Probabilistic Hash Embeddings for Online Learning of Categorical Features

Aodong Li, Abishek Sankararaman, Balakrishnan Murali Narayanaswamy

Amazon Web Services
aodongli@amazon.com, abisanka@amazon.com, muralibn@amazon.com

Abstract

We study streaming data with categorical features where the vocabulary of categorical feature values is changing and can even grow unboundedly over time. Feature hashing is commonly used as a pre-processing step to map these categorical values into a feature space of fixed size before learning their embeddings. While these methods have been developed and evaluated for offline or batch settings, in this paper we consider online settings. We show that deterministic embeddings are sensitive to the arrival order of categories and suffer from forgetting in online learning, leading to performance deterioration. To mitigate this issue, we propose a probabilistic hash embedding (PHE) model that treats hash embeddings as stochastic and applies Bayesian online learning to learn incrementally from data. Based on the structure of PHE, we derive a scalable inference algorithm to learn model parameters and infer/update the posteriors of hash embeddings and other latent variables. Our algorithm (i) can handle an evolving vocabulary of categorical items, (ii) is adaptive to new items without forgetting old items, (iii) is implementable with a bounded set of parameters that does not grow with the number of distinct observed values on the stream, and (iv) is invariant to the item arrival order. Experiments in classification, sequence modeling, and recommendation systems in online learning setups demonstrate the superior performance of PHE while maintaining high memory efficiency (consumes as low as 2$\sim$4% memory of a one-hot embedding table).

Code — https://github.com/aodongli/probabilistic-hash-embeddings

Extended version — https://arxiv.org/abs/2511.20893

1 Introduction

Categorical features occur in many high-value ML applications: finance (Clements et al. 2020), fraud detection (Al-Hashedi and Magalingam 2021), anomaly detection (Han et al. 2022), cybersecurity (Sarker et al. 2020), medical diagnosis (Shehab et al. 2022), recommendation systems (Ko et al. 2022; Lai et al. 2023) etc. Large vocabularies and embedding tables are strong characteristics of these categorical feature-intensive applications. While assigning each item[1] its own row in a large embedding table typically improves accuracy, this comes at a cost. Larger embedding tables require more resources and memory to deploy and slow down execution.

A common and well-established solution for learning categorical features at scale, without maintaining a large vocabulary and embedding table, is the use of hashing techniques (Weinberger et al. 2009; Tito Svenstrup, Hansen, and Winther 2017; Shi et al. 2020b; Kang et al. 2021; Lai et al. 2023; Coleman et al. 2024). For any categorical value (typically a string), a hash function maps it to an index in a small, fixed range. This index addresses a row in a much smaller embedding table. These rows, referred to as *hash embeddings*, are used in subsequent model training and inference.

Hash collisions–when different items map to the same hashed value and thus share an embedding–can degrade model performance. This issue can be mitigated in two ways: first, by designing sophisticated operations on the hashed values (Weinberger et al. 2009); second, by using multiple hash functions (Tito Svenstrup, Hansen, and Winther 2017; Coleman et al. 2024). Large technology firms like Yahoo and Google have successfully incorporated this approach in their applications (Weinberger et al. 2009; Coleman et al. 2024).

A major limitation of prior works is they focused on *offline* settings where data distributions are stationary and the entire test set is available in batch. In other words, the vocabulary of categorical items is fixed with no new or out-of-vocabulary items occurring during testing.

In many real applications, data arrives in a streaming fashion: *(i)* the vocabulary of categorical items can change, and/or *(ii)* the meaning of an item can evolve. These characteristics present challenges to offline predictive models. Failure to adapt to the expanding vocabulary leads to a loss in predictive performance, as shown in Fig. 1–in streaming binary classifications, new groups or sets of categorical items occur sequentially, and modeling these new items incrementally leads to significant accuracy improvement. This problem of vocabulary expansion is fairly common in practice: new products are added to grocery stores (Cheng et al. 2023), new usernames and application names appear in intrusion detection systems (Siadati and Memon 2017; Le et al. 2022), new patients arrive at hospitals, and so on.

More critically, as our analysis and experiments show (Secs. 3.4 and 4), naively adapting to new items while using hashing techniques suffers from *catastrophic forgetting* (Kirkpatrick et al. 2017). In hash embeddings, different items

[1]We use the term *item* to refer to a categorical value.

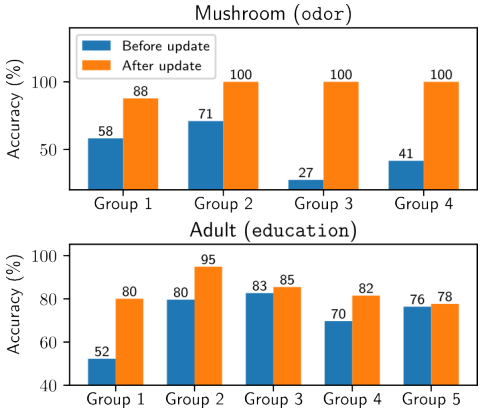

Figure 1: On two tabular datasets, Mushroom and Adult, we split the data into groups based on random partitions of categorical column vocabularies, such that each group has disjoint vocabulary sets. We report the results before and after online learning on each group. The performance gaps motivate the need to learn representations of new items. Bracketed values indicate the splitting columns. Results are averaged over five runs. Partition details are in Fig. 9.

may share embeddings, updating one can adversely affect another, causing the model to effectively "forget." Furthermore, this forgetting can be exacerbated depending on the arrival order. Consequently, hash embeddings are not yet fit for online learning in their vanilla form.

In this paper, we employ Bayesian online learning to mitigate the forgetting issue. This approach is theoretically shown to be as effective as offline batch learning, regardless of the order in which items arrive (Opper and Winther 1999). Specifically, we model hash embeddings as random variables and perform posterior inference upon observing new data. This probabilistic treatment enables continual adaptation to new items while preserving knowledge of previously seen ones.

Main Contributions: Our work proposes *probabilistic hash embeddings* (PHE) with Bayesian updates to handle dynamic vocabularies effectively and efficiently. The intuition behind PHE stems from its benefits in *(i)* efficiency, as memory and the number of model parameters is bounded, with only a small number of parameters requiring online updates (ie., less forgetting), and *(ii)* accuracy, as the Bayesian treatment provides implicit regularization that balances forgetting and adaptation while maintaining invariance to item arrival order, without dataset-specific regularization design.

We highlight PHE as a plug-in module, which can be applied to other probabilistic models such as Deep Kalman Filters (Krishnan, Shalit, and Sontag 2015) and Neural Collaborative Filtering (He et al. 2017). The usage of PHE allows these models to handle unbounded items in their domains in a principled way. We derive scalable variational inference algorithms for learning PHE and provide theoretical analysis demonstrating PHE's superiority for online learning. Empirically, our method outperforms baselines in three setups: online supervised learning with new items, conditional sequence modeling with growing vocabularies, and a recommendation system with evolving user-item interactions.

Organization: We survey related work in Sec. 2, present PHE, derive its inference algorithms in Sec. 3, demonstrate PHE's efficacy in Sec. 4, and conclude in Sec. 5. More details and limitation discussions are in supplementary materials.

2 Related Work

Hashing trick. Weinberger et al. (2009) first proposed using hashing to handle an unbounded number of categorical items. To mitigate degradation due to hash collisions, Serrà and Karatzoglou (2017) used bloom filters. More recently, Tito Svenstrup, Hansen, and Winther (2017); Cheng et al. (2023); Coleman et al. (2024) proposes shared embeddings across all categorical features for efficiency and multiple hashing functions to reduce collisions. However, unlike our method, these approaches are deterministic and are developed in offline settings. As shown in our experiments, deterministic hashing embeddings are vulnerable to evolving vocabularies.

Continual learning. Previous continual learning methods (Kirkpatrick et al. 2017; Nguyen et al. 2018; Lopez-Paz and Ranzato 2017) were designed for continuous-valued features and do not address how to learn evolving categorical features. This only existing approach, architecture-expanding methods that dynamically expand the embedding tables (see EE in experiments), leads to unbounded memory usage (Rusu et al. 2016; Yoon et al. 2017; Jerfel et al. 2019). In contrast, PHE is the first to maintain constant memory while preserving accuracy and invariance to category arrival order.

Temporal and recommendation models. Temporal and recommendation models are important applications for categorical feature embeddings. We extend Deep Kalman Filters (DKF) (Krishnan, Shalit, and Sontag 2015) to handle evolving vocabularies, whereas the original DKF assumes a fixed vocabulary. Similarly, previous recommendation methods (Ko et al. 2022; Shi et al. 2020b; Kang et al. 2021; Coleman et al. 2024) assume training data is given in batch and the item vocabulary remains fixed.

Tabular data models. Categorical features are ubiquitous in tabular datasets. In online and continual learning settings, deep learning-based methods have been studied in recent years (Huang et al. 2020; Du et al. 2021; Liu, Di, and Chen 2023). However, all of these works assume the vocabulary of categorical items is known and fixed in advance. *Ours is the first online learning method for tabular data that can handle increasing and unbounded items.* While Kim, Grinsztajn, and Varoquaux (2024) use string embeddings from language models for open-vocabulary categorical features in an offline setting, we focus on online settings. Moreover, language model representations cannot be exploited when strings lack semantic meaning (such as anonymized ZIP codes, IP addresses, etc.). We survey additional related work in Sec. D.

3 Methodology

We first establish the necessary notations and introduce our proposed probabilistic hash embedding (PHE) module. Next, we derive an online inference algorithm for PHE. Finally, we analyze and explain why PHE is superior in online learning.

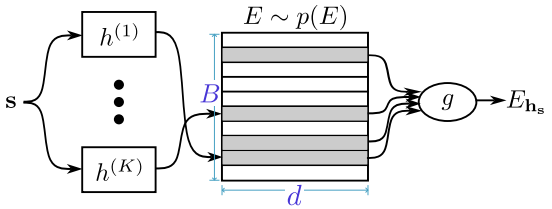

Figure 2: PHE architecture. The input $\mathbf{s}$ can be usernames or anonymized strings. The entire module represents $p(E_{\mathbf{h_s}})$.

3.1 Notations and Problem Setup

We consider the problem of learning predictive ML models with categorical features in an online learning environment. We assume the model input and output dimensions are fixed but new categorical features may occur over time.[2]

We denote categorical items by $s \in \mathcal{S}$. Typically, practitioners aim to predict certain target features using ML models based on other input features. We denote these target features by y. Depending on the task, y can be either numerical or categorical. We use subscript i to denote the i-th data point.

Hashing techniques use hash functions (e.g., MD5) to map categorical items to hashed values. These hashed values are then used to index rows of an embedding table. The range of hash values is usually much smaller than the vocabulary size of the categorical items, thus achieving memory and computational efficiency. We define a hash function $h : \mathcal{S} \to \mathbb{N}_{<B}$ that maps an item to a hash value, where B is a pre-defined upper bound known as the "bucket size." For simplicity, we use h_s to denote the hash value $h(s)$ of an item s. The hash value h_s indexes a row in the hash embedding table $E \in \mathbb{R}^{B\times d}$, yielding the embedding representation E_{h_s} of s. We use the same notation for both random variables and their realizations when the meaning is clear from context.

3.2 Probabilistic Hash Embeddings (PHE)

We first introduce our proposed encoding module for categorical items–probabilistic hash embeddings (PHE). A basic PHE has two components: a fixed hash function h and a hash embedding table E with a prior distribution $p(E)$. In this work, we assume each entry of E is independently Gaussian-distributed. Given an item s, PHE looks up the h_s-th row of E as its embedding E_{h_s}, whose distribution is $p(E_{h_s})$.

A single hash function may result in distinct inputs having the same hashing value, known as a hash collision, resulting in indistinguishable hash embeddings. With bucket size B, the collision probability is $O(1/B)$. To further reduce the collision rate, we employ universal hashing (Carter and Wegman 1977). Namely, instead of using one hash function, we use K hash functions with different random seeds but the same range. The collision probability then reduces to $O(1/B^K)$. Moreover, we share the same hash embedding table E across all K hash functions, which keeps the model size bounded.

[2]We also assume categorical features are single-valued, that is, each feature is assigned exactly one category. However, our work is compatible with multi-valued features.

We now describe how PHE encodes an item s. This procedure is illustrated in Fig. 2. With K hash functions, a categorical feature s results in K hash values $\mathbf{h}_s := \{h_s^{(1)}, \ldots, h_s^{(K)}\}$, which index K embeddings $\{E_{h_s^{(1)}}, \ldots, E_{h_s^{(K)}}\}$ where $E_{h_s^{(k)}}$ is the embedding retrieved using the k-th hash value $h_s^{(k)}$. The final representation $E_{\mathbf{h}_s} := g(E_{h_s^{(1)}}, \ldots, E_{h_s^{(K)}})$ of s is obtained using an assembly function $g : \mathbb{R}^{K\times d} \to \mathbb{R}^d$ to combine the K embeddings. Typical choices of g involve coordinate-wise summation, average, maximum, or minimum; other parametric choices of g include weighted sums where the weights come from another parametric model. The output of PHE is a probabilistic embedding $E_{\mathbf{h}_s}$ with distribution $p(E_{\mathbf{h}_s})$. Due to the shared embedding table, the memory cost of PHE is $O(Bd)$, independent of the number of hash functions K.

One core ML task is to model correlations between two variables. A common query is to compute the probability of observing a value of y given a categorical item s, i.e., $p(y|s)$. With PHE, we can compute it as

$$p(y|s) = \mathbb{E}_{p(E)}[p(y|E, \mathbf{h}_s)] = \mathbb{E}_{p(E)}[p(y|E_{\mathbf{h}_s})] \quad (1)$$

where we follow the data generating process and treat $\mathbf{h}_s$ to be the same as s (which holds in the absence of hash collisions). To compute Eq. (1), we use Monte Carlo sampling to estimate the expectation. (See Sec. G.1 for practical implementation details.) This approach enables probabilistic inference conditioned on categorical features.

3.3 Online Learning of PHE

In this subsection, we derive scalable inference algorithms for PHE in a simple yet general model. These algorithms can be adapted to other model variants. We first focus on the static setting before expanding to the online setting.

Given observations $\mathcal{D} := \{(y_i, \mathbf{s}_i)\}_{i=1}^N$ where $\mathbf{s}_i$ represents a set of categorical features, we want to learn correlations between y and $\mathbf{s}$ using PHE. We assume observations are conditionally independently and identically distributed (i.i.d.) given E, with likelihood $p(y|E_{\mathbf{h_s}})$.[3] PHE places a prior $p(E)$ over the hash embedding table E and infers the posterior given the observations $\mathcal{D}$. By Bayes rule, the posterior is $p(E|\mathcal{D}) \propto p(E)p(\mathcal{D}|E) = p(E)\prod_{i=1}^N p(y_i|E_{\mathbf{h}_{\mathbf{s}_i}})$, which is typically intractable with complex likelihoods.

Therefore, we turn to variational inference (Blei, Kucukelbir, and McAuliffe 2017; Zhang et al. 2018) and learn an approximate posterior by minimizing the Kullback-Leibler (KL) divergence $D_{\mathrm{KL}}(q_\lambda(E)|p(E|\mathcal{D}))$ between a variational distribution and the true posterior. We assume the variational posterior factorizes as $q_\lambda(E) = \prod_{b=1}^B \prod_{j=1}^d q_{\lambda_{bj}}(E_{bj})$ where each factor $q_{\lambda_{bj}}(E_{bj})$ is Gaussian with parameters $\lambda_{bj} := \{\mu_{bj} \in \mathbb{R}, \sigma_{bj} \in \mathbb{R}\}$. Finding the optimal variational distribution that minimizes the KL divergence is equivalent to finding the optimal parameters $\lambda^* := \{\lambda_{bj}^*\}$.

We assume the prior has the same factorization, with each entry independently following a standard Gaussian, i.e.,

[3]We use $E_{\mathbf{h_s}}$ to denote the concatenation of hash embeddings for each item if $\mathbf{s}$ contains more than one categorical items.

$p(E_{bj}) = \mathcal{N}(0, 1)$. Substituting $p(E)$ and $q_\lambda(E)$ into the KL divergence yields an objective function $\mathcal{L}(\lambda)$ (also known as the evidence lower bound, ELBO) to be maximized (see derivations in Supp. A):

$$\mathcal{L}(\lambda) := \mathbb{E}_{q_\lambda(E)}\left[\sum_{i=1}^{N} \log p(y_i|E_{\mathbf{h}_{s_i}})\right] - \sum_{b=1}^{B}\sum_{j=1}^{d} D_{\mathrm{KL}}(q_{\lambda_{bj}}(E_{bj})|p(E_{bj})), \quad (2)$$

where the KL divergence between two Gaussians have a closed-form solution. The KL term also serves as regularization, preventing the variational posterior from deviating too far from the prior. We optimize Eq. (2) using the reparametrization trick (Rezende, Mohamed, and Wierstra 2014; Kingma and Welling 2013) and gradient-based methods. Let $\lambda_0^* := \arg\max \mathcal{L}(\lambda)$ denote the optimal parameters, which define the approximate posterior $q_{\lambda_0^*}(E)$ that minimizes $D_{\mathrm{KL}}(q_{\lambda_0^*}(E)|P(E|\mathcal{D}))$. For prediction on an unseen data point $(\hat{y}, \hat{s})$, the predictive distribution is $p(\hat{y}|\hat{s}, \mathcal{D}) = \mathbb{E}_{p(E|\mathcal{D})}[p(\hat{y}|E_{\mathbf{h}_{\hat{s}}})] \approx \mathbb{E}_{q_{\lambda_0^*}(E)}[p(\hat{y}|E_{\mathbf{h}_{\hat{s}}})]$.

Suppose we observe a second dataset $\mathcal{D}_1 := \{(y_i, \mathbf{s}_i)\}_{i=1}^{N_1}$ to which we would like our model to adapt while maintaining effectiveness on $\mathcal{D}$. Bayes rule provides a principled online learning approach: $p(E|\mathcal{D}_1, \mathcal{D}) \propto p(E|\mathcal{D})p(\mathcal{D}_1|E)$. This allows us to accommodate both datasets without storing them simultaneous. We iteratively replace the prior with the previous posterior and repeat the inference procedure. This iteration assumes $\mathcal{D}$ and $\mathcal{D}_1$ are conditionally i.i.d. given E.

Variational inference produces an approximation of the true posterior. We thus use $q_{\lambda_0^*}(E)$ in place of $p(E|\mathcal{D})$ and infer $\tilde{p}(E|\mathcal{D}_1, \mathcal{D}) \propto q_{\lambda_0^*}(E)p(\mathcal{D}_1|E)$. This again faces intractability, so we apply variational inference with a new variational distribution $q_\lambda(E)$ to minimize $D_{\mathrm{KL}}(q_\lambda(E)|\tilde{p}(E|\mathcal{D}_1, \mathcal{D}))$. Rewrite the KL divergence gives us a new ELBO (the objective function)

$$\mathcal{L}^{(1)}(\lambda; \lambda_0^*) := \mathbb{E}_{q_\lambda(E)}\left[\sum_{i=1}^{N_1} \log p(y_i|E_{\mathbf{h}_{s_i}})\right] - \sum_{b=1}^{B}\sum_{j=1}^{d} D_{\mathrm{KL}}(q_{\lambda_{bj}}(E_{bj})|q_{\lambda_{0,bj}^*}(E_{bj})). \quad (3)$$

Compared to Eq. (2), the original prior $p(E)$ of E is replaced with the previous posterior approximation $q_{\lambda_0^*}(E)$. Upon optimization convergence, the new optimal variational distribution approximates the posterior given both datasets ($\mathcal{D}$ and $\mathcal{D}_1$). When new datasets arrive in the future, we repeat the above online learning iteration to accommodate new datasets. Although this procedure resembles traditional continual learning (Wang et al. 2023; Nguyen et al. 2018; Li et al. 2021), it differs in focusing on evolving discrete vocabularies rather than continuous feature distributions. For any specified ML task, we can plug the corresponding likelihood model into Eqs. (2) and (3) and optimize.

Variational EM. Most expressive models contain learnable parameters θ in their likelihood $p_\theta(y|E_{\mathbf{h}_s})$. One needs to learn θ in addition to inferring the posterior of E. Fortunately, a minor modification to our previously derived objective functions (Eqs. (2) and (3)) can achieve this. We replace $p(y|E_{\mathbf{h}_s})$ with the parametric one $p_\theta(y|E_{\mathbf{h}_s})$ and maximize $\mathcal{L}(\lambda, \theta)$ with respect to $\{\lambda, \theta\}$ jointly. These new objective functions are viable and correspond to variational EM algorithms (Rezende, Mohamed, and Wierstra 2014; Kingma and Welling 2013), for which we provide proofs in Sec. A.3. Let $\{\lambda_0^*, \theta^*\}$ denote the optimizers of the modified Eq. (2), $\mathcal{L}(\lambda, \theta)$. During online learning, we fix both θ^* and λ_0^* in the modified objective function Eq. (3), $\mathcal{L}^{(1)}(\lambda; \lambda_0^*, \theta^*)$, to efficiently update the posterior over hash embeddings as new categorical items arrive.

Benefits of PHE in online learning. In data streaming or continual learning setup, PHE has natural benefits in reducing catastrophic forgetting: 1) only a few embeddings need to be updated online. This sparse updating scheme seldom affects other item representations, thus having less forgetting and more computing efficiency. 2) The online updates apply Bayesian online learning, in which the prior distribution serves as a regularization of previous knowledge that also reduces forgetting. Although continually expanding the embedding table with rows for new items typically improves accuracy, it comes at a cost. A larger embedding table requires more resources to deploy, reduces memory efficiency, and slows down execution. In contrast, PHE's memory/storage cost is bounded and does not increase with the number of distinct categorical values.

3.4 Why Is PHE Superior for Online Learning?

We showcase the benefits of PHE with Bayesian online learning: 1) It is equivalent to batch learning where all data are available at once. 2) This equivalence is independent of the data arrival order.

Given a dataset $\mathcal{D} = \{(y_i, s_i)\}_{i=1}^{N}$ and a prior over E. Assume data points are conditionally i.i.d. and have likelihood $\prod_{i=1}^{N} p(y_i|E, s_i)$. The posterior obtained from Bayesian *batch* learning (where we assume the whole dataset is available at once) is $p_{\text{batch}}(E|\mathcal{D}) \propto p(E)\prod_{i=1}^{N} p(y_i|E, s_i)$.

Now, suppose $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_N)$ is any permutation of the sequence $(1, \ldots, N)$, and let data in $\mathcal{D}$ arrive sequentially per the order $\boldsymbol{\pi}$ in an online learning setup. Let $\mathcal{D}_{\boldsymbol{\pi}}$ denote the dataset that has the arrival order of $\boldsymbol{\pi}$. Suppose the posterior obtained by Bayesian *online* learning is $p(E|\mathcal{D}_{\boldsymbol{\pi}})$.

Proposition 3.1. *For every permutation $\boldsymbol{\pi}$, the posterior $p_{batch}(E|\mathcal{D}) = p(E|\mathcal{D}_{\boldsymbol{\pi}})$ almost-everywhere.*

We provide the proof in Sec. B. Prop. 3.1 shows Bayesian online learning has the same power as Bayesian batch learning irrespective of the data arrival order.

However, for point estimation, such as maximum likelihood estimation, in order to achieve the same optimality as offline batch learning, an online learning algorithm has to be carefully designed and has to have a sophisticated learning rate schedule (Opper and Winther 1999; Orabona 2019).

Demo. To demonstrate Prop. 3.1, we design a simple online learning experiment involving two categorical items that arrive alternately. Each item is repeated ten times before the other one arrives. Each item is associated with a target value, and we use both traditional deterministic hash embeddings and PHE to learn these two values in an online fashion. We plot the results in Fig. 5 in Sec. F. The results show that while

deterministic hash embeddings with a popular online learning algorithm incur large prediction errors on recurrent items, PHE makes much more accurate predictions, corroborating Prop. 3.1. Detailed settings and error analysis in Sec. F show that the *forgetting* phenomenon seen in traditional hash embeddings is caused by *parameter interference*, and that PHE alleviates the forgetting issue through adaptive regularization of the updated beliefs of hash embeddings.

Practical implementation Prop. 3.1 holds true for exact Bayesian inference, while our real-world data experiments utilize variational inference (VI) for approximate inference. There are approximation gaps (e.g., mean-field approximation), optimization gaps, and amortization gaps between VI and exact inference. In fact, we use Bayesian inference as a guide, and our experiments conform to the expectations. Secondly, unlike Bayesian neural networks that take long time to converge, PHE has sparse gradient updates and converges much faster. This is because PHE can be set to update only the embedding module while freezing other network parameters. Since each item activates at most K embeddings, by assuming independent embedding slots, the gradient updates on the embedding table are essentially sparse and data efficient. The elapsed time can be found in Sec. F.2.

4 Experiments

In this section, we conduct experiments to demonstrate the efficacy and memory efficiency of PHE in online learning settings where categorical features change. As follows, we introduce experimental protocols in Sec. 4.1. We then showcase the broad applicability of PHE as a plug-in module for a spectrum of models - both deterministic and probabilistic - and benchmark it against baselines in classification, multi-task sequence modeling, and recommendation systems in Secs. 4.2 to 4.4. Additional experiments and experimental settings are put in Secs. 4.5 and G. All results show that PHE outperforms its deterministic counterpart and performs similarly to the upper-bound collision-free embeddings in various domains and applications.

4.1 Experimental Protocols and Baselines

Experimental settings. We conducted our experiments using public datasets that contain categorical features and simulated them in data streaming environments where data points arrive sequentially. Our goal is to mimic practical settings where 1) new feature items can emerge and need to be learned, and 2) seen items can recur and need to be remembered.

Upon each data arrival, we conducted three operations in order: predict, evaluate, and update (embeddings). We report sequential results in plots and overall averaged results in tables. We repeated all experiments five times with different parameter initialization while keeping other settings fixed.

Baselines. Our work is the first work to handle open vocabulary in online dynamic environments. Appropriate baselines require handling both unbounded feature items and online updates simultaneously. To construct baselines for our setup, we need to combine existing embedding methods designed for offline inference with online updating techniques.

For embedding baselines, we use both hash embeddings (Ada) and collision-free expandable embeddings (EE). While EE is not desirable in practice due to its unbounded memory requirements and slow-down execution, we include it as a baseline to understand the performance gap (if any) resulting from our method's memory efficiency. We also implement a probabilistic version of EE, which we refer to as P-EE, to mitigate potential overfitting through Bayesian treatment.

For Ada, we choose fine-tuning as the online learning strategy, but vary the hyperparameter-training epoch-to give many candidates. We select different training epochs to account for various distribution shifts: FastAda assumes shifts are rapid while SlowAda assumes shifts happen slowly, and MedAda is in between. At each time step, FastAda trains 15 epochs as EE, while Med/SlowAda reduce training epochs to five and one, respectively. For all Ada baselines, we use the same learning rate as EE. We maintain this protocol across all applications to highlight the robustness of our method.

Training protocols. For all methods, we search hyperparameters on a validation set for each application. We test three training epochs (Fast/Medium/Slow) for Ada baselines and report the best results, giving them a competitive advantage. In all experiments, we employ a single shared hash embedding table (Coleman et al. 2024) for all categorical features. During online learning, we update only the hash embedding table while freezing all other model parameters, which were learned on an initial dataset. To determine the hash embedding table size, We selected $K = 3$ and a prime number B to support 10 times the expected size of the total vocabulary (where all tabular datasets use $B = 7$, the Retail dataset uses $B = 109$, and the MovieLens dataset uses $B = 10009$).

4.2 Application 1: Classification

We apply PHE to online learning classification tasks on four public datasets from scientific domains.

Methods. We now specify the likelihood model for classification tasks. Throughout the four datasets in use, we assume classification target variables follow categorical distributions and have a likelihood function $p_\theta(y|\mathbf{s}, \mathbf{x}) = \text{Cat}(K, \text{softmax}(\theta^\top[\mathbf{x}, E_{\mathbf{h_s}}]))$ where $\mathbf{x}$ are numeric features. This likelihood is then plugged into Eqs. (2) and (3).

Datasets. We apply four public static tabular datasets that are available in UCI Machine Learning Repository: Adult, Bank, Mushroom, and Covertype. These datasets contain a mixture of discrete and continuous columns and are collected for classification problems in various domains. For training stability, we normalized all continuous columns.

Experimental setups. To simulate the data-streaming setup, at each step we present a randomly sampled data mini-batch to the model and evaluate the online learning performance. We require only one column's item embeddings be updated, mimicking that column has a changing vocabulary. Besides, we initialize the model (both embeddings and neural network weights) with a separate random portion of the data.

Results. We reported the data-streaming online classification accuracy in Fig. 3 (left) and Fig. 6 in supplement. The

	Hash Embedding					Collision-Free Embedding	
	SlowAda	MediumAda	FastAda	PHE (ours)	Compression Ratio of PHE	EE	P-EE
Adult (↑)	82.2 ± 0.7	74.8 ± 4.5	71.1 ± 4.0	**84.1 ± 0.2**	0.09	84.2 ± 0.0	84.8 ± 0.0
Bank (↑)	**89.7 ± 0.1**	89.0 ± 0.9	86.9 ± 1.6	**89.6 ± 0.0**	0.2	90.0 ± 0.0	90.1 ± 0.0
Mushroom (↑)	97.7 ± 0.7	97.9 ± 0.5	98.3 ± 0.3	**98.8 ± 0.0**	0.62	98.8 ± 0.0	98.8 ± 0.0
CoverType (↑)	63.5 ± 0.5	59.1 ± 1.2	55.3 ± 1.2	**64.3 ± 0.2**	0.2	64.3 ± 0.1	64.0 ± 0.4
Retail (↓)	49.1±82.9	22.7±20.3	-	**3.0±0.2**	0.02	3.7±0.1	3.2±0.4
MovieLens (↓)	15.3±0.1	15.1±0.1	15.1±0.1	**14.7±0.0**	0.04	15.1±0.0	14.7±0.0

Table 1: Online learning results on all datasets. Adult, Bank, Mushroom, and Covertype are classification tasks evaluated by average accuracy, the larger the better. Retail and MovieLens-32M use mean absolute error, lower the better. All results are multiplied by 100 except Retail for visual clarity. PHE achieves the best performance among all hash embedding-based methods. We also include the compression ratios of PHE with respect to P-EE, which are computed by dividing the number of embedding parameters of PHE by the one of P-EE. (See details in Tab. 4 and Sec. G.6.) Notably, PHE takes as low as 2% memory of P-EE.

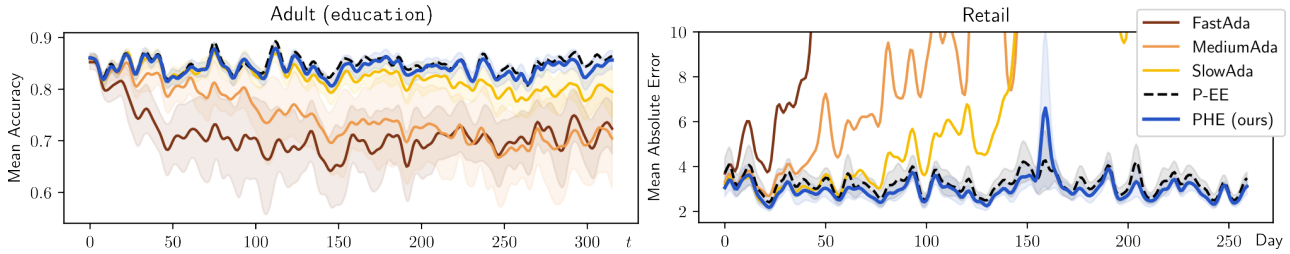

Figure 3: (**left**) Online classification results on `Adult` tabular data streams. In the parentheses is the column whose items embeddings get updated. The Ada results show a downward trend although there are no new items to learn, suggesting the deterministic hash embeddings suffer from forgetting during the learning. In contrast, the proposed PHE mitigates the forgetting issue and keeps performing as good as the upper-bound method P-EE. Other datasets in Fig. 6 in Sec. G.3 show similar conclusions. (**right**) Results of sequence modeling on `Retail` data-streams. It shows that PHE outperforms all Ada baselines that are sensitive to their optimization hyperparameters. Moreover, it is remarkable to note that PHE performs slightly better than the collision-free P-EE baseline, especially considering PHE consumes only 2% of the memory of P-EE.

facts that 1) any items seen during online learning have been learned at the initialization and that 2) the accuracy curves of Ada methods have a downward trend suggest hash embeddings suffers from forgetting. In fact, the forgetting is caused by *parameter interference* in shared hash embeddings: suppose items A and B share parameters in the hash embedding table, then updating A's embedding affect B's embedding.

We further reported an overall averaged accuracy in Tab. 1. The results show that our proposed PHE performs similarly with the upper-bound collision-free embeddings (EE), and the gap between PHE and all other deterministic counterparts proves the effectiveness of PHE in online learning. Besides, PHE is more stable and has a smaller variance. Notably, PHE applies the same set of hyperparameters and outperforms all Ada baselines across all datasets. The varying performances of the Ada baselines highlight the importance and sensitivity of hyperparameter tuning for deterministic hash embeddings. In contrast, the only demand of our method is to train the model until convergence–a simpler optimization criterion. Lastly, as summarized in Tabs. 1 and 4, PHE consumes noticeably lower memory than P-EE.

4.3 Application 2: Multi-Task Sequence Modeling

Sequence models can exploit temporal correlations among observations to make predictions based on histories. We now switch to a more sophisticated multi-task sequence modeling problem where each task has its own sequential characteristics, and we aim to personalize the sequence model for each task. Sequential Tasks are identified by categorical features.

Methods. We model sequences using deep Kalman filters (DKF) (Krishnan, Shalit, and Sontag 2015), which are latent variable models and can handle uncertainties and non-stationary processes. In sequence data, we have an additional timestamp feature t. We model the dependency between neighboring data points by a latent time variable $\mathbf{z}$. Specifically, we assume $\mathbf{z}$ is Gaussian distributed and follows the distribution $p(\mathbf{z}_i|\mathbf{z}_{i-1}, \Delta_i; \theta_z) = \mathcal{N}(\mathbf{z}_i|f_{\theta_z}(\mathbf{z}_{i-1}, \Delta_i))$ where $f_{\theta_z} := \{\mu_{\theta_z}, \Sigma_{\theta_z}\}$ is a multi-layer perceptron that outputs mean and covariance of $\mathbf{z}_i$. Δ_i is the difference in timestamp between the i-th and $i-1$-th observations. We assume the conditional likelihood model is $p(y_i|\mathbf{x}_i, E_{\mathbf{h}_{\mathbf{s}_i}}, \mathbf{z}_i)$, which is a parametric Poisson distribution.

In the above model, we need to infer the posteriors of E and $\mathbf{z}$. We assume the variational posterior distribution factorizes as $q_{\lambda,\phi}(E, \mathbf{z}_{\leq N}|\mathbf{x}_{\leq N}, \mathbf{y}_{\leq N}, \mathbf{h}_{\mathbf{s}_{\leq N}}) = q_\lambda(E)\prod_{i=1}^N q_\phi(\mathbf{z}_i|\mathbf{x}_{\leq i}, \mathbf{y}_{\leq i}, E_{\mathbf{h}_{\mathbf{s}_{\leq i}}})$ where $q_\lambda(E)$ is the same as before and $q_\phi(\mathbf{z}_i|\mathbf{x}_{\leq i}, \mathbf{y}_{\leq i}, E_{\mathbf{h}_{\mathbf{s}_{\leq i}}})$ is a Gaussian with diagonal covariance implemented as a recurrent neural network with parameters ϕ. Concretely, we use gated recurrent unit (GRU) (Chung et al. 2014).

We repeat the KL divergence minimization derivation procedure to obtain the following ELBO objective function $\mathcal{L}(\theta, \lambda, \phi) := \mathbb{E}_{q_\lambda(E)}\left[\sum_{i=1}^{N} \mathcal{L}_i(\theta, \phi|E)\right] - D_{\mathrm{KL}}(q_\lambda(E)|p(E))$ where $\mathcal{L}_i(\theta, \phi|E)$ is the conditional ELBO of the i-th data's log-likelihood $\mathcal{L}_i(\theta, \phi|E) := \mathbb{E}_{q_\phi(\mathbf{z}_i)}[\log p(\mathbf{y}_i|\mathbf{z}_i, \mathbf{x}_i, E_{\mathbf{h}_{\mathbf{s}_i}}; \theta_y)] - \mathbb{E}_{q_\phi(\mathbf{z}_{i-1})}[D_{\mathrm{KL}}(q_\phi(\mathbf{z}_i)|p(\mathbf{z}_i|\mathbf{z}_{i-1}; \theta_z))]$. We provide the full derivation of these ELBOs in Supp. A. After learning the initial dataset, we will fix all model parameters except the hash embedding table E in learning future datasets.

Datasets. We apply a public large-scale time-stamped tabular dataset, Retail. A snippet of this dataset can be found in Tab. 3. This dataset records all online transactions between 01/12/2010 and 09/12/2011 in a retail store. There are over 4,000 products and over 540K time-stamped invoice records in total. The task is to predict the sales for each product shown in each invoice given the product's historical sales.

Experimental setups. We use the first three month data to initialize the model. Then we make predictions on a daily basis following the invoice timestamp. And at each step, we predict the sales quantity for each product on invoices based on their sale history. After that, we will receive the prediction error and use it to update the product embeddings. We use mean absolute errors for evaluation (see Sec. G.4).

Results. Fig. 3 (right) shows the running performance (smoothed by a 1-D Gaussian filter): the Ada-family baselines favor shorter optimization time for Retail, as long optimization time like FastAda explodes after 50 days. (The error bar is omitted as it is too large to be meaningful.) On the other hand, PHE has lower error and is stable across all learning steps. Remarkably, on the average performance in Tab. 1, PHE significantly outperforms all baselines, including collision-free P-EE with only 2% memory usage. One possible reason is that P-EE initializes new embeddings from scratch and thus gets slow in warm-up, while PHE uses shared parameters from initial training. Similar observations also occur in the continual learning setup (see Fig. 11 and Tab. 5 in Sec. G.4). and the recommendation task below.

4.4 Application 3: Large-Scale Recommendation

Large-scale recommendation systems have seen quite a bit of change in categorical features. For example, new users or movies (categorical items) reach a streaming service, the recommender needs to incorporate them and make recommendations. We now demonstrate how PHE can assist recommendation systems in online learning.

Methods. We treat the recommendation problem as a rating prediction problem, where the task is predicting the rating a user gives to a movie. We combine PHE and Neural Collaborative Filtering (He et al. 2017) as the backbone model. We assume all ratings are iid Gaussian distributed conditioned on user, movie, and movie-genre embeddings. And we model user and movie embeddings through PHE, denoted by $E_{\mathbf{h}_u}$ and $E_{\mathbf{h}_m}$, while movie-genres are encoded as multi-hot embeddings denoted by $\mathbf{x}$. Then the likelihood is $p_\theta(y|E_{\mathbf{h}_u}, E_{\mathbf{h}_m}, \mathbf{x})$ with learnable parameters θ. θ are the weights of a two-layer neural network. We model the mean of y as the network output conditioned on input $[E_{\mathbf{h}_u}, E_{\mathbf{h}_m}, E_{\mathbf{h}_u} \odot E_{\mathbf{h}_m}, \mathbf{x}]$.

Datasets. We apply the largest MovieLens-32m (Harper and Konstan 2015) which contains 32 million ratings across over 87k movies and 200k users. These data were recorded between 1/9/1995 and 10/12/2023 for about 28 years. Each piece of data is a tuple of `(userId, movieId, rating, timestamp)`, recording when and which rating a user gave a movie. Ratings range from 0 to 5 stars with half-star increments.

Experimental setups. In implementation, ratings are normalized to $[0, 1]$ and are taken to be continuous albeit their increments are discrete. We simulate the experiment as in production – online prediction along the timestamp. The model is pre-trained on the first five years of data and then perform predict-update online learning on a daily basis. In this setup, both forgetting and adaptation in the hash embeddings are measured: the model should avoid forgetting for recurring users/movies and adapt for new users/movies. Prediction error is evaluated by mean absolute error.

Results. The results of all compared methods are shown in Fig. 6 in supplement and the memory efficiency of PHE is reported in Tabs. 1 and 4. It shows that PHE outperforms all deterministic hash embedding baselines (Fast/Medium/SlowAda) that have various forgetting-adaptation trade-offs. PHE also significantly outperforms the collision-free P-EE baseline. This is remarkable considering PHE consumes only 4% of the memory of P-EE. EE, the deterministic version of P-EE, has worse performance, possibly due to overfitting.

4.5 Additional Results

We conducted additional experiments and presented the results in Sec. G.6. We showcased additional motivating examples beside Fig. 1; demonstrated the **memory and hardware efficiency** of PHE in Tab. 4; analyzed **adaptation and forgetting** separately in Fig. 10; investigated classification and sequence modeling in classical **continual learning setup** in Fig. 11; performed **ablation studies** on the hash size B, the number of hash functions K, compared multiple **update schemes**, and the performance of **double-size** Ada baselines.

5 Conclusions

In this work we unveiled the ineffectiveness of hash embeddings in online learning of categorical features. We proposed probabilistic hash embeddings (PHE) and addressed the problem of online learning. We showcased PHE is a plug-in module for multiple ML models in various domains and applications, allowing these models to learn categorical features in a streaming fashion. We derive scalable inference algorithms to simultaneously learn the model parameters and infer the latent embeddings. Through Bayesian online learning, the model is able to adapt to new vocabularies without additional hyperparameters in a changing environment. We benchmark PHE and baselines on large-scale public datasets to demonstrate the efficacy of our method.

References

Al-Hashedi, K. G.; and Magalingam, P. 2021. Financial fraud detection applying data mining techniques: A comprehensive review from 2009 to 2019. *Computer Science Review*, 40: 100402.

Arik, S. Ö.; and Pfister, T. 2021. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 6679–6687.

Blei, D. M.; Kucukelbir, A.; and McAuliffe, J. D. 2017. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518): 859–877.

Carter, J. L.; and Wegman, M. N. 1977. Universal classes of hash functions. In *Proceedings of the ninth annual ACM symposium on Theory of computing*, 106–112.

Cheng, D. Z.; Wang, R.; Kang, W.-C.; Coleman, B.; Zhang, Y.; Ni, J.; Valverde, J.; Hong, L.; and Chi, E. 2023. Efficient Data Representation Learning in Google-scale Systems. In *Proceedings of the 17th ACM Conference on Recommender Systems*, 267–271.

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Clements, J. M.; Xu, D.; Yousefi, N.; and Efimov, D. 2020. Sequential deep learning for credit risk monitoring with tabular financial data. *arXiv preprint arXiv:2012.15330*.

Coleman, B.; Kang, W.-C.; Fahrbach, M.; Wang, R.; Hong, L.; Chi, E.; and Cheng, D. 2024. Unified Embedding: Battle-tested feature representations for web-scale ML systems. *Advances in Neural Information Processing Systems*, 36.

Du, L.; Gao, F.; Chen, X.; Jia, R.; Wang, J.; Zhang, J.; Han, S.; and Zhang, D. 2021. TabularNet: A neural network architecture for understanding semantic structures of tabular data. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 322–331.

Dulac-Arnold, G.; Evans, R.; van Hasselt, H.; Sunehag, P.; Lillicrap, T.; Hunt, J.; Mann, T.; Weber, T.; Degris, T.; and Coppin, B. 2015. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*.

Girin, L.; Leglaive, S.; Bie, X.; Diard, J.; Hueber, T.; and Alameda-Pineda, X. 2021. Dynamical Variational Autoencoders: A Comprehensive Review. *Foundations and Trends in Machine Learning*, 15(1-2): 1–175.

Han, S.; Hu, X.; Huang, H.; Jiang, M.; and Zhao, Y. 2022. Adbench: Anomaly detection benchmark. *Advances in Neural Information Processing Systems*, 35: 32142–32159.

Harper, F. M.; and Konstan, J. A. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4): 1–19.

He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; and Chua, T.-S. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, 173–182.

Huang, X.; Khetan, A.; Cvitkovic, M.; and Karnin, Z. 2020. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*.

Iida, H.; Thai, D.; Manjunatha, V.; and Iyyer, M. 2021. TABBIE: Pretrained Representations of Tabular Data. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 3446–3456.

Jerfel, G.; Grant, E.; Griffiths, T.; and Heller, K. A. 2019. Reconciling meta-learning and continual learning with online mixtures of tasks. *Advances in neural information processing systems*, 32.

Kang, W.-C.; Cheng, D. Z.; Yao, T.; Yi, X.; Chen, T.; Hong, L.; and Chi, E. H. 2021. Learning to embed categorical features without embedding tables for recommendation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 840–850.

Kim, M. J.; Grinsztajn, L.; and Varoquaux, G. 2024. CARTE: Pretraining and Transfer for Tabular Learning. In *Forty-first International Conference on Machine Learning*.

Kingma, D. P.; and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kireev, K.; Andriushchenko, M.; Troncoso, C.; and Flammarion, N. 2023. Transferable Adversarial Robustness for Categorical Data via Universal Robust Embeddings. *arXiv preprint arXiv:2306.04064*.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13): 3521–3526.

Ko, H.; Lee, S.; Park, Y.; and Choi, A. 2022. A survey of recommendation systems: recommendation models, techniques, and application fields. *Electronics*, 11(1): 141.

Kotelnikov, A.; Baranchuk, D.; Rubachev, I.; and Babenko, A. 2023. Tabddpm: Modelling tabular data with diffusion models. In *International Conference on Machine Learning*, 17564–17579. PMLR.

Krishnan, R. G.; Shalit, U.; and Sontag, D. 2015. Deep kalman filters. *arXiv preprint arXiv:1511.05121*.

Lai, F.; Zhang, W.; Liu, R.; Tsai, W.; Wei, X.; Hu, Y.; Devkota, S.; Huang, J.; Park, J.; Liu, X.; et al. 2023. {AdaEmbed}: Adaptive Embedding for {Large-Scale} Recommendation Models. In *17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23)*, 817–831.

Le, F.; Srivatsa, M.; Ganti, R.; and Sekar, V. 2022. Rethinking data-driven networking with foundation models: challenges and opportunities. In *Proceedings of the 21st ACM Workshop on Hot Topics in Networks*, 188–197.

Li, A.; Boyd, A.; Smyth, P.; and Mandt, S. 2021. Detecting and adapting to irregular distribution shifts in bayesian online learning. *Advances in neural information processing systems*, 34: 6816–6828.

Liu, H.; Di, S.; and Chen, L. 2023. Incremental Tabular Learning on Heterogeneous Feature Space. *Proceedings of the ACM on Management of Data*, 1(1): 1–18.

Liu, T.; Fan, J.; Li, G.; Tang, N.; and Du, X. 2023. Tabular data synthesis with generative adversarial networks: design space and optimizations. *The VLDB Journal*, 1–26.

Lopez-Paz, D.; and Ranzato, M. 2017. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30.

Nguyen, C. V.; Li, Y.; Bui, T. D.; and Turner, R. E. 2018. Variational Continual Learning. In *International Conference on Learning Representations*.

Opper, M.; and Winther, O. 1999. A Bayesian approach to on-line learning.

Orabona, F. 2019. A modern introduction to online learning. *arXiv preprint arXiv:1912.13213*.

Rezende, D. J.; Mohamed, S.; and Wierstra, D. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, 1278–1286. PMLR.

Rusu, A. A.; Rabinowitz, N. C.; Desjardins, G.; Soyer, H.; Kirkpatrick, J.; Kavukcuoglu, K.; Pascanu, R.; and Hadsell, R. 2016. Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

Sarker, I. H.; Kayes, A.; Badsha, S.; Alqahtani, H.; Watters, P.; and Ng, A. 2020. Cybersecurity data science: an overview from machine learning perspective. *Journal of Big data*, 7: 1–29.

Serrà, J.; and Karatzoglou, A. 2017. Getting deep recommenders fit: Bloom embeddings for sparse binary input/output networks. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, 279–287.

Shehab, M.; Abualigah, L.; Shambour, Q.; Abu-Hashem, M. A.; Shambour, M. K. Y.; Alsalibi, A. I.; and Gandomi, A. H. 2022. Machine learning in medical applications: A review of state-of-the-art methods. *Computers in Biology and Medicine*, 145: 105458.

Shi, H.-J. M.; Mudigere, D.; Naumov, M.; and Yang, J. 2020a. Compositional embeddings using complementary partitions for memory-efficient recommendation systems. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 165–175.

Shi, S.; Ma, W.; Zhang, M.; Zhang, Y.; Yu, X.; Shan, H.; Liu, Y.; and Ma, S. 2020b. Beyond user embedding matrix: Learning to hash for modeling large-scale users in recommendation. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, 319–328.

Siadati, H.; and Memon, N. 2017. Detecting structurally anomalous logins within enterprise networks. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 1273–1284.

Tito Svenstrup, D.; Hansen, J.; and Winther, O. 2017. Hash embeddings for efficient word representations. *Advances in neural information processing systems*, 30.

Wang, L.; Zhang, X.; Su, H.; and Zhu, J. 2023. A comprehensive survey of continual learning: Theory, method and application. *arXiv preprint arXiv:2302.00487*.

Weinberger, K.; Dasgupta, A.; Langford, J.; Smola, A.; and Attenberg, J. 2009. Feature hashing for large scale multitask learning. In *Proceedings of the 26th annual international conference on machine learning*, 1113–1120.

Xu, L.; Skoularidou, M.; Cuesta-Infante, A.; and Veeramachaneni, K. 2019. Modeling tabular data using conditional gan. *Advances in neural information processing systems*, 32.

Yin, P.; Neubig, G.; Yih, W.-t.; and Riedel, S. 2020. TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 8413–8426.

Yoon, J.; Yang, E.; Lee, J.; and Hwang, S. J. 2017. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*.

Zenke, F.; Poole, B.; and Ganguli, S. 2017. Continual learning through synaptic intelligence. In *International conference on machine learning*, 3987–3995. PMLR.

Zhang, C.; Bütepage, J.; Kjellström, H.; and Mandt, S. 2018. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8): 2008–2026.

Zhao, Z.; Kunar, A.; Birke, R.; and Chen, L. Y. 2021. Ctab-gan: Effective table data synthesizing. In *Asian Conference on Machine Learning*, 97–112. PMLR.