EXTRACTING AND LEVERAGING FEATURE INTERACTION INTERPRETATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recommendation is a prevalent application of machine learning that affects many users; therefore, it is crucial for recommender models to be accurate and interpretable. In this work, we propose a method to both interpret and augment the predictions of black-box recommender systems. In particular, we propose to extract feature interaction interpretations from a source recommender model and explicitly encode these interactions in a target recommender model, where both source and target models are black-boxes. By not assuming the structure of the recommender system, our approach can be used in general settings. In our experiments, we focus on a prominent use of machine learning recommendation: ad-click prediction. We found that our interaction interpretations are both informative and predictive, i.e., significantly outperforming existing recommender models. What's more, the same approach to interpret interactions can provide new insights into domains even beyond recommendation.

1 INTRODUCTION

Despite their impact on users, state-of-the-art recommender systems are becoming increasingly inscrutable. For example, the models that predict if a user will click on an online advertisement are often based on function approximators that contain complex components in order to achieve optimal recommendation accuracy. The complex components come in the form of modules for better learning relationships among features, such as interactions between user and ad features (Cheng et al., 2016; Guo et al., 2017; Wang et al., 2017; Lian et al., 2018; Song et al., 2018). Although efforts have been made to understand the feature relationships, there is still no method that can interpret the feature interactions learned by a generic recommender system, nor is there a strong commercial incentive to do so.

In this work, we identify and leverage feature interactions that represent how a recommender system generally behaves. We propose a novel approach, Global Interaction Detection and Encoding for Recommendation (`GLIDER`), which detects feature interactions that span globally across multiple data-instances from a source recommender model, then explicitly encodes the interactions in a target recommender model, both of which can be black-boxes. `GLIDER` achieves this by first utilizing feature interaction detection with a data-instance level interpretation method called LIME (Ribeiro et al., 2016) over a batch of data samples. `GLIDER` then explicitly encodes the collected global interactions into a target model via sparse feature crossing.

In our experiments on ad-click recommendation, we found that the interpretations generated by `GLIDER` are informative, and the detected global interactions can significantly improve the target model's prediction performance, even in a setting where the source and target models are the same. Because our interaction interpretation method is very general, we also show that the interpretations are informative in domains outside of recommendation, such as image and text classification.

Our contributions are as follows:

1. We propose `GLIDER` to detect and explicitly encode global feature interactions in black-box recommender systems.
2. Through experiments, we demonstrate the overall interpretability of detected feature interactions and show that they can be leveraged to improve recommendation accuracy.

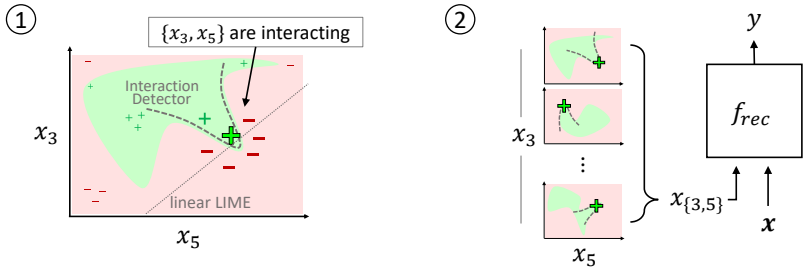


Figure 1: A simplified overview of GLIDER. ① GLIDER uses interaction detection and LIME together to interpret feature interactions learned by a source black-box (recommender) model at a data instance, denoted by the large green plus sign. ② GLIDER identifies interactions that consistently appear over multiple data samples, then explicitly encodes these interactions in a target black-box recommender model f_{rec}.

2 Related Works

Interaction Interpretations: A variety of methods exist to detect feature interactions learned in specific models but not black-box models. For example, RuleFit (Friedman et al., 2008), Additive Groves (Sorokina et al., 2008), and Tree-Shap (Lundberg et al., 2018) detect interactions learned in trees, and Neural Interaction Detection (Tsang et al., 2017) detects interactions learned in a multilayer perceptron. Some methods have attempted to interpret feature groups in black-box models, such as Anchors (Ribeiro et al., 2018), Agglomerative Contextual Decomposition (Singh et al., 2019), and Context-Aware methods (Singla et al., 2019); however, these methods were not intended to identify feature interactions.

Explicit Interaction Representation: There are increasingly methods for explicitly representing interactions in models. Cheng et al. (2016), Guo et al. (2017), Wang et al. (2017), and Lian et al. (2018) directly incorporate multiplicative cross terms in neural network architectures and Song et al. (2018) use attention as an interaction module, all of which are intended to improve the neural network's function approximation. This line of work found that predictive performance can improve with dedicated interaction modeling. Luo et al. (2019) followed up by detecting interactions in data then explicitly encoding them via feature crossing. Our work approaches this problem from a model interpretation standpoint to show that interaction interpretations are also useful in explicit encoding.

Black-Box Local vs. Global Interpretations: Data-instance level local interpretation methods are more flexible at explaining general black-box models; however, global interpretations, which cover multiple data instances, have become increasingly desirable to better summarize model behavior. Locally Interpretable Model-Agnostic Explanations (LIME) (Ribeiro et al., 2016) and Integrated Gradients (Sundararajan et al., 2017) are some of most used methods to locally interpret any classifier and neural predictor respectively. There are some methods for global black-box interpretations, such as shuffle-based feature importance (Fisher et al., 2018), submodular pick (Ribeiro et al., 2016), and visual concept extraction (Kim et al., 2018). §4.1 of this paper discusses local interaction interpretations, and §4.2-4.4 explains how we extract and utilize global interaction interpretations.

3 Notations and Background

Notations: Vectors are represented by boldface lowercase letters, such as $\mathbf{x}$ or $\mathbf{w}$. The i-th entry of a vector $\mathbf{x}$ is denoted by x_i. For a set $\mathcal{S}$, its cardinality is denoted by $|\mathcal{S}|$.

Let d be the number of features in a dataset. An *interaction*, $\mathcal{I}$, is the indices of a feature subset: $\mathcal{I} \subseteq \{1, 2, \ldots, d\}$, where *interaction order* $|\mathcal{I}|$ is always ≥ 2. A higher-order interaction always has order ≥ 3. For a vector $\mathbf{x} \in \mathbb{R}^d$, let $\mathbf{x}_\mathcal{I} \in \mathbb{R}^{|\mathcal{I}|}$ be restricted to the dimensions of $\mathbf{x}$ specified by $\mathcal{I}$.

Let a black-box model be $f(\cdot) : \mathbb{R}^p \rightarrow \mathbb{R}$. A black-box recommender model uses tabular feature types, as discussed later in this section. In classification tasks, we assume f is a class logit. p and d may be different depending on feature transformations.

Feature Interactions: By definition, a model f learns a statistical (non-additive) feature interaction $\mathcal{I}$ if and only if f cannot be decomposed into a sum of $|\mathcal{I}|$ arbitrary subfunctions δ_i, each not

depending on a corresponding interaction variable (Friedman et al., 2008; Sorokina et al., 2008; Tsang et al., 2017), i.e., $f(\mathbf{x}) \neq \sum_{i \in \mathcal{I}} \delta_i(\mathbf{x}_{\{1,\dots,d\} \setminus i})$.

For example, a multiplication between two features, x_1 and x_2, is a feature interaction because it cannot be represented as an addition of univariate functions, i.e., $x_1 x_2 \neq \delta_1(x_2) + \delta_2(x_1)$.

Recommendation Systems: A recommender system, $f_{rec}(\cdot)$, is a model of two feature types: dense numerical features and sparse categorical features. Since the one-hot encoding of categorical feature x_c can be high-dimensional, it is commonly represented in a low-dimensional embedding $\mathbf{e}_c = one\_hot(x_c)\mathbf{v}_c$ via embedding matrix $\mathbf{v}_c$.

4 GLIDER: GLOBAL INTERACTION DETECTION AND ENCODING FOR RECOMMENDATION

We now discuss the different components of GLIDER, starting from data-instance level (local) interpretations of interactions in §4.1, then global interaction detection in §4.2, and finally explicitly encoding the global interactions in §4.3. While our methodology is focused on recommender systems, it is not necessarily limited to this model type. Nonetheless, recommender systems are interesting because they have pervasive application in real-world systems, and their features are often very sparse. By sparse features, we mean features with many categories, e.g., millions of user IDs. The sparsity makes interaction detection challenging especially when applied directly on raw data because the one-hot encoding of sparse features creates an extremely large space of potential feature combinations (Fan et al., 2015).

4.1 INTERACTION DETECTION AND LIME

We start by explaining how to detect feature interactions in a black-box model at the data-instance level via interaction detection on feature perturbations.

LIME Perturbation and Inference: Given a data instance $\mathbf{x} \in \mathbb{R}^p$, LIME proposed to perturb the data instance by sampling a separate binary representation $\mathbf{x}' \in \{0,1\}^d$ of the same data instance. Let $\xi : \{0,1\}^d \rightarrow \mathbb{R}^p$ be the map from the binary representation to the perturbed data instance. Starting from a binary vector of all ones that map to the original features values in the data instance, LIME uniformly samples the number of random features to switch to 0 or the "off" state. In the data instance, "off" could correspond to a 0 embedding vector for categorical features or mean value over a batch for numerical features. It is possible for $d < p$ by grouping features in the data instance to correspond to single binary features in $\mathbf{x}'$. A key step in LIME interpretations is obtaining black-box predictions for the perturbed data instances to generate a dataset with binary inputs and prediction targets: $\mathcal{D} = \{(\mathbf{x}'^{(i)}, y^{(i)}) \mid y^{(i)} = f(\xi(\mathbf{x}'^{(i)})), \mathbf{x}'^{(i)} \in \{0,1\}^d\}$.

Feature Interaction Detection: Feature interaction detection is concerned with identifying feature interactions in a dataset (Bien et al., 2013; Purushotham et al., 2014; Lou et al., 2013; Friedman et al., 2008). Typically, proper interaction detection requires an expensive pre-processing step of feature selection to remove correlated features that adversely affect detection performance (Sorokina et al., 2008). Since the features in $\mathcal{D}$ are sampled randomly, they are uncorrelated by default, so we can directly use dataset $\mathcal{D}$ to detect feature interactions from black-box model f at a data instance $\mathbf{x}$.

f can be an arbitrary function and can generate highly nonlinear targets in $\mathcal{D}$, so we focus on detecting interactions that could have generic forms. In light of this, we use a state-of-the-art method called Neural Interaction Detection (NID), which accurately and efficiently detects generic non-additive and arbitrary-order statistical feature interactions (Tsang et al., 2017). NID detects these interactions by training a lasso-regularized multilayer perceptron (MLP) on a dataset, then identifying the features that have high-magnitude weights to common hidden units. NID is efficient by greedily testing the top-interaction candidates of every order at each of h first-layer hidden units, enabling arbitrary-order interaction detection in $O(hd)$ tests within one MLP.

Scope: We can now define a function, $\texttt{MADEX}(f, \mathbf{x})$, that inputs black-box f and data instance $\mathbf{x}$, and outputs $\mathcal{S} = \{\mathcal{I}_i\}_{i=1}^k$, a set of top-$k$ detected feature interactions. MADEX stands for "Model-Agnostic Dependency Explainer". As the name suggests, MADEX is not limited to recommender models; it can also be used for general black-box models.

In some cases, it is necessary to identify a k threshold. Because of the importance of speed for local interpretations, we simply use a linear regression with additional multiplicative terms to approximate

Algorithm 1 Global Interaction Detection in GLIDER

Input: dataset $\mathcal{B}$, recommender model f_{rec}
Output: $\mathcal{G} = \{(\mathcal{I}_i, c_i)\}$: global interactions $\mathcal{I}_i$ and their counts c_i over the dataset
1: $\mathcal{G} \leftarrow$ initialize occurrence dictionary for global interactions
2: **for** each data sample $\mathbf{x}$ within dataset $\mathcal{B}$ **do**
3: $\quad \mathcal{S} \leftarrow \texttt{MADEX}(f_{rec}, \mathbf{x})$
4: $\quad \mathcal{G} \leftarrow$ increment the occurrence count of $\mathcal{I}_j \in \mathcal{S},\ \forall j = 1, 2, \ldots, |\mathcal{S}|$
5: sort $\mathcal{G}$ by most frequently occurring interactions
6: [optional] prune subset interactions in $\mathcal{G}$ within a target number of interactions K

the gains given by interactions in $\mathcal{S}$, where k starts at 0 and is incremented until the linear model's predictions stop improving.

4.2 Global Interaction Detection

In this section, we discuss the first step of GLIDER. As defined in §4.1, MADEX takes as input a black-box model f and data instance $\mathbf{x}$. In the context of this section, MADEX inputs a source recommender system f_{rec} and data instance $\mathbf{x} = [x_1, x_2, \ldots, x_p]$. x_i is the i-th feature field and is either a dense or sparse feature. p is both the total number of feature fields and the number of perturbation variables ($p = d$). We define global interaction detection as repeatedly running MADEX over a batch of data instances, then counting the occurrences of the same detected interactions, shown in Alg. 1. The occurrence counts are not only a useful way to rank global interaction detections, but also a sanity check to rule out the chance that the detected feature combinations are random selections.

One potential concern with Alg. 1 is that it could be slow depending on the speed of MADEX. In our experiments, the entire process took less than 3 hours when run in serial over a batch of 1000 samples with ~ 40 features on a 32-CPU server. In addition, this algorithm is fully parallelizable and only needs to be run once to obtain the summary of global interactions.

4.3 Truncated Feature Crosses

Each global interaction $\mathcal{I}_i$ from Alg. 1 is used to create a synthetic feature $x_{\mathcal{I}_i}$ for a target recommender system. The synthetic feature $x_{\mathcal{I}_i}$ is created by explicitly crossing sparse features indexed in $\mathcal{I}_i$. If interaction $\mathcal{I}_i$ involves dense features, we bucketize the dense features before crossing them. The synthetic feature is sometimes called a cross feature (Wang et al., 2017; Luo et al., 2019) or conjunction feature (Rosales et al., 2012; Chapelle et al., 2015).

In this context, a cross feature is an n-ary Cartesian product among n sparse features. If we denote $X_1, X_2, \ldots, X_n$ as the set of IDs for each respective feature $x_1, x_2, \ldots, x_n$, then their cross feature $x_{\{1,\ldots,n\}}$ takes on all possible values in

$$X_1 \times \cdots \times X_n = \{(x_1, \ldots, x_n) \mid x_i \in X_i, \forall i = 1, \ldots, n\}$$

Accordingly, the cardinality of this cross feature is $|X_1| \times \cdots \times |X_n|$ and can be extremely large, yet many combinations of values in the cross feature are likely unseen in the training data. Therefore, we generate a truncated form of the cross feature with only seen combinations of values, $\mathbf{x}_{\mathcal{I}}^{(j)}$, where j is a sample index in the training data, and $\mathbf{x}_{\mathcal{I}}^{(j)}$ is represented as a sparse ID in the cross feature $x_{\mathcal{I}}$. We further reduce the cardinality by requiring the same cross feature ID to occur more than T times in a batch of samples, or set to a default ID otherwise. These truncation steps significantly reduce the embedding sizes of each cross feature while maintaining their representation power. Once cross features $\{x_{\mathcal{I}_i}\}_i$ are included in a target recommender system, it can be trained as per usual.

4.4 Model Distillation vs. Enhancement

There are dual perspectives of GLIDER: as a method for model distillation or model enhancement. If a strong source model is used to detect global interactions which are then encoded in more resource-constrained target models, then GLIDER adopts a teacher-student type distillation process. If interaction encoding augments the same model where the interactions were detected from, then GLIDER tries to enhance the model's ability to represent the interactions.

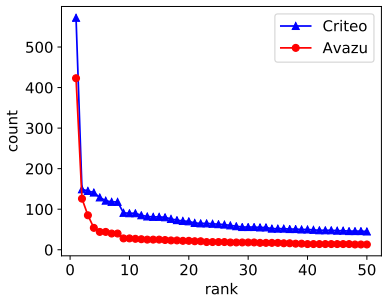


Figure 2: Occurrence counts (Total: 1000) vs. rank of detected interactions from AutoInt on Criteo and Avazu datasets

Table 2: Detected global interactions from the AutoInt baseline on Avazu data. "C14" is an anonymous feature.

Rank	Counts (Total: 1000)	Interaction
1	423	{hour, device_ip}
2	126	{hour, site_id}
3	85	{site_id, C14}
4	54	{site_id, site_domain}
5	44	{hour, site_id, device_ip}
6	44	{app_id, C14}
7	40	{site_domain, device_ip}
8	40	{hour, device_id}
9	28	{app_id, device_conn_type}
10	28	{site_id, device_model}

5 EXPERIMENTS

5.1 SETUP

In our experiments, we study the effectiveness of GLIDER on real-world data. The hyperparameters for local interaction interpretation in our experiments are as follows. For all experiments, we use 5000 perturbation samples to train the models used for interaction detection. We use NID as the interaction detector, which requires training an MLP to detect each set of interactions. The MLPs for §5.3 have architectures of 50-30-10 first-to-last hidden layer sizes, and in §5.2, architectures of 256-128-64. We apply an ℓ_1 regularization of $\lambda_1 = 5\text{e}{-5}$, and the learning rate is $5\text{e}{-3}$. In general, models are trained with early stopping on the validation set.

For LIME perturbations, we need to establish what a binary 0 maps to via ξ in the raw data instance (§4.1). In domains involving embeddings, i.e., sparse features and word embeddings, the 0 ("off") state is the zeroed embedding vector. For dense features, it is the mean feature value over a batch; for images, the mean of each RGB of the image. For our DNA experiment, we use a random nucleotide other than the original one. These settings correspond to what is used in literature (Ribeiro et al., 2016; 2018). In our graph experiment, the nodes within the neighborhood of a test node are perturbed, where each node is zeroed during perturbation.

5.2 EXPERIMENTS ON CTR RECOMMENDATION

In this section, we provide experiments with GLIDER on models trained for click-through-rate (CTR) prediction. The recommender models we study include commonly reported baselines, which all use neural networks: Wide&Deep (Cheng et al., 2016), DeepFM (Guo et al., 2017), Deep&Cross (Wang et al., 2017), xDeepFM (Lian et al., 2018), and AutoInt (Song et al., 2018).

Table 1: CTR dataset statistics

Dataset	# Samples	# Features	Total # Sparse IDs
Criteo	45,840,617	39	998,960
Avazu	40,428,967	23	1,544,428

AutoInt is the reported state-of-the-art in academic literature, so we use the model settings and data splits provided by AutoInt's official public repository[1]. For all other recommender models, we use public implementations[2] with the same original architectures reported in literature, set all embedding sizes to 16, and tune the learning rate and optimizer to try to reach or surpass the test logloss reported by the AutoInt paper (on AutoInt's data splits). From tuning, we use the Adagrad optimizer (Duchi et al., 2011) with learning rates in $\{0.01, 0.001\}$.

The datasets we use are benchmark CTR datasets with the largest number of features: Criteo[3] and Avazu[4], whose data statistics are shown in Table 1. Criteo and Avazu both contain $40+$ millions of user records on clicking ads, with Criteo being the primary benchmark in CTR research (Cheng

[1]https://github.com/shichence/AutoInt
[2]https://github.com/shenweichen/DeepCTR
[3]https://www.kaggle.com/c/criteo-display-ad-challenge
[4]https://www.kaggle.com/c/avazu-ctr-prediction

Table 3: Test prediction performance by encoding top-K global interactions in baseline recommender systems on the Criteo and Avazu datasets (5 trials). K are 40 and 20 for Criteo and Avazu respectively. "+ GLIDER" means the inclusion of detected global interactions to corresponding baselines. The "Setting" column is labeled relative to the source of detected interactions: AutoInt.

Setting	Model	Criteo		Avazu	
		AUC	logloss	AUC	logloss
Distillation	Wide&Deep	$0.8070 \pm 2e{-4}$	$0.4446 \pm 2e{-4}$	$0.7715 \pm 3e{-4}$	$0.3860 \pm 3e{-4}$
	+ GLIDER	$\mathbf{0.8080} \pm 2e{-4}$	$0.4438 \pm 1e{-4}$	$\mathbf{0.7731} \pm 3e{-4}$	$\mathbf{0.3847} \pm 2e{-4}$
	DeepFM	$0.8080 \pm 2e{-4}$	$0.4436 \pm 9e{-5}$	$0.7760 \pm 2e{-4}$	$0.3864 \pm 1e{-4}$
	+ GLIDER	$\mathbf{0.8091} \pm 1e{-4}$	$\mathbf{0.4425} \pm 1e{-4}$	$0.7768 \pm 2e{-4}$	$\mathbf{0.3854} \pm 1e{-4}$
	Deep&Cross	$0.8077 \pm 1e{-4}$	$0.4441 \pm 1e{-4}$	$0.7772 \pm 2e{-4}$	$0.3845 \pm 2e{-4}$
	+ GLIDER	$0.8082 \pm 2e{-4}$	$0.4436 \pm 8e{-5}$	$0.7779 \pm 1e{-4}$	$\mathbf{0.3826} \pm 2e{-4}$
	xDeepFM	$0.8078 \pm 3e{-4}$	$0.4438 \pm 3e{-4}$	$0.7768 \pm 2e{-4}$	$0.3832 \pm 2e{-4}$
	+ GLIDER	$\mathbf{0.8091} \pm 2e{-4}$	$\mathbf{0.4426} \pm 2e{-4}$	$0.7771 \pm 3e{-4}$	$0.3827 \pm 3e{-4}$
Enhancement	AutoInt	$0.8083 \pm 1e{-4}$	$0.4434 \pm 1e{-4}$	$0.7774 \pm 1e{-4}$	$0.3811 \pm 8e{-5}$
	+ GLIDER	$\mathbf{0.8093} \pm 1e{-4}$	$\mathbf{0.4424} \pm 6e{-5}$	$0.7773 \pm 2e{-4}$	$0.3811 \pm 2e{-4}$

Table 4: # parameters of the models in Table 3. M denotes million.

Model	Criteo	Avazu
Wide&Deep	18.1M	27.3M
+ GLIDER	19.1M (+5.6%)	27.8M (+2.0%)
DeepFM	17.5M	26.7M
+ GLIDER	18.1M (+3.5%)	27.1M (+1.3%)
Deep&Cross	17.5M	26.1M
+ GLIDER	18.7M (+6.8%)	26.7M (+2.0%)
xDeepFM	18.5M	27.6M
+ GLIDER	21.6M (+16.9%)	29.0M (+5.1%)
AutoInt	16.4M	25.1M
+ GLIDER	17.2M (+4.9%)	25.2M (+0.5%)

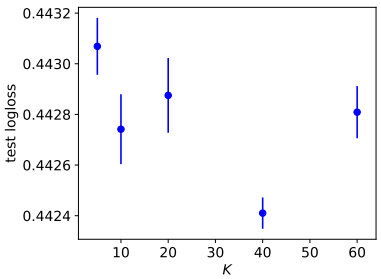


Figure 3: Test logloss vs. K of AutoInt on the Criteo dataset. Results over 5 trials are shown.

et al., 2016; Guo et al., 2017; Wang et al., 2017; Lian et al., 2018; Song et al., 2018; Luo et al., 2019).

5.2.1 Global Interaction Detection

For each dataset, we train a source AutoInt model, f_{rec}, then run global interaction detection via Algorithm 1 on a batch of 1000 samples from the validation set. A full global detection experiment finishes between 2-3 hours when run in serial on either Criteo or Avazu datasets in a 32-CPU Intel Xeon E5-2640 v2 @ 2.00GHz server, and significant speed-ups can be achieved by fully parallelizing Algorithm 1. The detection results across datasets are shown in Figure 2 as plots of detection counts versus rank. Because the Avazu dataset contains non-anonymized features, we directly show its top-10 detected global interactions in Table 2.

From Figures 2, we see that the top interactions are detected very frequently across data instances, once appearing across more than half of the batch. In Table 2, the top-interactions can be explained. For example, the interaction between "hour" (in UTC time) and "device_ip" makes sense because users - here identified by an IP address - have ad-click behaviors dependent on their time zones. We hypothesize that the global interaction detections are also informative for modeling purposes.

5.2.2 Interaction Encoding

Based on our results from the previous section (§5.2.1), we turn our attention to explicitly encoding the detected global interactions in target baseline models via truncated feature crosses (detailed in §4.3). In order to generate valid cross feature IDs, we bucketize dense features into a maximum of 100 bins before crossing them and require that final cross feature IDs occur more than $T = 100$ times over a training batch of one million samples.

We take AutoInt's top-K global interactions on each dataset from §5.2.1 with subset interactions excluded (Algorithm 1, line 6) and encode the interactions in each baseline model including AutoInt

Table 5: Prediction performance (mean-squared error; lower is better) with ($k > 0$) and without ($k = 0$) interactions for random data instances in the test sets of respective black-box models. $k = L$ corresponds to the interaction at a rank threshold. $2 \leq k < L$ are excluded because not all instances have 2 or more interactions. Only results with detected interactions are shown. At least 80% (≥ 320) of the data instances possessed interactions over 10 trials for each model/performance statistic.

	k	DNA-CNN	Sentiment-LSTM	ResNet152	GCN
linear LIME	0	$9.8\text{e}{-3} \pm 9\text{e}{-4}$	$0.101 \pm 7\text{e}{-3}$	0.25 ± 0.07	$0.080 \pm 3.0\text{e}{-4}$
MADEX	1	$8\text{e}{-3} \pm 1\text{e}{-3}$	$0.056 \pm 9\text{e}{-3}$	0.22 ± 0.06	$0.062 \pm 8.1\text{e}{-3}$
MADEX	L	$6\text{e}{-3} \pm 1\text{e}{-3}$	$0.024 \pm 7\text{e}{-3}$	0.16 ± 0.05	$0.038 \pm 9.6\text{e}{-3}$

itself. There is consensus that **0.001 logloss or AUC improvements are significant** in CTR prediction tasks (Cheng et al., 2016; Guo et al., 2017; Wang et al., 2017; Song et al., 2018). K is tuned on valiation sets, and model hyperparameters are the same between a baseline and one with encoded interactions. We set $K = 40$ for Criteo and $K = 20$ for Avazu.

We found that using GLIDER can often reach or exceed the 0.001 significance level, especially for the main Criteo benchmark dataset, as shown in Table 3. These performance gains are obtained at limited cost of extra model parameters (Table 4) thanks to the truncations applied to our feature crosses. In Figure 3, we also show how the test performance of AutoInt varies with different K on the Criteo dataset.

One one hand, the evidence that AutoInt's detected interactions can improve other baselines' performance suggests the viability of interaction distillation. On the other hand, evidence that AutoInt's performance on Criteo can improve using its own detected interactions suggests that AutoInt may benefit from learning interactions more explicitly. In either model distillation or enhancement settings, we found that GLIDER performs especially well on industry production models trained on large private datasets with thousands of features.

5.3 Interpretations on Other Domains

Since the proposed interaction interpretations by GLIDER are not entirely limited to recommender systems, we demonstrate interpretations on more general black-box models. Specifically, we experiment with the function MADEX$(\cdot)$ defined in §4.1, which inputs a black-box f, data-instance $\mathbf{x}$, and outputs a set of interactions. The models we use are trained on very different tasks: ResNet152– an image classifier pretrained on ImageNet '14 (Russakovsky et al., 2015; He et al., 2016), Sentiment-LSTM– a 2-layer bi-directional long short-term memory network (LSTM) trained on the Stanford Sentiment Treebank (SST) (Socher et al., 2013; Tai et al., 2015), DNA-CNN– a 2-layer 1D convolutional neural network (CNN) trained on MYC-DNA binding data (Mordelet et al., 2013; Yang et al., 2013; Alipanahi et al., 2015; Zeng et al., 2016; Wang et al., 2018), and GCN– a 3-layer Graph Convolutional Network trained on the Cora dataset (Kipf & Welling, 2016; Sen et al., 2008).

We first provide quantitative validation for the detected interactions of all four models in §5.3.1, followed by qualitative results for ResNet152, Sentiment-LSTM, and DNA-CNN in §5.3.2.

5.3.1 Quantitative

To provide quantitative validation of interaction interpretations of black-box models, we evaluate the predictive power of the interactions at the data instance level. As suggested in §4.1 and §4.3, encoding feature interactions is a way to increase a model's function representation, but this also means that prediction performance gains over simpler first-order models (e.g., linear regression) is a way to test the significance of the detected interactions. In this section, we use neural network function approximators for each top-interaction from the ranking $\{\mathcal{I}_i\}$ given by MADEX's interaction detector (in this case NID). Similar to the k-thresholding description in §4.1, we start at $k = 0$, which is a linear regression, then increment k with added MLPs for each $\mathcal{I}_i$ among $\{\mathcal{I}_i\}_{i=1}^{k}$ until validation performance stops improving, denoted at $k = L$. The MLPs all have architectures of 30-10 first-to-last hidden layer sizes and use the binary perturbation dataset $\mathcal{D}$ (introduced in §4.1).

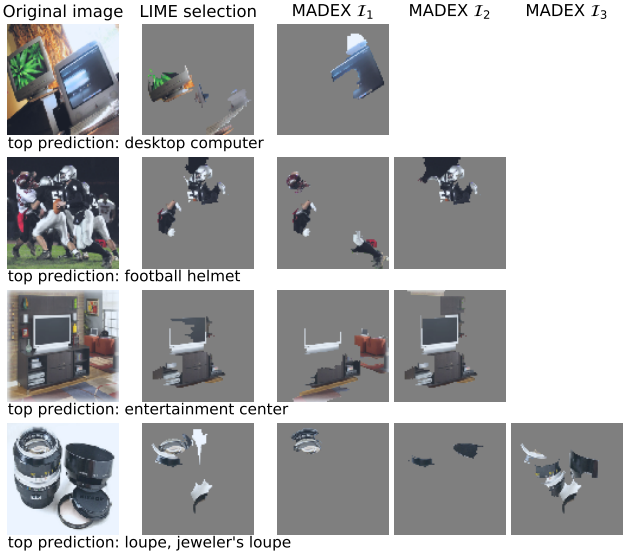
Test prediction performances are shown in Table 5 for $k \in \{0, 1, L\}$. The average number of features of $\mathcal{D}$ among the black-box models ranges from 16 to 189. Our quantitative validation shows that adding feature interactions for DNA-CNN, Sentiment-LSTM, and ResNet152, and adding node in-

teractions for GCN result in significant performance gains when averaged over 40 randomly selected data instances in the test set.

5.3.2 QUALITATIVE

For our qualitative analysis, we provide interaction interpretations via `MADEX`$(\cdot)$ of ResNet152, Sentiment-LSTM, and DNA-CNN on test examples. The interpreations are given by $\mathcal{S} = \{\mathcal{I}_i\}_{i=1}^{k}$, a set of k detected interactions, which are shown in Figures 4a and 4b for ResNet152 and Sentiment-LSTM respectively. Interactions that have majority overlap among $\mathcal{S}$ are merged, i.e., overlap coefficient > 0.5 (Vijaymeena & Kavitha, 2016). For reference, we also show the selected features by LIME's original linear regression, which takes the top-5 features that attribute towards the predicted class[5].

In Figure 4a, the `MADEX` columns show selected features from the detected interactions between Quickshift superpixels (Vedaldi & Soatto, 2008; Ribeiro et al., 2016). We see that the interactions can form a single region or multiple regions of the image, and they are complementary to LIME's feature selection. For example, the interpretations of the "deskop computer" classification show that interaction detection finds one of the computers and feature selection finds the other. For Sentiment-LSTM interpretations in Figure 4b, we also see that `MADEX`'s interactions can complement LIME's selected features. Here, the interactions show salient combinations of words, such as "science fiction" and "I like pug".



(a) ResNet152 interpretations

Original sentence	prediction	LIME selection	MADEX	
			$\mathcal{I}_1$	$\mathcal{I}_2$
The film is really not so much bad as bland.	neg.	the, so, much, bad, bland	film, not, bad, bland	
A very average science fiction film.	neg.	very, average	science, fiction	a, very, average
I like Frank the pug, though.	pos.	I, Frank, the, pug	I, like, pug	

(b) Sentiment-LSTM interpretations

Figure 4: Qualitative results of the detected interactions by `MADEX` and the selected features by LIME's original linear regression ("LIME selection") on (a) ResNet152 and (b) Sentiment-LSTM. The interpretations between `MADEX` and LIME selection are complementary.

In our experiments on DNA-CNN, we consistently detected the interaction between "CACGTG" nucleotides, which form a canonical DNA sequence (Staiger et al., 1989). The interaction was detected 76.5% out of 187 CACGTG appearances in the test set.

6 CONCLUSION

We proposed `GLIDER` that detects and explicitly encodes global feature interactions in black-box recommender systems. In our experiments, we found that the detected global interactions are informative and that explicitly encoding interactions can improve the accuracy of CTR predictions. We further validated interaction interpretations on image, text, and graph classifiers. We hope `GLIDER` encourages investigation into the complex interaction behaviors of recommender models to understand why certain feature interactions are very predictive. For future research, we wish to understand how feature interactions play a role in the integrity of automatic recommendations.

[5]Based on LIME's official code: https://github.com/marcotcr/lime

REFERENCES

Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey. Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, 33 (8):831, 2015.

Jacob Bien, Jonathan Taylor, and Robert Tibshirani. A lasso for hierarchical interactions. *Annals of statistics*, 41(3):1111, 2013.

Olivier Chapelle, Eren Manavoglu, and Romer Rosales. Simple and scalable response prediction for display advertising. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(4): 61, 2015.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794. ACM, 2016.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pp. 7–10. ACM, 2016.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

Yingying Fan, Yinfei Kong, Daoji Li, Zemin Zheng, et al. Innovated interaction screening for high-dimensional nonlinear classification. *The Annals of Statistics*, 43(3):1243–1272, 2015.

Aaron Fisher, Cynthia Rudin, and Francesca Dominici. Model class reliance: Variable importance measures for any machine learning model class, from the "rashomon" perspective. *arXiv preprint arXiv:1801.01489*, 2018.

Jerome H Friedman, Bogdan E Popescu, et al. Predictive learning via rule ensembles. *The Annals of Applied Statistics*, 2(3):916–954, 2008.

Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pp. 1725–1731. AAAI Press, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Giles Hooker. Discovering additive structure in black box functions. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 575–580. ACM, 2004.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International Conference on Machine Learning*, pp. 2673–2682, 2018.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1754–1763. ACM, 2018.

Yin Lou, Rich Caruana, Johannes Gehrke, and Giles Hooker. Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 623–631. ACM, 2013.

Scott M Lundberg, Gabriel G Erion, and Su-In Lee. Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888*, 2018.

Yuanfei Luo, Mengshuo Wang, Hao Zhou, Quanming Yao, Wei-Wei Tu, Yuqiang Chen, Wenyuan Dai, and Qiang Yang. Autocross: Automatic feature crossing for tabular data in real-world applications. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P11-1015`.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715.

Fantine Mordelet, John Horton, Alexander J Hartemink, Barbara E Engelhardt, and Raluca Gordân. Stability selection for regression-based models of transcription factor–dna binding specificity. *Bioinformatics*, 29(13):i117–i125, 2013.

W James Murdoch, Peter J Liu, and Bin Yu. Beyond word importance: Contextual decomposition to extract interactions from lstms. *International Conference on Learning Representations*, 2018.

Sanjay Purushotham, Martin Renqiang Min, C-C Jay Kuo, and Rachel Ostroff. Factorized sparse learning models with interpretable high order feature interactions. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 552–561. ACM, 2014.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144. ACM, 2016.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI Conference on Artificial Intelligence*, 2018.

Rómer Rosales, Haibin Cheng, and Eren Manavoglu. Post-click conversion modeling and analysis for non-guaranteed delivery display advertising. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pp. 293–302. ACM, 2012.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Chandan Singh, W James Murdoch, and Bin Yu. Hierarchical interpretations for neural network predictions. *International Conference on Learning Representations*, 2019.

Sahil Singla, Eric Wallace, Shi Feng, and Soheil Feizi. Understanding impacts of high-order loss approximations and features in deep learning interpretation. *arXiv preprint arXiv:1902.00407*, 2019.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. Autoint: Automatic feature interaction learning via self-attentive neural networks. *arXiv preprint arXiv:1810.11921*, 2018.

Daria Sorokina, Rich Caruana, Mirek Riedewald, and Daniel Fink. Detecting statistical interactions with additive groves of trees. In *Proceedings of the 25th international conference on Machine learning*, pp. 1000–1007. ACM, 2008.

Dorothee Staiger, Hildegard Kaulen, and Jeff Schell. A cacgtg motif of the antirrhinum majus chalcone synthase promoter is recognized by an evolutionarily conserved nuclear protein. *Proceedings of the National Academy of Sciences*, 86(18):6930–6934, 1989.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3319–3328. JMLR. org, 2017.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

Michael Tsang, Dehua Cheng, and Yan Liu. Detecting statistical interactions from neural network weights. *arXiv preprint arXiv:1705.04977*, 2017.

Andrea Vedaldi and Stefano Soatto. Quick shift and kernel methods for mode seeking. In *European Conference on Computer Vision*, pp. 705–718. Springer, 2008.

MK Vijaymeena and K Kavitha. A survey on similarity measures in text mining. *Machine Learning and Applications: An International Journal*, 3(2):19–28, 2016.

Meng Wang, Cheng Tai, Weinan E, and Liping Wei. Define: deep convolutional neural networks accurately quantify intensities of transcription factor-dna binding and facilitate evaluation of functional non-coding variants. *Nucleic acids research*, 46(11):e69–e69, 2018.

Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*, pp. 12. ACM, 2017.

Lin Yang, Tianyin Zhou, Iris Dror, Anthony Mathelier, Wyeth W Wasserman, Raluca Gordân, and Remo Rohs. Tfbsshape: a motif database for dna shape features of transcription factor binding sites. *Nucleic acids research*, 42(D1):D148–D155, 2013.

Haoyang Zeng, Matthew D Edwards, Ge Liu, and David K Gifford. Convolutional neural network architectures for predicting dna–protein binding. *Bioinformatics*, 32(12):i121–i127, 2016.

A EFFECT OF EXTRA PARAMETERS BY INTERACTION ENCODINGS VS. ENLARGED EMBEDDINGS

In this section, we study whether increasing embedding size can obtain similar prediction performance gains as explicitly encoding interactions via GLIDER. We increase the embedding dimension sizes of every sparse feature in baseline recommender models to match the total number of model parameters of baseline + GLIDER as close as possible. The embedding sizes we used to obtain similar parameter counts are shown in Table 6. For the Avazu dataset, most of the embedding sizes remain unchanged because they were already the target size. The corresponding prediction performances of all models are shown in Table 7. We observed that directly increasing embedding size / parameter counts generally did not give the same level of performance gains that GLIDER provided.

Table 6: Comparison of # model parameters between baseline models with enlarged embeddings and original baselines + GLIDER (from Tables 3 and 4). The models with enlarged embeddings are denoted by the asterick (*). The embedding dimension of sparse features is denoted by "emb. size". Percent differences are relative to baseline* models. M denotes million, and the ditto mark (") means no change in the above line.

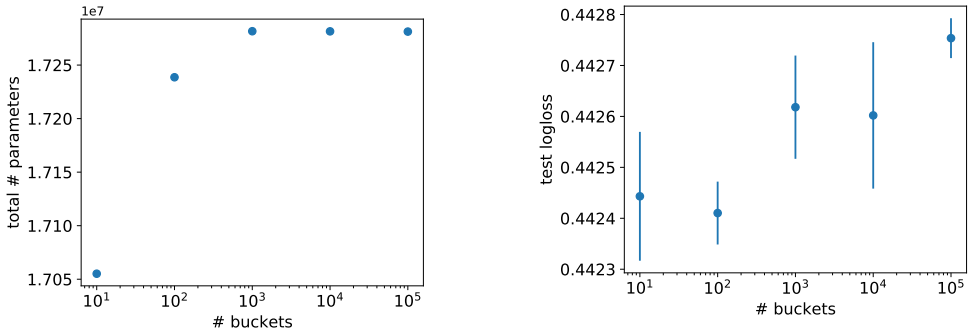
Model	Criteo		Avazu	
	emb. size	# params	emb. size	# params
Wide&Deep*	17	19.1M	16	27.3M
Wide&Deep	16	18.1M	16	"
+ GLIDER	16	19.1M (−0.1%)	16	27.8M (+2.0%)
DeepFM*	17	18.5M	16	26.7M
DeepFM	16	17.5M	16	"
+ GLIDER	16	18.1M (−2.1%)	16	27.1M (+1.3%)
Deep&Cross*	17	15.5M	16	26.1M
Deep&Cross	16	17.5M	16	"
+ GLIDER	16	18.7M (+0.8%)	16	26.7M (+2.0%)
xDeepFM*	19	21.5M	17	29.1M
xDeepFM	16	18.5M	16	27.6M
+ GLIDER	16	21.6M (+0.5%)	16	29.0M (−0.5%)
AutoInt*	17	17.4M	16	25.1M
AutoInt	16	16.4M	16	"
+ GLIDER	16	17.2M (−1.2%)	16	25.2M (+0.5%)

Table 7: Test prediction performance coresponding to the models shown in Table 6

Model	Criteo		Avazu	
	AUC	logloss	AUC	logloss
Wide&Deep*	0.8074	0.4443	0.7714	0.3859
Wide&Deep	0.8069	0.4447	"	"
+ GLIDER	0.8080	0.4439	0.7734	0.3847
DeepFM*	0.8085	0.4432	0.7761	0.3864
DeepFM	0.8081	0.4435	"	"
+ GLIDER	0.8092	0.4425	0.7770	0.3853
Deep&Cross*	0.8081	0.3445	0.7770	0.3844
Deep&Cross	0.8076	0.4440	"	"
+ GLIDER	0.8080	0.4436	0.7778	0.3826
xDeepFM*	0.8079	0.4439	0.7772	0.3835
xDeepFM	0.8079	0.4441	0.7769	0.3835
+ GLIDER	0.8093	0.4425	0.7773	0.3828
AutoInt*	0.8087	0.4431	0.7774	0.3811
AutoInt	0.8083	0.4434	"	"
+ GLIDER	0.8093	0.4424	0.7773	0.3811

B EFFECT OF DENSE FEATURE BUCKETIZATION

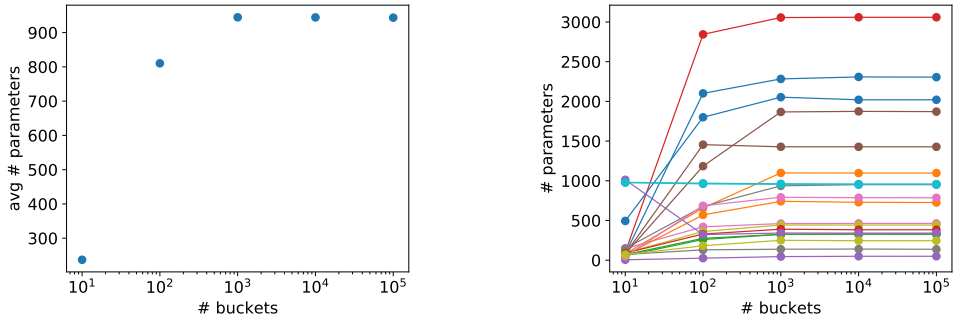
We examine the effect of dense feature bucketization on the parameter efficiency and prediction performance of AutoInt. Results are provided for the Criteo dataset, which contains 13 dense features. Figure 5 shows the effects of varying the number of dense feature buckets on the total number of parameters and the test logloss of AutoInt. Figure 6 shows the effects of varying the number of dense buckets on the embedding sizes of the cross features involving dense features. Both the effects on the average and individual embedding size are shown. 20 of the cross features involved a dense feature. Patterns to note include the largely asymptotic behavior of the parameter plots as the number of buckets increases (Figures 5a and 6). Our requirement that a valid cross feature ID occurs more than T times (§4.3) restricts the growth in parameters. In some cases, the number of cross feature IDs (embedding size) decreases (Figure 6b), which happens when the dense bucket size becomes too small to satisfy the T occurrence restriction. In Figure 5b, prediction performance degrades beyond 100 buckets, yet it is still an improvement over the baseline without cross features (0.4434 in Table 3). The degradation may be caused by overfitting.



(a) effect on total # parameters

(b) effect on test logloss

Figure 5: The effects of varying the number of dense feature buckets (in log-scale) on (a) the total # model parameters and (b) test logloss (5 trials). The model is AutoInt on the Criteo dataset.



(a) effect on avg. cross feature embedding size

(b) effect on each cross feature embedding size

Figure 6: The effects of varying the number of buckets on (a) on the average embedding size of cross features involving dense features and (b) the individual embedding sizes of the same cross features.

C TOP-RANKED INTERACTION RESULTS FOR SENTIMENT-LSTM

In this section, we show ranked results for interactions discovered by MADEX in Sentiment-LSTM.

C.1 TOP-RANKED WORD INTERACTIONS PER SENTENCE

The top-1 interactions are provided on random sentences from the SST test set. For every sentence, we preprocess it to remove stop words. If interactions are not detected in a sentence, that sentence is excluded. Results are shown in Table 8.

We use the same stop words suggested by (Manning et al., 2008), i.e., {a, an, and, are, as, at, be, by, for, from, has, he, in, is, it, its, of, on, that, the, to, was, were, will, with}.

Table 8: Top-ranked word interactions from MADEX interpreting Sentiment-LSTM on randomly selected sentences in the SST test set.

Original sentence	Word interaction
Not sweet enough to liven up its predictable story and will leave even fans of hip-hop sorely disappointed.	not, predictable
Uneasy mishmash of styles and genres.	uneasy, mishmash
Without September 11, Collateral Damage would have been just another bad movie.	bad, movie
One of the greatest family-oriented, fantasy-adventure movies ever.	family, oriented
Dazzling and sugar-sweet, a blast of shallow magnificence that only sex, scandal, and a chorus line of dangerous damsels can deliver.	shallow, magnificence
The problem with concept films is that if the concept is a poor one, there's no saving the movie.	no, saving
Despite some gulps the film is a fuzzy huggy.	despite, gulps, huggy
A modest and messy metaphysical thriller offering more questions than answers.	modest, metaphysical
Consider the title's clunk-on-the-head that suggests the overtime someone put in to come up with an irritatingly unimaginative retread concept.	unimaginative, retread
Reggio falls victim to relying on the very digital technology that he fervently scorns, creating a meandering, inarticulate and ultimately disappointing film.	inarticulate, disappointing

C.2 MOST FREQUENT WORD INTERACTIONS ACROSS SENTENCES

In order to obtain occurrence counts of word interactions, we need to detect the same word interactions across different sentences. Naturally, different sentences will often have different words, so it is nontrivial to identify consistent word interactions.

We start by collecting interaction candidates by running MADEX over all sentences in the SST test set, then identifying the word interactions that appear multiple times. Here, we make the assumption that word interactions are ordered but not necessarily adjacent or positionally bound. Therefore, a hypothetical interaction, (not, good), could be found in either sentence: "This is not good", or "This movie is not so good". The order of the words matter, so (not, good) $\neq$ (good, not).

Table 9: Most frequent interactions from MADEX across sentences in the IMDB dataset (Maas et al., 2011)

Rank	Count (Total: 40)	Interaction
1	23	well, worth
2	19	too, bad
3	19	if, you, liked
4	16	pretty, good
5	15	not, good
6	13	very, funny
7	13	not, funny
8	13	neither, nor
9	13	no, denying
10	11	good, movie

After collecting interaction candidates, we then look for sentences in the larger IMDB dataset (Maas et al., 2011) that contain the same ordered words of an interaction candidate. Therefore, each interaction candidate $\mathcal{I}_i$ will have its own set of sentences $\mathcal{W}_i$ that could potentially yield that interaction. Let $\mathcal{W}_i$ contain a random selection of 40 viable sentences for $\mathcal{I}_i$. The interactions $\{\mathcal{I}_i\}$ with the highest detection counts across corresponding $\mathcal{W}_i$ are shown in Table 9. Candidates that contain common stop words (defined in Appendix C.1) are excluded. The average sentence length is 29 words.

D ADDITIONAL QUALITATIVE RESULTS FOR RESNET152

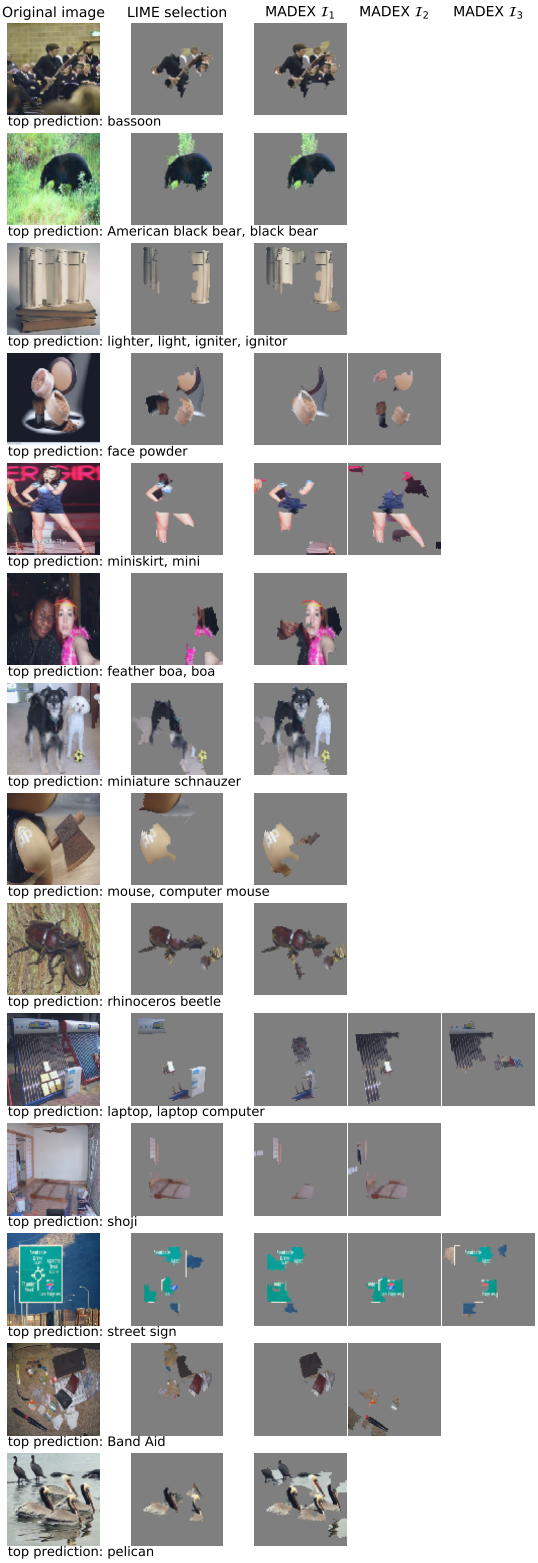


Figure 7: Additional qualitative results, following Figure 4a, on random test images in ImageNet. Overlapping interactions with overlap coefficient ≥ 0.5 are merged to reduce $|\{\mathcal{I}_i\}|$ per test image.

E DETECTION PERFORMANCE OF MADEX VS. BASELINES

We compare the detection performances between MADEX and baselines on identifying feature interactions learned by complex models, i.e., XGBoost (Chen & Guestrin, 2016), Multilayer Perceptron (MLP), and Long Short-Term Memory Network (LSTM) (Hochreiter & Schmidhuber, 1997). The baselines are Tree-Shap: a method to identify interactions in tree-based models like XGBoost (Lundberg et al., 2018), MLP-ACD+: a modified version of ACD (Singh et al., 2019; Murdoch et al., 2018) to search all pairs of features in MLP to find the best interaction candidate, and LSTM-ACD+: the same as MLP-ACD+ but for LSTMs. All baselines are local interpretation methods. For MADEX, we sample continuous features from a truncated normal distribution $\mathcal{N}(\mathbf{x}, \sigma^2\mathbf{I})$ centered at a specified data instance $\mathbf{x}$ and truncated at σ.

We evaluate interaction detection performance by using synthetic data where ground truth interactions are known (Hooker, 2004; Sorokina et al., 2008). We generate $10\text{e}3$ samples of synthetic data using functions $F_1 - F_4$ (Table 10) with continuous features uniformly distributed between -1 to 1. Next, we train complex models (XGBoost, MLP, and LSTM) on this data. Lastly, we run MADEX and the baselines on 10 trials of 20 data instances at randomly sampled locations on the synthetic function domain. Between trials, the complex models are trained with different random initialization to test the stability of each interpretation method. Interaction detection performance is computed by the average R-precision (Manning et al., 2008)[6] of interaction rankings across the sampled data instances.

Table 10: Data generating functions with interactions

$F_1(\mathbf{x}) =$	$10x_1x_2 + \sum_{i=3}^{10} x_i$
$F_2(\mathbf{x}) =$	$x_1x_2 + \sum_{i=3}^{10} x_i$
$F_3(\mathbf{x}) =$	$\exp(\lvert x_1 + x_2 \rvert) + \sum_{i=3}^{10} x_i$
$F_4(\mathbf{x}) =$	$10x_1x_2x_3 + \sum_{i=4}^{10} x_i$

Results are shown in Table 11. On the tree-based model, MADEX can compete with the tree-specific baseline Tree-Shap, which only detects pairwise interactions. On MLP and LSTM, MADEX performs significantly better than ACD+. The performance gain is especially large in the LSTM setting.

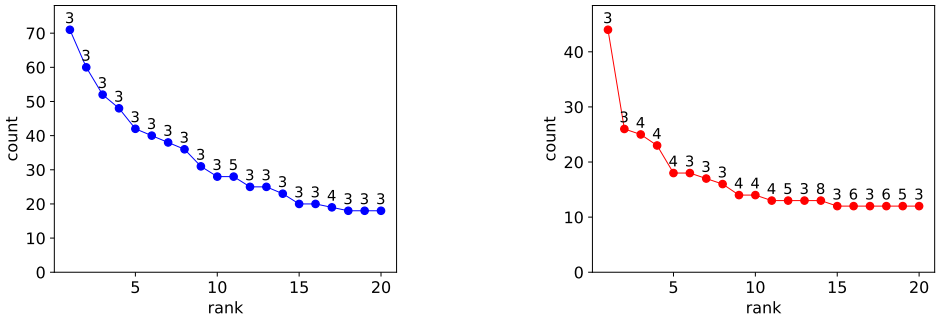
Table 11: Detection Performance in R-Precision (higher the better). $\sigma = 0.6$ (max: 3.2). "Tree" is XGBoost. *Does not detect higher-order interactions. †Requires an exhaustive search of all feature combinations.

	Tree		MLP		LSTM	
	Tree-Shap	MADEX	MLP-ACD+	MADEX	LSTM-ACD+	MADEX
$F_1(\mathbf{x})$	1 ± 0	1 ± 0	0.63 ± 0.08	1 ± 0	0.3 ± 0.2	1 ± 0
$F_2(\mathbf{x})$	1 ± 0	0.14 ± 0.09	0.41 ± 0.06	0.97 ± 0.03	0.01 ± 0.02	0.96 ± 0.03
$F_3(\mathbf{x})$	1 ± 0	1 ± 0	0.3 ± 0.2	1 ± 0	0.05 ± 0.08	1 ± 0
$F_4(\mathbf{x})$	*	0.2 ± 0.1	†	0.61 ± 0.07	†	0.54 ± 0.09

[6]R-precision is the percentage of the top-R items in a ranking that are correct out of R, the number of correct items. $R = 1$ in these experiments.

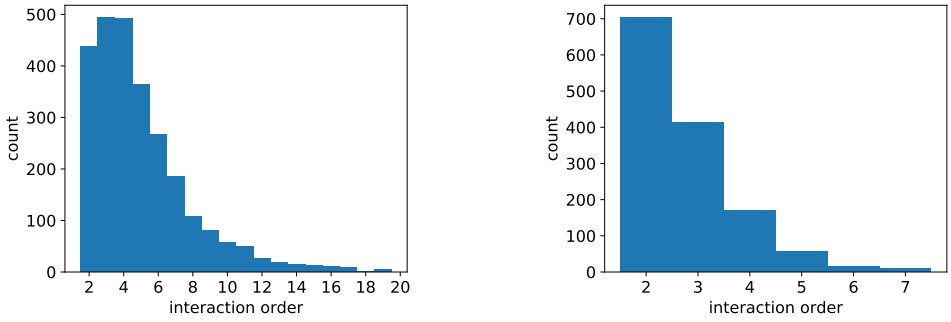
F HIGHER-ORDER INTERACTIONS

This section shows how often higher-order interactions are identified by GLIDER / MADEX. Figure 8 plots the occurrence counts of global interactions detected in AutoInt for the Criteo and Avazu dataset, which correspond to the results in Figure 2. Here we only show the occurrence counts of higher-order interactions, where the exact interaction order is annotated besides each data point. 3rd-order interactions are the most common type, and interestingly, an 8th-order interaction appears 13 times in the Avazu dataset. Figure 9 plots histograms of interaction orders for all interactions detected from ResNet152 and Sentiment-LSTM across 1000 random samples in their test sets. The average number of features are 67 and 19 for ResNet152 and Sentiment-LSTM respectively. Higher-order interactions are especially common in ResNet152.



(a) Criteo (b) Avazu

Figure 8: Occurrence counts (total: 1000) vs. rank of *only* the higher-order interactions detected from AutoInt on (a) Criteo and (b) Avazu datasets. Each data point is annotated with its interaction order.



(a) ResNet152 (b) Sentiment-LSTM

Figure 9: Histograms of interaction orders for interactions detected in (a) ResNet152 and (b) Sentiment-LSTM across 1000 random samples in respective test sets.