatives within each session. Finally, using a data augmentation technique makes the training harder and the model needs more iterations to train, as such the number of epochs is increased from 30 to 100.

Table 4
Performance Metrics with Different Mixup Values

Mixup	Top-1 acc.	Top-3 acc.	Top-5 acc.
No Mixup	29.57 (0.35)	56.8 (0.28)	70.38 (0.22)
$a = 0.6$	29.80 (0.32)	57.09 (0.29)	70.87 (0.15)
$a = 0.8$	29.67 (0.36)	**57.4** (0.57)	**70.96** (0.33)
$a = 1.0$	**30.03** (0.27)	57.1 (0.51)	70.75 (0.45)
$a = 1.2$	29.89 (0.31)	57.03 (0.44)	70.92 (0.23)

5. Effective strategy on various models and dataset

In Step 1, we optimized the Transformer architecture. In the subsequent steps, we fixed the architecture to the ChoiceTransformer and focused on enhancing the overall training methodology of the choice model. Therefore, a natural question arises: Do the improvements introduced in Steps 2 and 3-namely, lowering the embedding dimensions, incorporating engineered features, and applying mixup—generalize to other model architectures beyond the original ChoiceTransformer and accross to other datasets?

To investigate this, we apply the training strategy developed in Steps 2 and 3 to both a larger Transformer model, ChoiceTransformer-L, and to a different class of neural network architecture, PCMC-net [9], using the AID dataset. The PCMC-net has a hyperparameter "nodes per layer" which is set to 128, resulting in a model with 54 128 parameters. For each model and training configuration, we perform a 5-fold cross-validation. Each fold uses 70% of the data for training, 20% for testing, and the remaining 10% for validation. The validation set is used to monitor performance per epoch. We select for testing the model checkpoint with the highest top-5 accuracy on the validation set. The maximum number of epochs is set to 100, with early stopping triggered if no improvement in validation accuracy is observed for over 10 consecutive epochs.

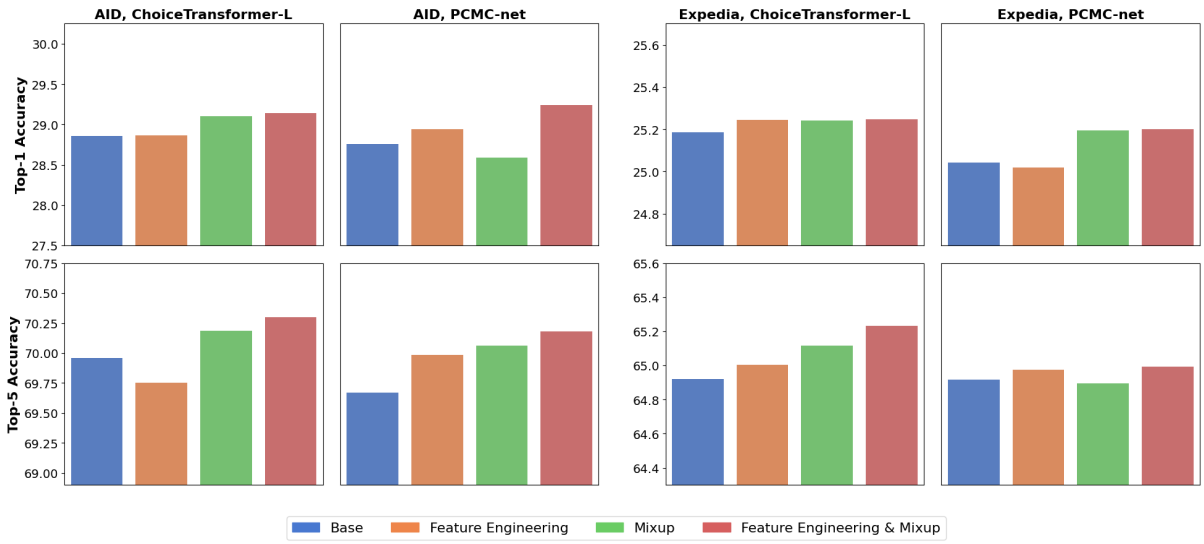

Figure 3: Top-1 and Top-5 accuracies for Transformer and PCMC-net models across AID and Expedia datasets.

Figure 3 shows the effect of our proposed training methodology to the top-1 and top-5 accuracy for both ChoiceTransformer-L and PCMC-net. The "Base" bar corresponds to training the model without any of our enhancements. The "Feature Engineering" bar represents the impact of incorporating Step 2 only. The "Mixup" bar shows the effect of applying the mixup strategy alone (i.e. only applying Step 3). Finally, the "Feature Engineering & Mixup" bar represents the full application of our proposed training method, combining both steps.

To further evaluate the robustness of our method, we also apply it to a different dataset, Expedia hotels. We keep the training procedure identical to the one used for AID, with the sole exception of increasing the batch size from 32 to 256 to accommodate the larger dataset, which contains approximately 8 times more sessions. For the Expedia dataset, the engineered ratio features (see Equation 1) are constructed from the following features: price, number of stars, review score, and the two location-based scores.

We emphasize that throughout these experiments, we maintain the same hyperparameters found to perform best in Sections 4.2 and 4.3: a dropout rate of 0.4, categorical embedding dimension defined by $min(\lceil c_i/10 \rceil, 4)$, and a mixup coefficient α of 0.8. As shown in Figure 3, our proposed training methodology improves the performance of both models across both datasets, without requiring any additional hyperparameter tuning. We remark that applying only mixup already improves top-5 accuracy in most cases, except for PCMC-net on the Expedia dataset. Most importantly, the full method not only gives the best top-5 accuracy, but also consistently maintains or improves top-1 accuracy, demonstrating its robustness and broad applicability.

McNemar's test can be used to evaluate the statistical significance of the difference in performance between two deep learning models when using larger datasets [45]. A two-sided McNemar's test was performed for each model and dataset evaluating the difference between the base configuration and the optimised process with feature engineering and mixup. These results are shown in Table 5, with bold lettering indicating significance at $p < 0.01$ threshold.

Table 5
p-values of McNemar's test comparing base performance of models to optimised version (i.e. with feature engineering and mixup) over two datasets

	Airline dataset	Expedia dataset
PCMC-net Top-1	4.6×10^{-2}	1.1×10^{-1}
PCMC-net Top-5	1.2×10^{-1}	4.2×10^{-1}
Transformer Top-1	$\mathbf{1.2 \times 10^{-3}}$	$\mathbf{2.7 \times 10^{-5}}$
Transformer Top-5	$\mathbf{1.4 \times 10^{-3}}$	$\mathbf{2.7 \times 10^{-5}}$

6. Discussion

Neural networks and indeed Transformers have shown state of the art performance for choice modeling in recent studies. In this work, we experiment with different architectures, feature engineering, and data augmentation techniques in order to further improve the performance of a standard Transformer.

From the results obtained in this study, we can make some interesting conclusions. Transformers perform with high accuracy compared to other methods, as already established in the literature. We observe that relatively small Transformers achieve good performance and may be preferable to larger architectures, as classification accuracy gains may be marginal when scaling the model. In this study, we find a model of size 25313 parameters to be competitive with a model 8 times larger. This reflects a similar outcome from [39], where the smallest Transformer with only 2448 parameters outperformed all other non-Transformer methods in the comparison on the Expedia hotels dataset. We also note that we found single-headed attention to outperform multi-headed attention for the vast majority of configurations of the Transformer implemented in our study.

Furthermore, we find that feature representation in the embedding stage can be compressed further than established in previous works. By representing categorical features with a maximum of 4 dimensions, versus 50 using the encoding of [9], we reduce the overall number of parameters from 35277 to 26713 for ChoiceTransformer while maintaining the classification accuracy.

Similarly to previous studies using machine learning models, we find that explicitly providing features engineered over the assortment of alternatives provides benefits to the model. These types of features can be beneficial for two reasons. In some cases, as is well known for the MNL model, the utility of each alternative is computed independently of the features of other alternatives. In such cases, these features provide valuable contrastive information to the model that would not be available to it otherwise. More powerful non-linear models may be able to do such comparisons between alternatives, but the mechanisms involved might be limited, such as pairwise comparisons, or simply require significant computational cost in order to approximate complex decision rules. In this study, such aggregated features allow the model to exploit the information provided by the features with simpler rules, and thus the model does not need to allocate a larger number of parameters to replicate simple decision processes. As the old adage goes "don't learn what you already know".

A novel contribution of this work is in the exploration of data augmentation techniques from different domains. In particular, it is shown that mixup, which operates over multiple choice sessions, can be used during training of the ChoiceTransformer in order to improve its testing accuracy. This result is significant as we believe this is the first implementation of mixup in the field of choice modeling.

The overall performance of discrete choice models in these real life industrial contexts may appear somewhat low, specifically achieving under 30% Top-1 accuracy. It is important to remember that the model is comparing a large number of alternatives per assortment (typically over 30), that contain a large number of singletons, there is high similarity between many alternatives, and limited context with no-personalisation is provided to the model. Taking an example from the AID dataset, there may be 2 alternatives that differ only by marketing carrier, or alternatives that differ only by a minor time difference for one flight segment.

The improvement in performance from the base models to the augmented ones, with feature engineering and mixup, is *typically less than 1%* for top-1 and top-5 accuracies. It is important to consider the context of choice modeling where such small increases in performance are expected even from major model improvements, e.g. similar gains were observed from RUMnet to Transformers [39]. *Even small gains can translate into significant business impact in real-world applications such as travel recommendation systems that are called with 100,000 transactions per second.* The top-1 and top-5 accuracies represent our ability to show customers the most appealing offers at the top of the list, leading to higher conversion.

Finally, while many of the experiments in this study relate to improving the performance of a Transformer based choice model, we show that the proposed pipeline of data embedding, feature engineering and mixup can benefit other neural networks including previous state of the art models such as the PCMC-net, although we did not find the improvements to be as significant. Furthermore, we

show that the selection of hyperparameters of the feature processing stage, such as embedding dimension size and mixup weight, produce better performance for both larger and smaller implementations of the Transformer. This strategy was then applied to a different dataset with no changes in hyperparameters, where it was confirmed that the proposed pipeline leads to better results across both accuracy metrics, for both models.

7. Conclusion

In many applications of machine learning, there is a clear trend toward the dominance of attention-based neural networks, particularly the Transformer architecture. While previous studies have demonstrated that Transformers are indeed a powerful tool for choice modeling, the results of this study highlight that improved feature representation and novel training procedures can further enhance performance. Future research could explore alternative methods for feature processing and data augmentation. Another promising direction is the development of more interpretable methods to better understand the internal decision-making logic of Transformers and to revisit one of the original goals of discrete choice modeling: gaining deeper insight into the factors that drive individual decisions.

8. Ethical Considerations

This study investigates the potential of using Transformers for choice modeling. For the experimental section, we utilized known datasets which have been used previously in the domain by multiple groups and are publicly accessible. The authors commit to releasing the code upon request should this article be published.

9. Declaration on Generative AI

Generative AI tool (ChatGPT) was used for minor text editing: paraphrasing, rewording and sentence polishing. The authors are fully responsible for the scientific content of the paper.

References

[1] D. McFadden, The measurement of urban travel demand, Journal of public economics 3 (1974) 303–328.

[2] A. K. Strauss, R. Klein, C. Steinhardt, A review of choice-based revenue management: Theory and methods, European Journal of Operational Research 271 (2018) 375–387.

[3] L. A. Garrow, Discrete choice modelling and air travel demand: theory and applications, Routledge, 2016.

[4] L. Bernardi, T. Mavridis, P. Estevez, 150 successful machine learning models: 6 lessons learned at booking. com, in: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, 2019, pp. 1743–1751.

[5] R. D. Luce, Individual Choice Behavior: A Theoretical analysis, Wiley, New York, NY, USA, 1959.

[6] J. Marschak, Binary Choice Constraints on Random Utility Indicators, Cowles Foundation Discussion Papers 74, Cowles Foundation for Research in Economics, Yale University, 1959.

[7] K. E. Train, Discrete choice methods with simulation, Cambridge university press, 2009.

[8] M. Haldar, M. Abdool, P. Ramanathan, T. Xu, S. Yang, H. Duan, Q. Zhang, N. Barrow-Williams, B. C. Turnbull, B. M. Collins, et al., Applying deep learning to airbnb search, in: proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & Data Mining, 2019, pp. 1927–1935.

[9] A. Lhéritier, PCMC-Net: Feature-based pairwise choice markov chains, in: International Conference on Learning Representations, 2020.

[10] S. Ragain, J. Ugander, Pairwise choice markov chains, in: Advances in Neural Information Processing Systems, 2016, pp. 3198–3206.
[11] R. Acuna-Agost, E. Thomas, A. Lhéritier, Price elasticity estimation for deep learning-based choice models: an application to air itinerary choices, Journal of Revenue and Pricing Management 20 (2021) 213–226.
[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in neural information processing systems, 2017, pp. 5998–6008.
[13] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu, Exploring the limits of transfer learning with a unified text-to-text transformer, Journal of machine learning research 21 (2020) 1–67.
[14] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al., An image is worth 16x16 words: Transformers for image recognition at scale, arXiv preprint arXiv:2010.11929 (2020).
[15] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, et al., On the opportunities and risks of foundation models, arXiv preprint arXiv:2108.07258 (2021).
[16] A. Lhéritier, M. Bocamazo, T. Delahaye, R. Acuna-Agost, Airline itinerary choice modeling using machine learning, Journal of Choice Modelling 31 (2019) 198 – 209.
[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, The journal of machine learning research 15 (2014) 1929–1958.
[18] A. Krogh, J. Hertz, A simple weight decay can improve generalization, Advances in neural information processing systems 4 (1991).
[19] A. Parsi, M. Jafari, S. Sabzekar, Z. Amini, Improving trip mode choice modeling using ensemble synthesizer (ensy), arXiv preprint arXiv:2407.01769 (2024).
[20] H. Zhang, mixup: Beyond empirical risk minimization, arXiv preprint arXiv:1710.09412 (2017).
[21] V. Dahmen, S. Weikl, K. Bogenberger, Interpretable machine learning for mode choice modeling on tracking-based revealed preference data, Transportation Research Record 2678 (2024) 2075–2091.
[22] F. El Zarwi, A. Vij, J. L. Walker, A discrete choice framework for modeling and forecasting the adoption and diffusion of new transportation services, Transportation Research Part C: Emerging Technologies 79 (2017) 207–223.
[23] S. F. Varotto, R. Krueger, M. Bierlaire, Modelling travel behaviour: a choice modelling perspective, in: Handbook of Travel Behaviour, Edward Elgar Publishing, 2024, pp. 118–139.
[24] L. Ding, N. Zhang, A travel mode choice model using individual grouping based on cluster analysis, Procedia engineering 137 (2016) 786–795.
[25] M. Danaf, F. Becker, X. Song, B. Atasoy, M. Ben-Akiva, Online discrete choice models: Applications in personalized recommendations, Decision Support Systems 119 (2019) 35–45.
[26] F. Ricci, L. Rokach, B. Shapira, Introduction to recommender systems handbook, in: Recommender systems handbook, Springer, 2010, pp. 1–35.
[27] A. I. Schein, A. Popescul, L. H. Ungar, D. M. Pennock, Methods and metrics for cold-start recommendations, in: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, 2002, pp. 253–260.
[28] A. Seshadri, A. Peysakhovich, J. Ugander, Discovering context effects from raw choice data, in: International conference on machine learning, PMLR, 2019, pp. 5660–5669.
[29] L. Cheng, X. Chen, J. De Vos, X. Lai, F. Witlox, Applying a random forest method approach to model travel mode choice behavior, Travel behaviour and society 14 (2019) 1–10.
[30] A. Aouad, A. Désir, Representing random utility choice models with neural networks, arXiv preprint arXiv:2207.12877 (2022).
[31] H. Wang, Z. Cai, X. Li, K. Talluri, A neural network based choice model for assortment optimization, arXiv preprint arXiv:2308.05617 (2023).
[32] A. Mottini, R. Acuna-Agost, Deep choice model using pointer networks for airline itinerary

prediction, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017, pp. 1575–1583.
[33] V. Auriau, A. Aouad, A. Désir, E. Malherbe, Choice-learn: large-scale choice modeling for operational contexts through the lens of machine learning, Journal of Open Source Software 9 (2024) 6899.
[34] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu, et al., A survey on vision transformer, IEEE transactions on pattern analysis and machine intelligence 45 (2022) 87–110.
[35] L. Dong, S. Xu, B. Xu, Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition, in: 2018 IEEE international conference on acoustics, speech and signal processing (ICASSP), IEEE, 2018, pp. 5884–5888.
[36] H. Wang, X. Li, K. Talluri, Transformer choice net: A transformer neural network for choice prediction, arXiv preprint arXiv:2310.08716 (2023).
[37] Z. Peng, Y. Rong, T. Zhu, Transformer-based choice model: A tool for assortment optimization evaluation, Naval Research Logistics (NRL) (2024).
[38] J. Ko, A. A. Li, Modeling choice via self-attention, arXiv preprint arXiv:2311.07607 (2023).
[39] Q. Jiang, Choice Modeling and Assortment Optimization on the Transformer Model, Ph.D. thesis, Massachusetts Institute of Technology, 2024.
[40] Adam, B. Hamner, D. Friedman, SSA_Expedia, Personalize expedia hotel searches - icdm 2013, https://kaggle.com/competitions/expedia-personalized-sort, 2013. Kaggle.
[41] J. Devlin, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).
[42] J. L. Ba, Layer normalization, arXiv preprint arXiv:1607.06450 (2016).
[43] I. Loshchilov, Decoupled weight decay regularization, arXiv preprint arXiv:1711.05101 (2017).
[44] L. Carratino, M. Cissé, R. Jenatton, J.-P. Vert, On mixup regularization, Journal of Machine Learning Research 23 (2022) 1–31.
[45] T. G. Dietterich, Approximate statistical tests for comparing supervised classification learning algorithms, Neural computation 10 (1998) 1895–1923.