ARM-Net: Adaptive Relation Modeling Network for Structured Data

Shaofeng Cai
National University of Singapore
shaofeng@comp.nus.edu.sg

Kaiping Zheng
National University of Singapore
kaiping@comp.nus.edu.sg

Gang Chen
Zhejiang University
cg@zju.edu.cn

H. V. Jagadish
University of Michigan
jag@umich.edu

Beng Chin Ooi
National University of Singapore
ooibc@comp.nus.edu.sg

Meihui Zhang
Beijing Institute of Technology
meihui_zhang@bit.edu.cn

ABSTRACT

Relational databases are the *de facto* standard for storing and querying structured data, and extracting insights from structured data requires advanced analytics. Deep neural networks (DNNs) have achieved super-human prediction performance in particular data types, e.g., images. However, existing DNNs may not produce meaningful results when applied to structured data. The reason is that there are correlations and dependencies across combinations of attribute values in a table, and these do not follow simple additive patterns that can be easily mimicked by a DNN. The number of possible such "cross features" is combinatorial, making them computationally prohibitive to model. Furthermore, the deployment of learning models in real-world applications has also highlighted the need for interpretability, especially for high-stakes applications, which remains another issue of concern to DNNs.

In this paper, we present ARM-Net, an adaptive relation modeling network tailored for structured data, and a lightweight framework ARMOR based on ARM-Net for relational data analytics. The key idea is to model feature interactions with cross features selectively and dynamically, by first transforming the input features into exponential space, and then determining the *interaction order* and *interaction weights* adaptively for each cross feature. We propose a novel sparse attention mechanism to dynamically generate the interaction weights given the input tuple, so that we can explicitly model cross features of arbitrary orders with noisy features filtered selectively. Then during model inference, ARM-Net can specify the cross features being used for each prediction for higher accuracy and better interpretability. Our extensive experiments on real-world datasets demonstrate that ARM-Net consistently outperforms existing models and provides more interpretable predictions for data-driven decision making.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Mathematics of computing**; • **Applied computing**;

SIGMOD '21, June 20–25, 2021, Virtual Event, China

ACM ISBN 978-1-4503-8343-1/21/06…$15.00
https://doi.org/10.1145/3448016.3457321

KEYWORDS

Neural Networks; Structured data; Feature interaction; Multi-Head Gated Attention; Interpretability; Feature Importance

ACM Reference Format:
Shaofeng Cai, Kaiping Zheng, Gang Chen, H. V. Jagadish, Beng Chin Ooi, and Meihui Zhang. 2021. ARM-Net: Adaptive Relation Modeling Network for Structured Data. In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21), June 20–25, 2021, Virtual Event, China.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3448016.3457321

1 INTRODUCTION

Relational databases are the *de facto* standard for storing and querying structured data that are critical to the operation of most businesses [27, 32, 40, 45]. They capture a huge wealth of information that can be used for data-driven decision making, and for identifying risks and opportunities. Extracting insights from data for decision making requires advanced analytics. In particular, deep learning, which is much more complex than statistical aggregation, has recently shown great promise.

Deep neural networks (DNNs) have led to breakthroughs in images, audio and text data [1, 20, 31]. DNNs such as CNNs [20] and LSTM [31] are well-suited to particular data types for which they have been designed, e.g., CNNs for images and LSTM for sequential data. One major advantage of using DNNs is that their adoption obviates the need for manual feature engineering, which however may not produce meaningful results when applied to structured data as in relational tables. Specifically, there are intrinsic correlations and dependencies among attribute values of structured data, and such feature interactions are essential for predictive analytics. Although theoretically, a DNN can approximate any target function given sufficient data and capacity, the interactions captured for a conventional DNN layer are additive. As a result, prohibitively large and increasingly inscrutable models that stack multiple layers with nonlinear activation functions in-between are required to model such multiplicative interactions [6, 25]. Previous studies [2, 6, 13, 25] have also suggested that modeling such "cross features" implicitly with DNNs may take a substantial number of hidden units, which greatly increases the computational cost and at the same time renders it difficult to be interpreted.

Formally, structured data can be viewed as a logical table of n rows (tuples/samples) and m columns (attributes/features) [11, 32], which is extracted from relational databases via core relational operations such as select, project and join. The predictive modeling is to learn the functional dependency (predictive function) of

the *dependent attribute* y on the *determinant attributes* $\mathbf{x}$, namely, $f : \mathbf{x} \rightarrow y$, where $\mathbf{x}$ is typically called the feature vector, and y is the prediction target. The major challenge for the predictive modeling of structured data is actually how to model these dependencies and correlations between attributes, called *feature interactions*, via *cross features* [2, 6, 13, 45, 59] that create new features by capturing interactions of raw input features. Specifically, a cross feature can be defined as $\prod_{i=1}^{m} x_i^{w_i}$, i.e, the product of input features with their respective *interaction weights*. The weight w_i specifies the contribution of the i-th feature to the cross feature; $w_i = 0$ deactivates the corresponding feature x_i in the feature interaction, and the *interaction order* of a cross feature refers to the number of its non-zero interaction weights. Such cross features for relation modeling are central to the structured data learning, which enables the learning model to represent more sophisticated functions beyond simple linear aggregation of input features for predictive analytics.

A preferred alternative to DNNs in relational analytics would be modeling feature interactions explicitly, thereby obtaining generally better performance and interpretability in terms of feature attribution [18, 46]. However, the number of possible feature interactions is combinatorially large. The central problem of explicit cross feature modeling is therefore how to identify the right set of features and meanwhile, specify the corresponding interaction weights. Most existing studies sidestep this problem by capturing cross features with the interaction order confined within a predefined maximum integer [26, 32, 44, 45, 59]. However, the number of cross features still grows near exponentially as the maximum order increases. AFN [13] takes a step further by modeling cross features with logarithmic neurons [22], each of which transforms features into a logarithmic space so that the powers of features are converted into learnable coefficients, specifically, $\exp(\sum_{i=1}^{m} w_i \log x_i)$. In this manner, each logarithmic neuron can capture a specific feature interaction term of arbitrary orders. Nonetheless, AFN has inherent limitations that the input features of interaction terms are restricted to positive values on account of the logarithmic transformation, and the interaction order of each interaction term is unconstrained and remains static after training.

We contend that cross features should only account for certain input features, and feature interactions should be modeled dynamically in an instance-aware manner. The rationale is that not all input features are constructive for the interaction term, and modeling with irrelevant features could simply introduce noise and thus reduce effectiveness and interpretability. In particular, the deployment of learning models in real-world applications has highlighted the need for not only accuracy, but also efficiency and interpretability [7, 18, 38, 46]. Notably, understanding the general behavior and the whole logic of the learning model (*global interpretability*) and providing reasons for making a specific decision (*local interpretability*) [18, 46] is of paramount importance for high-stakes applications in critical decisions making, e.g., healthcare or finance [61]. Despite their great predictive power, many "black box" models such as DNNs model inputs in an implicit way, which is inexplicable and sometimes can learn unintended patterns [53, 60]. In this light, explicitly modeling feature relations with a minimum set of constituent features adaptively would yield desirable inductive biases for effectiveness, efficiency and interpretability.

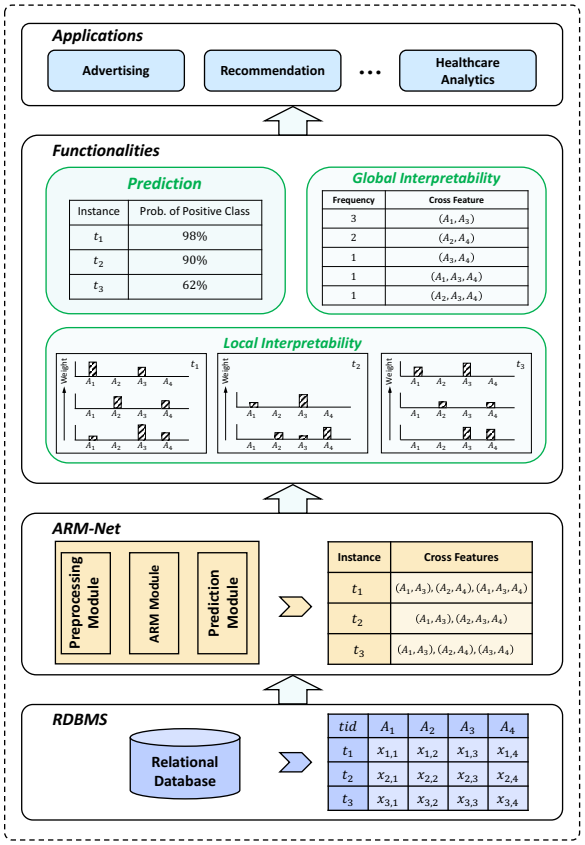

Figure 1: The overview of ARMOR based on the relation modeling of ARM-Net for structured data analytics.

In this paper, we present an **A**daptive **R**elation **M**odeling **Net**work (ARM-Net) for structured data, which models feature interactions of arbitrary orders selectively and dynamically. To this end, we address the issue of adaptive feature selection for cross features with a gated attention mechanism, and model feature interaction weights and the interaction order dynamically with novel exponential neurons. The key idea is to model feature interactions in the exponential space and determine the interaction weights dynamically based on the current input instance, i.e., each tuple of the structured data. In particular, exponential neurons transform input features into an exponential space, and then the interaction weights are dynamically determined by attention alignment followed by the gating with sparse softmax [41]. As a consequence, each exponential neuron captures a specific cross feature of arbitrary orders with irrelevant features filtered by gated attention adaptively. To the best of our knowledge, we are the first to propose an adaptive relation modeling network for structured data. Based on ARM-Net, we develop a lightweight **A**daptive **R**elation **M**odeling framew**OR**k (ARMOR) for relational data analytics. The overview of ARMOR is illustrated in Figure 1. In the training phase, ARM-Net is trained to model feature interactions in a selective and dynamic manner. In the inference phase, given the input tuples, ARMOR supports key functionalities such as prediction, global interpretability and local interpretability for various structured data analytics. Let us consider a use case when a company would like to make predictions

about monthly sales, and a data table containing attribute fields of (*month*, *regionID*, *storeID*, *productID*) and some predictive targets *total sales* are available. In such an application, ARMOR can learn to predict the monthly sales target and disclose the cross features that have been used to make the prediction. In this example, a particular store may perform better at selling a particular product locally, and all the stores may sell way more of a particular product in certain months/regions globally. ARMOR is able to dynamically identify the interactions of these features and highlight such cross features in human understandable terms, on which the predictive analytics are based. We summarize the main contributions as follows.

- We propose exponential neurons and the gated attention mechanism for adaptive feature interaction modeling, which capture cross features selectively and dynamically.
- We develop an adaptive relation modeling framework, which during inference, takes structured data tuples as inputs and produces compact relational representations for predictive analytics, and meanwhile provides both global and local interpretation results for deriving insights.
- We conduct extensive experiments on real-world datasets. The results confirm that our exponential neurons with gated attention can adaptively capture cross features of arbitrary orders, and our ARMOR consistently achieves superior prediction performance and better interpretability.

In the remainder of this paper, we introduce preliminaries in Section 2. In Section 3, we present ARM-Net with a detailed introduction of its modules and optimization schemes, and then discuss the effectiveness, efficiency and interpretability respectively. Experimental results and evaluation of effectiveness and interpretability are provided in Section 4. We review related work in Section 5. Finally, we conclude the paper and envision future work in Section 6.

2 PRELIMINARIES

In this section, we introduce the preliminaries of structured data, Logarithmic Neural Network (LNN) and sparse softmax. We first discuss structured data considered in this paper. Then, we present the two relevant techniques central to our ARM-Net, namely LNN for modeling higher-order interactions and sparse softmax for sparsely selecting informative features. Scalars, vectors and matrices are denoted as x, $\mathbf{x}$ and $\mathbf{X}$ respectively.

Structured Data. Most businesses to date rely on structured data for their data storage and predictive analytics [11, 27, 32, 45]. Relational Database Management System (RDBMS) has become the predominant database system adopted by the industry. Structured data (or relational data, tabular data) [11, 32] refers to the type of data that can be represented in tables. Structured data is generally stored in a set of tables (relations) $\{\mathbf{T}_1, \mathbf{T}_2, \dots\}$ of columns and rows, which can be extracted from a relational database with feature extraction queries, e.g., the projection, natural join, and aggregation of these tables in the database [9, 11, 27, 32]. Each column conforms to a domain of certain constraints and corresponds to a specific feature in learning models. Tables of structured data are linked to other tables via *foreign key* attributes, i.e., the value of a column in a table relates to a unique row of another table. For ease of discussion, we therefore formulate structured data as one logical table $\mathbf{T}$ of n rows and m columns. Specifically, each row can be denoted as a tuple $(\mathbf{x}, y) = (x_1, x_2, \dots, x_m, y)$, where y is the dependent attribute (prediction target), $\mathbf{x}$ is the determinant attributes (feature vector), and x_i is the i-th attribute value which is either numerical or categorical. As existing solutions are not well-suited in terms of effectiveness, efficiency and interpretability, there has been a growing interest in designing learning models for structured data [3, 15, 45] and integrating predictive analytics into RDBMS [9, 27, 32, 58].

Logarithmic Neural Network. Feed-forward neural networks (FNNs) are known to be *universal function approximators*. Each neuron y of FNN simply aggregates inputs $\mathbf{x}$ with corresponding learnable weights $\mathbf{w}$: $y = \sum_i^m w_i x_i$ followed by a non-linear activation. Although FNNs can arbitrarily approximate any continuous function [16], they are not well suited to model unbounded non-linear functions [22], particularly, functions involving multiplication, division and power interactions between inputs. Logarithmic Neural Networks [13, 22] (LNNs) instead approximate these higher-order interactions directly in the logarithmic space $y = \exp(\sum_i^m w_i \ln x_i) = \prod_i^m x_i^{w_i}$, where each logarithmic neuron y operates on inputs $\mathbf{x}$ that are transformed into the logarithmic space. As a consequence, multiplication, division and powers interactions between inputs can be converted into addition, subtraction and multiplication of the weights $\mathbf{w}$. With such logarithmic transformation, the interaction weights between inputs can be determined adaptively, and each logarithmic neuron captures a specific interaction term, producing one cross feature accordingly.

Sparse Softmax. Softmax transformation [10] is a crucial function in neural network models, which maps an input vector $\mathbf{z}$ into a probability distribution $\mathbf{p}$ whose probabilities correspond proportionally to the exponentials of input values, i.e., $\text{softmax}(\mathbf{z})_j = \frac{\exp(z_j)}{\sum_i \exp(z_i)}$. The output of softmax can thus be subsequently used as the model output denoting the class probabilities or attention weights indicating the relative importance of inputs in the attention mechanism. Denoting the d-dimension probability simplex as $\Delta^d := \{\mathbf{p} \in \mathbb{R}^d : \mathbf{p} \geq 0, \|\mathbf{p}\|_1 = 1\}$, softmax can then be interpreted in the variational form with entropy [56]:

$$\text{softmax}(\mathbf{z}) = \underset{\mathbf{p} \in \Delta^d}{\text{argmax}} \{\mathbf{p}^T \mathbf{z} + \mathbf{H}^{\text{S}}(\mathbf{p})\} \tag{1}$$

where $\mathbf{H}^{\text{S}}(\mathbf{p}) = -\sum_j p_j \log p_j$ is the Shannon entropy. The softmax function is straightforward to compute, differentiable and convex and is thus extensively used in neural networks. However, softmax always assigns dense positive credits (probabilities) to all inputs, which compared with sparse credit assignment, is less interpretable and effective in attention alignment [41]. To mitigate this issue, sparse softmax [37, 41] is proposed to produce sparse distributions, by assigning zero probability to certain outputs. Entmax [41] generalizes dense and sparse softmax with Tsallis α-entropies $\mathbf{H}^{\text{T}}_\alpha(\mathbf{p})$ [52]:

$$\alpha\text{-entmax}(\mathbf{z}) = \underset{\mathbf{p} \in \Delta^d}{\text{argmax}} \{\mathbf{p}^T \mathbf{z} + \mathbf{H}^{\text{T}}_\alpha(\mathbf{p})\} \tag{2}$$

where $\mathbf{H}^{\text{T}}_\alpha(\mathbf{p}) = \frac{1}{\alpha(\alpha-1)} \sum_j (p_j - p_j^\alpha)$ if $\alpha \neq 1$, else $\mathbf{H}^{\text{T}}_1(\mathbf{p}) = \mathbf{H}^{\text{S}}(\mathbf{p})$. With a larger α, α-entmax tends to produce a sparser probability distribution and consequently, a larger proportion of input features will be filtered out.

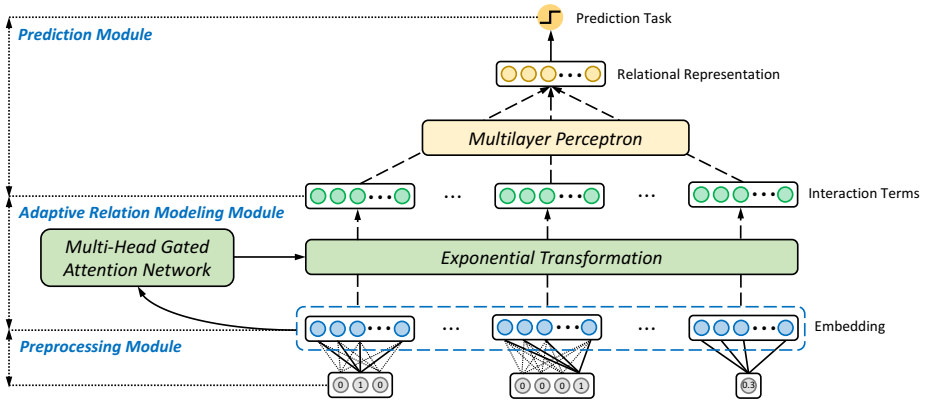

Figure 2: The overview of ARM-Net.

3 ARM-NET FOR STRUCTURED DATA ANALYTICS

In this section, we first present the overview of ARM-Net, which is the core component of ARMOR and is designed to adaptively model feature interactions for structured data. We then elaborate on each module of ARM-Net and introduce the optimization scheme. We further discuss the feature interactions captured in the relation modeling of ARM-Net for local and global interpretability, and analyze both effectiveness and efficiency.

3.1 Overview

The overview of ARM-Net is illustrated in Figure 2. The main intuition is that attribute value distributions have "structure", and feature interactions have "structure". This structure can be learned on a per-input basis for more effective and explicable interaction modeling. We further contend that not all features are useful in interaction modeling, and capturing cross features by simply introducing more constituent features is neither efficient nor effective. Instead, we propose to capture cross features in a selective and dynamic manner.

Specifically, given input features $\mathbf{x}$, we propose to first transform each input feature into the exponential space. Next, we adaptively model feature interaction terms with our proposed exponential neurons with a multi-head gated attention mechanism. Each exponential neuron is designed to model a specific cross feature of arbitrary orders explicitly, which is more effective and understandable by humans; and the gated attention generates the interaction weights dynamically to filter noisy features selectively, which makes the modeling process more efficient, effective, and interpretable. We then feed the relational representation, i.e., the captured cross features that model the interactions, to a final prediction module for the prediction task. We call our model ARM-Net, since it performs Adaptive Relation Modeling.

3.2 Architecture

3.2.1 Preprocessing Module. The input of ARM-Net can be denoted as a vector $\mathbf{x} = [x_1, \ldots, x_m]$ of m attribute fields, which can be either categorical or numerical. Each field of the raw input features $\mathbf{x}$ is then transformed into an embedding vector. In particular, categorical fields are mapped into a low-dimensional latent space via an embedding lookup, i.e., $\mathbf{e}_i = \mathbf{E}_i[:, x_i], \mathbf{e}_i \in \mathbb{R}^{n_e}$, where $\mathbf{E}_i$ is the embedding matrix of the i-th attribute field, and n_e is the embedding size. Note that embedding vectors of $\mathbf{E}_i$ correspond to their respective categories in this field, and the number of different categories in a field can be extremely large for real-world applications. Meanwhile, numerical attribute fields also need to be transformed into embeddings of the same dimension: $\mathbf{e}_j = x_j \hat{\mathbf{e}}_j$, where x_j is a scalar feature value (e.g., scaled into $(0, 1]$), and $\hat{\mathbf{e}}_j$ is the embedding vector for the j-th feature. In this way, we can obtain fixed size inputs, i.e., embedding vectors $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_m]$, for the m attribute fields uniformly as the model input.

3.2.2 Adaptive Relation Modeling Module. To dynamically model the feature interactions, we propose Adaptive Relation Modeling Module (ARM-Module) as illustrated in Figure 3, in which we devise novel exponential neurons to model cross features of arbitrary orders. Compared with LNNs [13, 22], our exponential neurons relax the limitation of positive inputs for logarithmic neurons. With exponential neurons, ARM-Module can determine the orders dynamically via a multi-head gated attention mechanism on a per-instance basis. The detailed design of ARM-Module is introduced as follows.

Exponential Neurons. First, to address the limitation that inputs must be maintained positive for logarithmic neurons [13, 22], we propose to process inputs in the exponential space instead of the logarithmic space, i.e., treat each input feature as the exponent of the natural exponential function. We then propose the exponential neuron for exponential transformation accordingly:

$$\begin{aligned} \mathbf{y}_i &= \exp\Big(\sum_{j=1}^{m} w_{ij}\mathbf{e}_j\Big) \\ &= \exp(\mathbf{e}_1)^{w_{i1}} \circ \exp(\mathbf{e}_2)^{w_{i2}} \circ \cdots \circ \exp(\mathbf{e}_m)^{w_{im}} \end{aligned} \tag{3}$$

$$\frac{\partial \mathbf{y}_i}{\partial \mathbf{e}_j} = \mathrm{diag}(w_{ij}\mathbf{y}_i), \frac{\partial \mathbf{y}_i}{\partial w_{ij}} = \mathbf{y}_i \circ \mathbf{e}_j \tag{4}$$

where $\circ$ denotes Hadamard product, $\exp(\cdot)$ and the corresponding exponent w_{ij} are applied element-wise, and $\mathrm{diag}(\cdot)$ is the diagonal matrix function. Feature interactions modeled in exponential neurons are thus based on feature embeddings transformed in the exponential space, namely, $\exp(\mathbf{e}_j)$, and the interaction weights are given by the weights $\mathbf{w}_i$ correspondingly.

Multi-Head Gated Attention. For the i-th exponential neuron $\mathbf{y}_i$, the power terms $\mathbf{w}_i = [w_{i1}, w_{i2}, \ldots, w_{im}]$ are dynamically determined via a multi-head gated attention mechanism on a per-input basis. Such a selective attention mechanism guides each exponential neuron to attend to more informative features and suppress others for adaptive relation modeling, and due to its flexibility, the relation modeling process is more effective and parameter-efficient in capturing cross features than static approaches [8, 13, 44].

To this end, we associate each exponential neuron with a learnable attention weight *value vector* $\mathbf{v}_i \in \mathbb{R}^m$ shared across instances, which encodes the *global attention weights* for each of the respective m attribute field embeddings. Further, we propose to dynamically recalibrate attention values $\mathbf{v}_i$ of the i-th exponential neuron by the attention query alignment with embeddings of attribute fields followed by sparse softmax, so that noisy feature terms can be filtered, and the resultant interaction terms are only dedicated to informative features and thus more effective and interpretable. Specifically, we associate each exponential neuron with another attention *query vector* $\mathbf{q}_i \in \mathbb{R}^{n_e}$ to generate the attention recalibration weights dynamically via the bilinear attention alignment score [35]:

$$\begin{aligned} \tilde{z}_{ij} &= \phi_{att}(\mathbf{q}_i, \mathbf{e}_j) = \mathbf{q}_i^\top \mathbf{W}_{att} \mathbf{e}_j \\ \mathbf{z}_i &= \alpha\text{-entmax}(\tilde{\mathbf{z}}_i), \tilde{\mathbf{z}}_i \in \mathbb{R}^m \end{aligned} \tag{5}$$

where $\mathbf{W}_{att} \in \mathbb{R}^{n_e \times n_e}$ is the weight matrix for the bilinear attention function ϕ_{att} shared among exponential neurons, and α-entmax is the sparse softmax introduced in Section 2, the sparsity of which increases with a larger α. The bilinear attention first projects the embedding vector $\mathbf{e}_j$ of the j-th attribute field dynamically into the query space of the i-th neuron, and then generates the attention score $\tilde{z}_{ij}$ by the alignment with the corresponding query vector $\mathbf{q}_i$. We then calculate attention weights and obtain the dynamic feature interaction weights for each exponential neuron:

$$\mathbf{w}_i = \mathbf{z}_i \circ \mathbf{v}_i, \mathbf{w}_i \in \mathbb{R}^m \tag{6}$$

where the recalibration weights $\mathbf{z}_i$ is introduced as a *gate* to deactivate noisy features and tone down less informative features for more effective interaction modeling and better interpretability, which is reminiscent of the output gate in LSTM [23] and the output channel recalibration in SENet [24].

Further, instead of modeling feature interactions with a single bilinear attention function, we adopt the multi-head attention mechanism [30, 54, 55] with K sets of different bilinear attention functions and attention value/query vectors accordingly, in order to support interaction modeling from different representation spaces. Denoting the number of exponential neurons as o for each attention head, we then have o attention weight value vectors $\mathbf{V}^{(k)} = [\mathbf{v}_1^{(k)}, \ldots, \mathbf{v}_o^{(k)}], \mathbf{V}^{(k)} \in \mathbb{R}^{m \times o}$ and query vectors $\mathbf{Q}^{(k)} = [\mathbf{q}_1^{(k)}, \ldots, \mathbf{q}_o^{(k)}], \mathbf{Q}^{(k)} \in \mathbb{R}^{n_e \times o}$, and obtain representations from the o exponential neurons $\mathbf{Y}^{(k)} = [\mathbf{y}_1^{(k)}, \ldots, \mathbf{y}_o^{(k)}]$ for the k-th attention function $\phi_{att}^{(k)}$. There are thus $K \cdot o$ feature interaction terms altogether, each of which models a specific feature interaction with the corresponding attention function and value/query vectors.

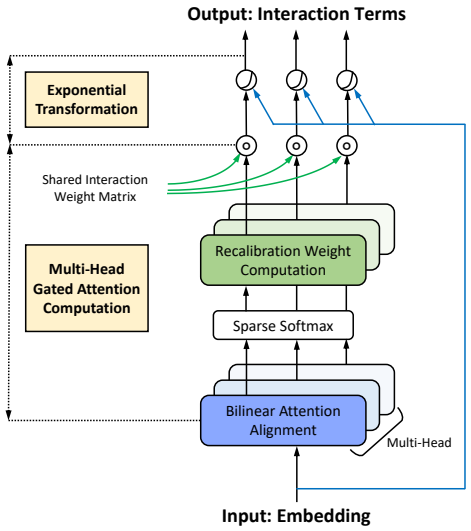

Figure 3: Adaptive Relation Modeling Module of ARM-Net.

3.2.3 Prediction Module. With ARM-Module, we dynamically model vector-wise feature interactions via exponential neurons with the multi-head gated attention and obtain $K \cdot o$ cross features. We vectorize and concatenate all the interaction terms $\{\mathbf{Y}^{(1)}, \ldots, \mathbf{Y}^{(K)}\}$ captured with exponential neurons to $\mathbf{y}$, where $\mathbf{y} \in \mathbb{R}^{K \cdot o \cdot n_e}$, then feed $\mathbf{y}$ to a multilayer perceptron (MLP) to further capture element-wise non-linear feature interactions, and obtain a vector $\mathbf{h}$ that encodes the relational representations:

$$\mathbf{h} = \phi_{MLP}(\mathbf{y}), \ \phi_{MLP} : \mathbb{R}^{K \cdot o \cdot n_e} \mapsto \mathbb{R}^{n_h} \tag{7}$$

which is then forwarded to the final prediction layer:

$$\hat{\mathbf{y}} = \mathbf{W}_p \mathbf{h} + \mathbf{b}_p \tag{8}$$

where $\mathbf{W}_p \in \mathbb{R}^{n_p \times n_h}$ and $\mathbf{b}_p \in \mathbb{R}^{n_p}$ are the weight and bias respectively, and n_p corresponds to the number of the prediction targets of the learning task. We propose to enhance the learning of ARM-Net with the ensemble of DNN, which is introduced in the following section.

3.3 Optimization

ARM-Net Training. As shown in Equation 7 and Equation 8, ARM-Net can be adopted in various learning tasks, such as classification, regression with a proper objective function, e.g., MSE, cross entropy, etc. Take the binary classification tasks for example, the corresponding objective function is the binary cross entropy:

$$\text{Logloss}(\hat{\mathbf{y}}, \bar{\mathbf{y}}) = -\frac{1}{N} \sum_{i=1}^{N} \bar{y}_i \log \sigma(\hat{y}_i) + (1 - \bar{y}_i) \log(1 - \sigma(\hat{y}_i)) \tag{9}$$

where $\hat{\mathbf{y}}$ and $\bar{\mathbf{y}}$ are the prediction labels and the ground truth labels respectively, N is the number of training instances, and $\sigma(\cdot)$ is the sigmoid function. With the objective function specified, ARM-Net can be trained effectively with popular gradient-based optimizers such as SGD, Adam [28] and etc.

ARM-Net Ensemble with a DNN. Deep neural networks are known to be universal approximators with a sufficient number of hidden units and are powerful in capturing nonlinear feature interactions. Following prior research [13, 19, 33, 57], we further enhance the prediction output of ARM-Net with the deep interaction modeling of a DNN. For the DNN model, we adopt another set of embedding vectors of attribute fields and vectorize them as the raw input to capture the fine-grained nonlinear feature interactions. In particular, the final prediction output of the ensemble of ARM-Net and a DNN is obtained by:

$$\hat{\mathbf{y}}_{+} = \mathbf{w}_1 \hat{\mathbf{y}}_{ARM-Net} + \mathbf{w}_2 \hat{\mathbf{y}}_{DNN} + \mathbf{b}_f \tag{10}$$

where $\mathbf{w}_1$ and $\mathbf{w}_2$ are the ensemble weights for ARM-Net and DNN respectively, $\mathbf{b}_f \in \mathbb{R}^{n_p}$ is the bias, and n_p is again the number of the prediction targets of the learning task. The entire ensemble model can then be readily trained end-to-end by optimizing the objective function, e.g., Equation 9. We denote the ensemble model of ARM-Net and a DNN as **ARM-Net+**.

3.4 Analysis and Discussion

Effectiveness. Most existing studies for feature interaction modeling either capture possible cross features statically with a predefined maximum interaction order [26, 44, 59], or model cross features in an implicit manner [12, 48]. However, different input instances should have distinct relations with varying constituent attributes. Some relations are informative, whereas others might be simply noise. Therefore, modeling cross features in a static manner is both parameter and computation inefficient, and less effective. In particular, each exponential neuron output $y_i^{(k)}$ captures a specific cross feature of arbitrary orders and could potentially represent any combination of interacting features by deactivating other features. With the proposed exponential neurons and multi-head gated attention mechanism, ARM-Net can thus model feature interactions adaptively for better predictive performance.

Interpretability. Interpretability measures the extent to which decisions made by the model can be understood by human [7, 18, 38], which engenders user trust and provides new insights. There exist general post-hoc interpretation approaches explaining how a *black box* model works, including perturbation based [34, 46], gradient-based [51] and attention-based [47] methods. However, explanations produced by a different model is often not reliable, which can be misleading [18, 34]. ARM-Net instead follows *transparent box design* [18], whose internal modeling process is more transparent and thus explainable for relational analytics.

Specifically, the interaction weights $\mathbf{w}_i^{(k)}$ for each feature interaction term $\mathbf{y}_i^{(k)}$ are derived from the attention values $\mathbf{v}_i^{(k)}$ shared across instances globally and are recalibrated by attention alignment dynamically for each instance. Hence, the shared attention weight *value vectors* encode the *global interaction weights* before calibration for respective attribute fields over the instance population. We can thus aggregate the absolute values of all the *value vectors* of exponential neurons for *global interpretability*, which indicates the general focus of ARM-Net on each attribute field in the data, namely the *feature importance* of attribute fields. Meanwhile, the proposed gated attention mechanism also encourages *local interpretability*, which supports feature attribution on a per-input basis. Note that each exponential neuron specifies a sparse set of attribute fields that are being used dynamically via the attention alignment. Therefore, we can identify the cross features captured dynamically and similarly, obtain relative feature importance by aggregating the interaction weights of all exponential neurons for each instance. The cross feature terms captured can also be analyzed globally/locally for understanding the internal modeling process.

Efficiency. Besides effectiveness and interpretability, model complexity is another important criterion for model deployment in real-world applications. For simplicity of analysis and to reduce the number of hyperparameters, we set the embedding and attention vector sizes to n_e, and denote the parameter size of all MLPs in ARM-Net as n_w. Recall that m, K, o denotes the number of attribute fields, attention heads and exponential neurons for each attention head respectively. Then for the preprocessing module, there are $O(Mn_e)$ feature embedding parameters with only m attribute field embeddings being used for each instance, where M is the number of distinct features, and $\frac{m}{M}$ is the overall sparsity. As m is typically small and the preprocessing is simply embedding lookup and rescaling, the complexity of this part is negligible.

For the ARM-Module, the $K \cdot o$ exponential neurons of Equation 3 can be computed in $O(Komn_e)$; the parameter size of value/query vectors is $O(Kon_e)$; and the computation complexity of the bilinear attention alignment for all the m input embeddings is $O(Komn_e^2)$ during training, which can be reduced to $O(Komn_e)$ after training by precomputing the alignment of query vectors with respective attention weight matrices. To further reduce complexity for certain applications, the shared attention weight matrix in Equation 5 can be removed, resulting in single-head ARM-Net of complexity $O(Komn_e)$ throughout. For the prediction module, the complexity of the non-linear feature interaction function ϕ_{MLP} of Equation 7 is $O(n_w)$. Therefore, the overall parameter size and computational complexity for processing each input is $O(mn_e + n_w)$ and $O(Komn_e + n_w)$ respectively during inference, which is linear to the number of attribute fields and thus is efficient and scalable.

4 EXPERIMENTS

In this section, we evaluate the effectiveness, efficiency and interpretability of ARMOR. Figure 4 illustrates the overview of the experimental studies on ARMOR using five real-world datasets.

4.1 Experimental Setup

4.1.1 Datasets and Applications. We evaluate ARMOR with five real-world relational datasets on representative domains, namely app recommendation (Frappe[1]), movie recommendation (MovieLens[2]), click-through rate prediction (Avazu[3], Criteo[4]) and healthcare (Diabetes130[5]). We summarize the statistics of the datasets in Table 1 with the hyperparameters searched for ARM-Net.

Frappe is a real-world app recommendation dataset, which contains a context-aware app usage log and consists of 96,203 tuples by

[1] https://www.baltrunas.info/research-menu/frappe
[2] https://grouplens.org/datasets/movielens/
[3] https://www.kaggle.com/c/avazu-ctr-prediction
[4] https://labs.criteo.com/2014/02/kaggle-display-advertising-challenge-dataset/
[5] https://archive.ics.uci.edu/ml/datasets

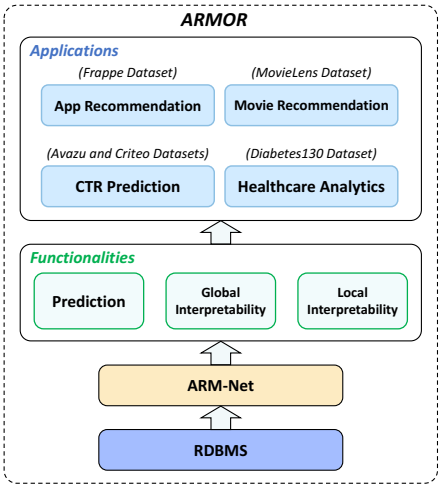

Figure 4: Predictive analytics using ARMOR.

957 users for 4,082 apps used in different contexts. Frappe samples two negative tuples for every one positive app usage log, leading to 288,609 tuples in total. The learning task is to predict the app usage given the usage context, which includes 10 semantic attribute fields, such as the previous app usage count, weather, time, location, etc, with 5,382 different numerical/categorical embedding vectors.

MovieLens contains the user tagging records of movies collected over various periods of time. In this paper, we focus on personalized tag recommendation by preparing the dataset into the (user, movie, tag) format. There are 2,006,859 tuples in total, and each tuple is a triplet of 3 categorical attribute fields of user ID, movie ID and tag, with 90,445 different categorical embedding vectors.

Avazu is a publicly accessible click-through rate (CTR) dataset provided by the mobile advertising platform Avazu. In online advertising, CTR is an important metric for evaluating advertisement performance. This dataset provides 11 days of data from Avazu for building models to predict whether a mobile ad will be clicked. The dataset contains 40,428,967 tuples and 22 attribute fields with 1,544,250 different numerical/categorical embedding vectors such as app and device information. The dataset is split sequentially for training, validation and testing.

Criteo is also a widely benchmarked CTR dataset, which contains attribute values and click feedback for millions of display ads. Display advertising is of significant commercial value and one of the primary machine learning models deployed in the real world. The learning task is to predict whether the user will click on a given ad under the page context. The Criteo dataset comprises 45,840,617 tuples of 39 attribute fields with 2,086,936 different numerical/categorical embedding vectors, including 13 numerical attribute fields (mostly counts) and 26 categorical attribute fields.

Diabetes130 collects 10 years (1999-2008) clinical diabetes encounters at 130 US hospitals. This dataset is presented for the analysis of a large clinical database by examining the patterns of historical diabetes care, which might lead to improvements in providing safe and personalized healthcare for patients. There are 101,766 encounters, each of which corresponds to a unique patient diagnosed with diabetes, and the learning task is the inpatient readmission prediction. The dataset comprises 43 attribute fields with 369 different numerical/categorical embedding vectors, mainly including patient demographics and illness severity, such as gender, age, race, discharge disposition, primary diagnosis, etc.

Table 1: Dataset statistics and best ARM-Net configurations.

Dataset	Tuples	Fields	Features	ARM-Net Hyperparameters
Frappe	288,609	10	5,382	$K = 8, o = 32, \alpha = 2.0$
MovieLens	2,006,859	3	90,445	$K = 1, o = 16, \alpha = 2.0$
Avazu	40,428,967	22	1,544,250	$K = 1, o = 32, \alpha = 1.5$
Criteo	45,302,405	39	2,086,936	$K = 4, o = 64, \alpha = 2.0$
Diabetes130	101,766	43	369	$K = 1, o = 32, \alpha = 1.7$

4.1.2 Baseline Methods. We compare ARM-Net with five categories of feature interaction modeling baselines: (1) Linear Regression (LR) that linearly aggregates input attributes with their respective importance weights without considering feature interaction; (2) models capturing second-order feature interactions, namely FM [44], AFM [59]; (3) models capturing higher-order feature interactions, namely HOFM [8], DCN [57], CIN [33], and AFN [13]; (4) neural networks based methods, namely DNN, and graph neural networks GCN [29] and GAT [55]. (5) models that are an ensemble of explicit cross feature modeling and implicit feature interaction modeling by DNNs, namely Wide&Deep [12], KPNN [43], NFM [21], DeepFM [19], DCN+ [57], xDeepFM [33] and AFN+ [13]. We briefly introduce these baseline methods as follows.

- **LR** takes the raw field features as the input for prediction, which simply aggregates these features with respective weights.
- **FM** [44] explicitly models second-order feature interactions with the factorization technique for efficiency.
- **AFM** [59] enhances FM by capturing the relative importance of second-order cross features with attention dynamically.
- **HOFM** [8] is a generalization of FM, which models higher-order feature interactions explicitly.
- **DCN** [57] captures cross features by computing the feature cross of the input feature embeddings.
- **CIN** [33] models higher-order feature interactions by performing the compressed interaction with input embeddings iteratively.
- **AFN** [13] models feature interactions of arbitrary orders with logarithm neurons.
- **DNN** captures fine-grained nonlinear feature interaction implicitly with a multilayer perceptron.
- **GCN** [29] considers attribute fields as graph nodes and captures their interactions with neighboring nodes via graph convolution.
- **GAT** [55] also considers attribute fields as nodes and models their interaction with neighboring nodes with attention dynamically.
- **Wide&Deep** [12] is an ensemble of LR and a DNN.
- **KPNN** [43] captures feature interactions by computing kernel product between input embeddings and is enhanced with a DNN.
- **NFM** [21] aggregates the element-wise product of all input pairs, and the bi-interaction features are enhanced with a DNN.
- **DeepFM** [19] is an ensemble model of FM and a DNN.
- **DCN+** [57] is an ensemble model of DCN and a DNN.
- **xDeepFM** [33] is an ensemble model of CIN and a DNN.
- **AFN+** [13] is an ensemble model of AFN and a DNN.

Table 2: Overall prediction performance with the same training settings.

Model Class	Model	Frappe		MovieLens		Avazu		Criteo		Diabetes130	
		AUC	Param	AUC	Param	AUC	Param	AUC	Param	AUC	Param
First-Order	LR	0.9336	5.4K	0.9215	90K	0.6900	1.5M	0.7741	2.1M	0.6701	370
Second-Order	FM	0.9709	5.4K	0.9384	90K	0.6797	1.5M	0.7663	2.1M	0.6594	370
	AFM	0.9665	5.7K	0.9473	91K	0.6857	1.6M	0.7847	2.1M	0.6774	7.6K
Higher-Order	HOFM	0.9778	170K	0.9435	2.9M	0.6919	18M	0.7788	107M	0.6714	11K
	DCN	0.9583	56K	0.9401	510	0.7460	3.3K	0.7959	6.6K	0.6765	2.2K
	CIN	0.9766	111K	0.9416	153K	0.6859	5.2M	0.7904	4.2M	0.6776	23K
	AFN	0.9779	3.1M	0.9470	242K	0.7456	3.3M	0.8061	7.8M	0.6778	306K
	ARM-Net	**0.9786**	867K	**0.9550**	140K	**0.7651**	147K	**0.8086**	1.5M	**0.6853**	102K
NN-based	DNN	**0.9787**	122K	**0.9540**	101K	0.7513	126K	**0.8082**	449K	0.6753	130K
	GCN	0.9732	1.6M	0.9404	365K	0.7506	964K	0.7984	2.2M	0.6828	4.0M
	GAT	0.9744	404K	0.9420	821K	**0.7525**	302K	0.8047	2.2M	**0.6846**	1.3M
Ensemble	Wide&Deep	0.9762	127K	0.9477	192K	0.6893	1.7M	0.7913	2.5M	0.6626	130K
	KPNN	0.9787	140K	0.9546	102K	0.7514	195K	0.8089	893K	0.6794	582K
	NFM	0.9745	100K	0.9214	186K	0.6874	1.6M	0.7833	2.3M	0.6695	4.6K
	DeepFM	0.9773	127K	0.9481	192K	0.6891	1.7M	0.7899	2.5M	0.6683	131K
	DCN+	0.9786	213K	0.9553	192K	0.7487	168K	0.8079	703K	0.6844	227K
	xDeepFM	0.9775	538K	0.9481	215K	0.6913	1.8M	0.7917	5.0M	0.6659	55M
	AFN+	0.9790	365K	0.9563	343K	0.7524	3.0M	0.8074	8.0M	0.6825	741K
	ARM-Net+	**0.9800**	263K	**0.9592**	217K	**0.7656**	339K	**0.8090**	1.3M	**0.6871**	1.7M

4.1.3 Evaluation Metrics. We evaluate the effectiveness of ARMOR with the metrics of AUC (the area under the ROC curve, higher is better), and Logloss (cross entropy, lower is better). For both AUC and Logloss, an improvement at 0.001 level is considered to be significant on the adopted benchmark datasets [13, 19]. We split the dataset in 8:1:1 for training, validation and testing respectively, and report the mean values of the evaluation metrics from five independent runs with early stopping on the validation set.

4.1.4 Hyperparameter Settings. For a fair comparison, we evaluate the effectiveness of all the models with the same training settings by fixing the embedding size to 10 consistently across datasets and sharing the best searched configurations of DNN for all ensemble models. Particularly, we obtain the best DNN depth from 1~8 and width from 100~800. The number of GCN/GAT layers are best of $\{1, 2, \ldots, 8\}$; the interaction orders for HOFM, DCN, CIN, DCN+ and xDeepFM are searched in 1~8, and the number of features/neurons for GCN, GAT, CIN, xDeepFM and AFN are searched in $\{10, 25, 50, 100, \ldots, 1600\}$.

For ARM-Net in particular, the sparsity α is searched in 1.0~3.0; and the number of attention heads K and exponential neurons per head o are grid searched with the constraint $K \cdot o \leq 1024$. We perform sensitivity analysis on these key hyperparameters in Section 4.3 and report the results of ARM-Net and the best searched baseline models in Table 2.

4.1.5 Implementation Details.

Training Details. We adopt Adam [28] optimizer with a learning rate searched in 0.1~1e-3 and generally a batch size of 4096 for all the models. In particular, we adopt a batch size of 1024 for the smaller dataset Diabetes130, and evaluate the larger dataset Avazu every 1000 training steps.

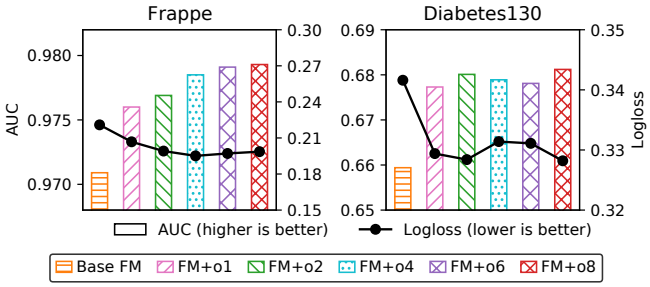

Figure 5: The effectiveness of cross features captured by ARM-Net in enhancing FM (0 to 8 features respectively).

Experimental Environment. The experiments are conducted in a server with Xeon(R) Silver 4114 CPU @ 2.2GHz (10 cores), 256G memory and GeForce RTX 2080 Ti. Models are implemented in PyTorch 1.6.0 with CUDA 10.2.

4.2 Effectiveness

Explicit Interaction Modeling with Single Models. We first compare ARM-Net with single baseline models that capture first-order, second-order and higher-order cross features explicitly. In Table 2, we summarize the overall experimental results measured in AUC with respective parameter sizes used during inference. Based on these results, we have the following findings.

First, ARM-Net consistently outperforms explicit interaction modeling baselines in terms of AUC. The better predictive performance confirms the effectiveness of ARM-Net across datasets and

Table 3: Training and inference efficiency of ARM-Net.

Dataset	Attribute Field	Training Throughput CPU	Training Throughput GPU	Inference Throughput CPU	Inference Throughput GPU	Avg GPU Speedup
MovieLens	3	5,454	131,864	8,005	189,460	23.92x
Frappe	10	2,102	72,612	2,718	91,572	34.12x
Avazu	22	1,065	41,313	1,284	48,055	38.11x
Criteo	39	661	24,717	781	26,543	34.93x
Diabetes130	43	581	23,547	746	23,615	36.10x

domains, including app recommendation (Frappe), movie tagging (MovieLens), click-through rate prediction (Avazu and Criteo) and medical readmission (Diabetes130). Second, we find that higher-order models, e.g., HOFM and CIN, generally achieve better prediction performance than lower-order ones such as LR and FM, which validates that higher-order cross features are indeed essential for prediction, and the absence of which greatly reduces the modeling capacity. Third, both AFN [13] and ARM-Net significantly outperform baseline models of a fixed order, which verifies the efficacy of modeling feature interaction of arbitrary orders in an adaptive and data-driven manner. Lastly, ARM-Net obtains noticeably higher AUC than the generally best-performing baseline model AFN.

The better performance of ARM-Net can be mainly ascribed to the exponential neurons and the gated attention mechanism. Specifically, the restriction of positive inputs for logarithmic transformation in AFN limits its representation, while ARM-Net circumvents this problem by modeling feature interactions in the exponential space instead. Further, instead of modeling interactions statically as in AFN, the multi-head gated attention of ARM-Net selectively filter noisy features and generates the interaction weights accounting for the characteristics of each input instance dynamically. Therefore, ARM-Net can capture more effective cross features for better prediction performance on a per-input basis Due to such runtime flexibility, ARM-Net is also more parameter efficient. As shown in Table 1, the best ARM-Net takes only dozens to a few hundreds of exponential neurons for different scale datasets, while the best AFN generally takes over a thousand neurons to obtain its best results, e.g., 32 as compared with 1600 for ARM-Net and AFN respectively on the large dataset Avazu.

NN-based Models and Ensemble Models. The results of NN-based models and DNN ensemble models are shown in Table 2. From these results, we can summarize the following key findings. (1) Despite not explicitly modeling feature interactions, the best NN-based models generally achieve strong prediction performance against other single model baselines. Particularly, the attention-based graph network GAT obtains noticeably higher AUC than other single models on Avazu and Diabetes130. However, their performance is not as consistent as ARM-Net, which varies greatly across different datasets, e.g., GAT performs much worse than DNN and ARM-Net on Frappe and MovieLens. (2) Model ensembles with a DNN significantly improve their respective predictive performance. This can be observed consistently throughout the baseline models, e.g., DCN+, xDeepFM and AFN+, which suggests that the nonlinear interactions captured by DNNs are complementary to explicitly captured interactions. (3) ARM-Net achieves comparable performance with DNN, and ARM-Net+ further improve the performance considerably, which obtains the overall best performance across all the benchmark datasets. In a nutshell, these results

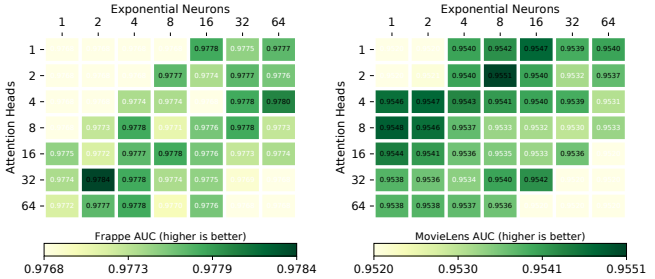

Figure 6: Sensitivity analysis of the number of attention heads and exponential neurons per head ($\alpha = 1.7$).

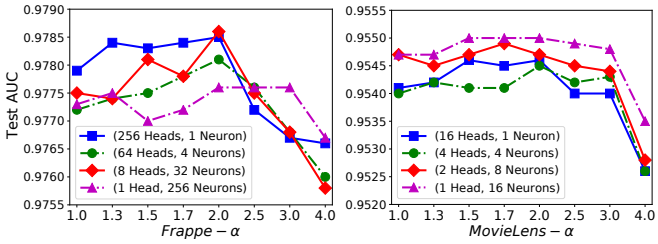

Figure 7: The impact of varying the sparsity α on the prediction performance of different $K \cdot o$ configurations.

further confirm the effectiveness of ARM-Net in modeling feature interactions of arbitrary orders in a selective and dynamic manner.

4.3 Efficiency and Ablation Studies

Efficiency. We further evaluate the training and inference efficiency of ARM-Net on the adopted benchmark datasets of different attribute fields size (m) in Table 3, which shows the *training/inference throughput* (the number of training/inference tuples per second). We consistently adopt $K = 4$, $o = 64$ and $n_e = 10$ for the benchmark ARM-Net, and train the model on one CPU or GPU. The mini-batch size is set to 16,384 consistently for all datasets to saturate the computation of the GPU.

From Table 3, we can observe that ARM-Net is rather efficient in both training and inference, whose throughputs are high and decrease linearly with the size of attribute fields m. This is in line with our analysis in Section 3.4 that the computational complexity of ARM-Net scales linearly to m. Further, we find that GPU can considerably speed up both training and inference, with a ratio from 23.92x to 38.11x for the benchmark ARM-Net. The speedup suggests that the efficiency of ARM-Net can be further improved by adopting more powerful hardware or more computational resources.

Enhancing FM with Exponential Neurons. To further evaluate the effectiveness of the cross features captured by the exponential neurons, we collect the feature interaction terms in ARM-Net and augment the learning of the factorization machine model (FM [44]) with these interaction features on top of the FM feature embeddings. Figure 5 shows the results measured in AUC and Logloss of the baseline FM model and FMs enhanced with different numbers of exponential neurons on Frappe and Diabetes130.

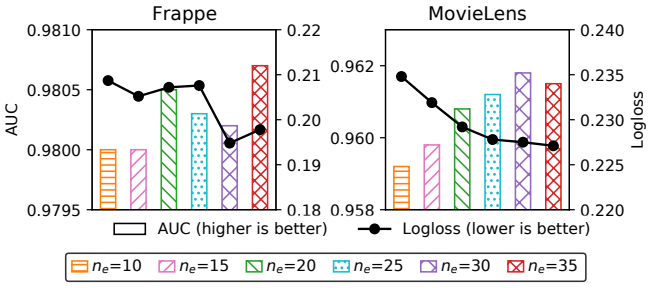

Figure 8: The impact of increasing the input embedding size on the prediction performance of ARM-Net+.

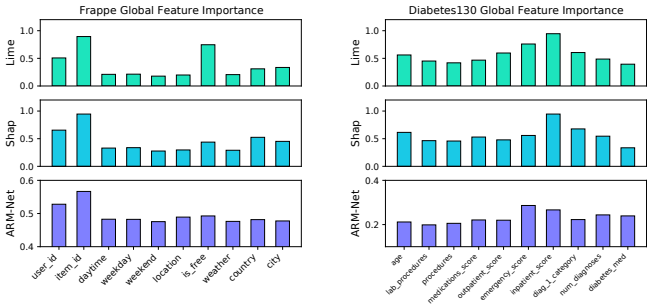

Figure 9: Global feature attribution with Lime, Shape and ARM-Net respectively on Frappe and Diabetes130.

We can observe that the interaction features dynamically captured with the exponential neurons consistently improve the prediction performance of FM by a large margin, which indicates that the cross features captured by ARM-Net are orthogonal to the FM features, and the integration leads to better interaction modeling. Remarkably, the improvement is significant even with only one exponential neuron, which enhances the AUC from 0.9709 of the Base FM to 0.9760 of FM+o1. Further, the prediction performance increases as we integrate more cross features encoded by exponential neurons. These findings also corroborate the modeling efficiency with the gated attention mechanism, which greatly reduces redundancy thanks to the runtime interaction order determination.

Multi-head Attention and Sparsity. As discussed in Section 3.4 and Section 4.1.4, K, o, and α are the key hyperparameters of ARM-Net. Specifically, K and o are the numbers of attention heads and exponential neurons per head respectively, and α controls the sparsity of α-entmax for the sparse feature selection in the multi-head attention alignment. We then evaluate these hyperparameters on Frappe and Diabetes130 and show the sensitivity analysis of K and o in Figure 6 and the impact of sparsity α on different configurations of $K \cdot o$ in Figure 7.

In Figure 6, we can notice that ARM-Net obtains relatively stable and high prediction performance under different configurations of $K \cdot o$. In particular, AUC scores achieved by ARM-Net on MovieLens are consistently higher than 0.9470 by the best-performing baseline single model AFN [13]. Further, we can also find that simply increasing K or o may not necessarily lead to higher prediction performance. This finding suggests that modeling from a moderate number of representation spaces with the multi-head attention mechanism is beneficial, and sharing the linear attention alignment weight within each attention head also facilitates learning and meanwhile, greatly reduces the parameter sizes.

Based on the results of varying sparsity on different $K \cdot o$ configurations in Figure 7, we find that a moderate sparsity α consistently leads to better prediction performance. Specifically, with a sparsity of 2.0 and 1.7 for Frappe and MovieLens respectively, ARM-Net achieve noticeably higher AUC than the corresponding architectures with dense softmax, namely $\alpha = 1.0$, across different configurations. These results support our contention that the proposed sparse attention mechanism can help filter noisy features dynamically for more effective interaction modeling.

Enhancing ARM-Net+ with a Larger Embedding Size. For a fair comparison, all the results reported in Table 2 consistently adopt an embedding size of 10. To evaluate the impact of the embedding size n_e on the prediction performance of ARM-Net+, we adopt the best ARM-Net+ and show the results of varying n_e in Figure 8 measured in AUC and Logloss. We can find that increasing the embedding size can further improve the prediction performance. Particularly, the AUC increases from 0.9800 to 0.9807 on Frappe and from 0.9592 to 0.9615 on MovieLens when adopting a larger embedding size of 35. The results suggest that by embedding the input into a larger latent space, we can further increase the model capacity of ARM-Net+.

4.4 Interpretability

Interpretability Settings. We demonstrate the interpretability results of ARMOR in two representative domains, namely the app usage prediction on Frappe and the diabetes readmission prediction on Diabetes130. The learning task on Frappe is to predict the app usage status given the use context. The context includes 10 attribute fields, *{user_id, item_id, daytime, weekday, weekend, location, is_free, weather, country, city}*, which together characterize the usage patterns of mobile end-users; For Diabetes130, the learning task is to predict the probability of readmission by analyzing factors related to readmission and other outcomes of patients with diabetes. There are 43 attribute fields for the prediction, and we show the 10 most important ones for illustration. The explanations of the attribute fields for both datasets are publicly available [5, 50], with which the interpretability results produced by ARMOR can be validated.

For both datasets, we first show the *global feature importance* of respective attribute fields obtained by aggregating the value vectors of exponential neurons as discussed in Section 3.4, and compare the global feature attribution of ARM-Net with two widely adopted interpretation approaches Lime [46] and Shap [34], which are input perturbation interpretation methods based on linear regression and game theory respectively to identify the feature importance of the models to be interpreted. Specifically, the interpretation results of Lime and Shap for Frappe and Diabetes130 is based on the best single-model baseline DNN and GAT [55] respectively, and the global feature importance given by both methods are obtained by the aggregation of local feature attribution of all instances of the test dataset. We then show the top interaction terms captured by ARM-Net with corresponding *frequency* and *orders*, which denote the average occurrence count per instance[6] and the number of features

[6]Note that there are $K \cdot o$ exponential neurons in ARM-Net, and each neuron captures a specific interaction term given the input instance dynamically.

Table 4: Top Global Interaction Terms for Frappe.

Frequency	Orders	Interaction Term
3.71	3	(weekday, location, is_free)
3.52	3	(user_id, item_id, is_free)
3.37	3	(item_id, weekend, is_free)
3.36	3	(item_id, is_free, city)
3.32	3	(user_id, weekend, is_free)
3.24	3	(user_id, is_free, city)
3.22	2	(item_id, is_free)
3.00	3	(user_id, item_id, weather)

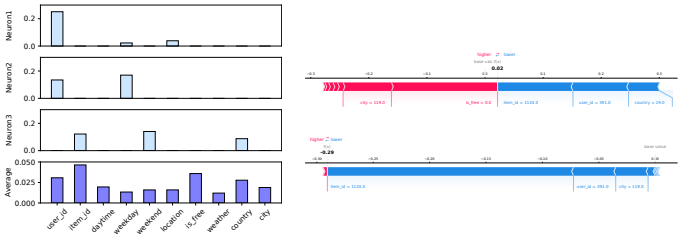

Figure 10: Local feature attribution with ARM-Net (left) and local feature importance weights given by Lime (right top) and Shap (right bottom) on Frappe.

Table 5: Top Global Interaction Terms for Diabetes130.

Frequency	Orders	Interaction Term
3.75	1	(inpatient_score)
2.41	1	(diag_1_category)
2.39	2	(A1Cresult, glimepiride)
2.10	2	(nateglinide, glyburide_metformin)
1.87	1	(num_diagnoses)
1.65	3	(metformin, nateglinide, glyburide_metformin)
1.45	2	(num_diagnoses, diabetes_med)
1.36	2	(inpatient_score, diabetes_med)

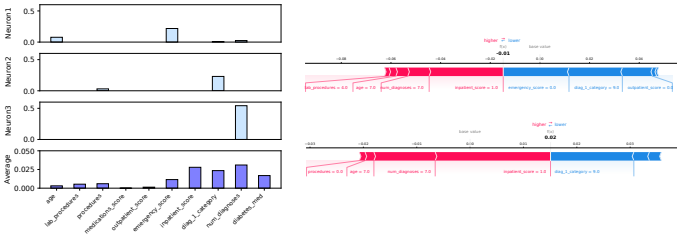

Figure 11: Local feature attribution with ARM-Net (left) and local feature importance weights given by Lime (right top) and Shap (right bottom) on Diabetes130.

captured for each interaction term respectively. We also illustrate local interpretation by showing feature interaction weights assigned by ARM-Module via aggregation, and again compare the local feature attribution results of ARM-Net with Lime and Shap.

Global Interpretability. We illustrate the *global feature attribution* in Figure 9 and summarize the top interaction terms captured by ARM-Net in Table 4 and Table 5 for the two datasets respectively.

From Figure 9, we can observe that the most important features identified by ARM-Net for Frappe are *{user_id, item_id, is_free}*. The global focus on these attributes are reasonable since *user_id* and *item_id* identify the user and item and are the two primary features used in learning tasks such as collaborative filtering, and *is_free* indicates whether the user pays for the app, which is highly correlated with the preference of the user for the app [5]. Likewise, the features of the highest importance identified by ARM-Net for Diabetes130 include *{emergency_score, inpatient_score, num_diagnoses}*, which is in line with the coefficients of attribute fields estimated by the logistic regression model in [50]. We can also notice that the global feature importance provided by ARM-Net is consistent with the two general interpretation approaches, namely Lime and Shap. However, we note that the interpretation results provided by ARM-Net are relatively more reliable since ARM-Net inherently supports global feature attribution and its modeling process is more transparent, whereas Lime and Shap are typically adopted as a medium to interpret other "black box" models via approximation.

From the top global interaction terms for Frappe in Table 4, we can find that firstly, the most frequently modeled attribute fields for the interaction terms include *use_id*, *item_id* and *is_free*, which is consistent with the global feature importance in Figure 9. Secondly, these interaction terms occur quite frequently in the interaction modeling, e.g., the frequency of the interaction term *(weekday, location, is_free)*, *(item_id, is_free, city)* and *(item_id, is_free)* are 3.71, 3.36 and 3.22 respectively, which indicates that these cross features (with different interaction weights) are used multiple times on average for each instance (note that there are $K \cdot o$ interaction terms during inference). Thirdly, the orders of the top interaction terms are mostly 2 and 3, which suggests that identifying the proper set of attributes for interaction modeling is necessary, and capturing cross features by enumerating all possible feature combinations is inefficient and ineffective, which may simply introduce noise.

From the top global interaction terms for Diabetes130 in Table 5, we can observe that the most frequently modeled attribute fields in the interaction terms are quite diversified, indicating that different exponential neurons indeed capture diverse cross features, which is more parameter efficient in modeling feature interactions. Further, the orders of the top interaction terms are less than 3, and there are many first-order terms, which indicates that for some datasets such as Diabetes130, modeling cross features of high interaction orders may not be necessary.

Local Interpretability. The local feature attribution of ARM-Net for a representative input instance of Frappe is illustrated in Figure 10, which shows the interaction weights of three representative exponential neurons and the average weights of all neurons. We can notice that different exponential neurons capture distinct cross features selectively in a sparse manner. For example, Neuron3 captures feature interaction term *(item_id, weekend, country)*, which indicates that for this specific instance, Neuron3 is responsive to these three attributes. Further, the aggregated interaction weights for this instance show that *item_id*, *is_free* and *user_id* are the three most discriminative attributes, which is consistent with the global interpretation results in Figure 9. We also illustrate the local feature attribution by Lime [46] and Shap [34] in Figure 10. We can notice that although both Lime and Shap identify *item_id*, *user_id* and *city* as the three most important features as ARM-Net, Lime also assigns large importance weight to other features, namely *is_free* and *country*, which indicates that external interpretation approaches

may not be consistent and reliable as they are only approximating the models to be interpreted from different perspectives.

Similar local feature attribution results for Diabetes130 can be found in Figure 11. We can find that different exponential neurons lay emphasis on distinct cross features. Specifically, Neuron1 and Neuron2 are more attentive to *emergency_score* and *diag_1_category* respectively, and Neuron3 are more focused on *num_diagnoses*. Further, for this specific diabetes patient, the last five features, i.e., *emergency_score*, *inpatient_score*, *diag_1_category*, *num_diagnoses* and *diabetes_med* are the most informative attributes for the readmission prediction. With such local interpretation, ARMOR can support more personalized analytics and management.

5 RELATED WORK

Feature Interaction Modeling. Cross features explicitly model feature interactions among attribute fields by multiplying corresponding constituent features, which proves to be essential for the predictive analytics of various applications, e.g., app recommendation [12] and click prediction [48]. Many existing works [12, 48] capture cross features in an implicit manner with DNNs. However, modeling multiplicative interaction implicitly with DNNs requires a substantial number of hidden units [2, 6, 25], which makes the modeling process inefficient and less interpretable [13, 18, 61].

Alternatively, many models [8, 26, 33, 43, 44, 57, 59] propose to capture cross features explicitly, which generally obtains better prediction performance. Among these studies, [26, 44, 59] capture second-order feature interactions, and [8] models higher order interactions feature interactions within a predefined maximum order. Many explicit interaction modeling studies further integrate a DNN to enhance the modeling capacity as DNNs are powerful at capturing fine-grained non-linear interactions. Representative studies include: (i) NFM [21] with bi-interaction pooling, i.e., an element-wise product between input feature embeddings, (ii) DeepFM [19] as an ensemble model of FM and DNN, (iii) DCN and its DNN ensemble model with feature cross operations on input feature embeddings, (iv) PNN [43] with both inner and outer pairwise products of input feature embeddings, and (v) CIN and its DNN ensemble model xDeepFM [33] with compressed interaction of input feature embeddings. Recently, AFN [13] propose to model cross features of arbitrary orders with logarithmic neurons, which however has the inherent limitation of only positive input due to logarithmic transformation and meanwhile lacks runtime flexibility. ARM-Net instead models feature interactions with exponential neurons adaptively with gated attention, which is more effective, interpretable and parameter-efficient.

Interpretability. With the increasingly dominant role played by machine learning models, there is a growing demand for model transparency and interpretability [17, 18], which can help debug the learning models and contribute to the verification and improvement of these models. In addition, an interpretable model also improves human understanding of many domains and engenders trust in the analytical results [17, 39].

A simple yet effective approach acting as either global or local interpretability is feature attribution, which produces the feature importance for input instance in terms of the weight and magnitudes of the features being used [17, 18, 46, 49], e.g., attributing a medical diagnosis to each lab test input. There exist general interpretation methods in this line of research. Notably, Shapley value [34, 49] assesses the importance of each feature in model predictions based on game theory. LIME [46] builds a linear model to locally approximate the model by input perturbation, which provides model-agnostic local explanations. Grad-CAM [47] provides a visual explanation based on gradient-weighted class activation mapping for CNN-based models to highlight local regions.

There are also domain-specific interpretation methods with integrated domain expertise. For instance, in healthcare analytics and finance, deep models are increasingly being adopted to achieve high predictive performance [42, 62–64]; however, such critical and high-stakes applications stress the need for interpretability. Particularly, the attention mechanism [4] is widely adopted to facilitate the interpretability of deep models by visualizing the attention weights. With the attention mechanism integrated into the model design, many studies [14, 36, 61] manage to achieve interpretable healthcare analytics. Specifically, Dipole [36] supports the visit-level interpretation in the diagnosis prediction with three attention mechanisms. RETAIN [14] and TRACER [61] can support both the visit-level and the feature-level interpretation. However, one inherent limitation of most existing methods is that their interpretability is built upon single input features, while disregarding feature interactions that are essential for relational analytics.

6 CONCLUSION

Existing DNNs, while producing super-human results for certain application domains, have not been shown to produce meaningful results when applied to structured data. To fill this gap, we present a learning model tailored for structured data, called ARM-Net, and a lightweight framework ARMOR based on ARM-Net for relational data analytics, which is accurate, efficient and more interpretable. The key idea is to model attribute dependencies and correlations selectively and dynamically, through cross features, derived by first transforming input features into exponential space, and then determining interaction weights and the interaction order adaptively for each cross feature. To model cross features of arbitrary orders dynamically and filter noisy features selectively, we propose a novel gated attention mechanism to generate interaction weights given the input tuple. Consequently, ARM-Net can identify the most informative cross features in an input-aware manner for more accurate prediction and better interpretability during inference. An extensive experimental study on real-world datasets confirms that our ARM-Net consistently achieves superior prediction performance compared to existing models, and ARMOR facilitates global interpretability and local instance-aware interpretability.

7 ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive comments. This research is supported by Singapore Ministry of Education Academic Research Fund Tier 3 under MOE's official grant number MOE2017-T3-1-007. H. V. Jagadish was supported in part by NSF grants 1741022 and 1934565. Meihui Zhang's work is supported by National Natural Science Foundation of China (62050099).

REFERENCES

[1] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. 2016. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*. 173–182.

[2] Alexandr Andoni, Rina Panigrahy, Gregory Valiant, and Li Zhang. 2014. Learning Polynomials with Neural Networks. In *Proceedings of the 31th International Conference on Machine Learning, ICML*.

[3] Sercan Ömer Arik and Tomas Pfister. 2019. TabNet: Attentive Interpretable Tabular Learning. *CoRR* abs/1908.07442 (2019).

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*.

[5] Linas Baltrunas, Karen Church, Alexandros Karatzoglou, and Nuria Oliver. 2015. Frappe: Understanding the Usage and Perception of Mobile App Recommendations In-The-Wild. *arXiv preprint arXiv:1505.03014* (2015).

[6] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H. Chi. 2018. Latent Cross: Making Use of Context in Recurrent Recommender Systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM*. ACM.

[7] Or Biran and Courtenay Cotton. 2017. Explanation and justification in machine learning: A survey. In *IJCAI-17 workshop on explainable AI (XAI)*, Vol. 8. 8–13.

[8] Mathieu Blondel, Akinori Fujino, Naonori Ueda, and Masakazu Ishihata. 2016. Higher-order factorization machines. In *Advances in Neural Information Processing Systems*. 3351–3359.

[9] Rajesh Bordawekar and Oded Shmueli. 2017. Using Word Embedding to Enable Semantic Queries in Relational Databases. In *Proceedings of the 1st Workshop on Data Management for End-to-End Machine Learning, DEEM@SIGMOD 2017, Chicago, IL, USA*. ACM, 5:1–5:4.

[10] John S Bridle. 1990. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing*. 227–236.

[11] Lingjiao Chen, Arun Kumar, Jeffrey F. Naughton, and Jignesh M. Patel. 2017. Towards Linear Algebra over Normalized Data. *Proceedings of VLDB Endowment.* 10, 11 (2017), 1214–1225.

[12] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 7–10.

[13] Weiyu Cheng, Yanyan Shen, and Linpeng Huang. 2020. Adaptive Factorization Network: Learning Adaptive-Order Feature Interactions. In *34th AAAI Conference on Artificial Intelligence*.

[14] Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. 2016. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In *Advances in Neural Information Processing Systems*. 3504–3512.

[15] Milan Cvitkovic. 2020. Supervised Learning on Relational Databases with Graph Neural Networks. *CoRR* abs/2002.02046 (2020).

[16] George Cybenko. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems* 2, 4 (1989), 303–314.

[17] Krishna Gade, Sahin Cem Geyik, Krishnaram Kenthapadi, Varun Mithal, and Ankur Taly. 2019. Explainable AI in industry. In *Proceedings of International Conference on Knowledge Discovery & Data Mining, SIGKDD*. 3203–3204.

[18] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2019. A Survey of Methods for Explaining Black Box Models. *Comput. Surveys* 51, 5 (2019).

[19] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI*. 1725–1731.

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

[21] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 355–364.

[22] J. Wesley Hines. 1996. A logarithmic neural network architecture for unbounded non-linear function approximation. In *Proceedings of International Conference on Neural Networks (ICNN'96)*. IEEE, 1245–1250.

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

[24] Jie Hu, Li Shen, and Gang Sun. 2018. Squeeze-and-Excitation Networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018*.

[25] Siddhant M. Jayakumar, Wojciech M. Czarnecki, Jacob Menick, Jonathan Schwarz, Jack W. Rae, Simon Osindero, Yee Whye Teh, Tim Harley, and Razvan Pascanu. 2020. Multiplicative Interactions and Where to Find Them. In *8th International Conference on Learning Representations, ICLR*.

[26] Yu-Chin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware Factorization Machines for CTR Prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems*.

[27] Mahmoud Abo Khamis, Hung Q. Ngo, XuanLong Nguyen, Dan Olteanu, and Maximilian Schleich. 2020. Learning Models over Relational Data Using Sparse Tensors and Functional Dependencies. *ACM Transactions on Database Systems.* 45, 2 (2020), 7:1–7:66.

[28] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR*.

[29] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR*. OpenReview.net.

[30] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The Efficient Transformer. In *8th International Conference on Learning Representations, ICLR*.

[31] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent Convolutional Neural Networks for Text Classification. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence,*. AAAI, 2267–2273.

[32] Side Li, Lingjiao Chen, and Arun Kumar. 2019. Enabling and Optimizing Non-linear Feature Interactions in Factorized Linear Algebra. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD*. ACM.

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

[34] Scott M. Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems, USA*. 4765–4774.

[35] Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP*. 1412–1421.

[36] Fenglong Ma, Radha Chitta, Jing Zhou, Quanzeng You, Tong Sun, and Jing Gao. 2017. Dipole: Diagnosis prediction in healthcare via attention-based bidirectional recurrent neural networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 1903–1911.

[37] André F. T. Martins and Ramón Fernández Astudillo. 2016. From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016*, Vol. 48. 1614–1623.

[38] Tim Miller. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence* 267 (2019), 1–38.

[39] Christoph Molnar. 2019. *Interpretable Machine Learning*. https://christophm.github.io/interpretable-ml-book/. https://christophm.github.io/interpretable-ml-book/.

[40] Milos Nikolic, Haozhe Zhang, Ahmet Kara, and Dan Olteanu. 2020. F-IVM: Learning over Fast-Evolving Relational Data. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD*. ACM, 2773–2776.

[41] Ben Peters, Vlad Niculae, and André F. T. Martins. 2019. Sparse Sequence-to-Sequence Models. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019*. 1504–1519.

[42] Sanjay Purushotham, Chuizheng Meng, Zhengping Che, and Yan Liu. 2018. Benchmarking deep learning models on large healthcare datasets. *Journal of biomedical informatics* 83 (2018), 112–134.

[43] Yanru Qu, Bohui Fang, Weinan Zhang, Ruiming Tang, Minzhe Niu, Huifeng Guo, Yong Yu, and Xiuqiang He. 2018. Product-based neural networks for user response prediction over multi-field categorical data. *ACM Transactions on Information Systems (TOIS)* 37, 1 (2018), 1–35.

[44] Steffen Rendle. 2010. Factorization Machines. In *ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010*.

[45] Steffen Rendle. 2013. Scaling Factorization Machines to Relational Data. *Proceedings of VLDB Endowment.* 6, 5 (2013), 337–348.

[46] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD*. ACM, 1135–1144.

[47] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*. 618–626.

[48] Ying Shan, T Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. 2016. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 255–262.

[49] Lloyd S Shapley. 1953. A value for n-person games. *Contributions to the Theory of Games* 2, 28 (1953), 307–317.

[50] Beata Strack, Jonathan P DeShazo, Chris Gennings, Juan L Olmo, Sebastian Ventura, Krzysztof J Cios, and John N Clore. 2014. Impact of HbA1c measurement on hospital readmission rates: analysis of 70,000 clinical database patient records. *BioMed research international* 2014 (2014).

[51] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic Attribution for Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML*, Vol. 70. 3319–3328.

[52] Constantino Tsallis. 1988. Possible generalization of Boltzmann-Gibbs statistics. In *Journal of Statistical Physics*. 52:479–487.

[53] Kush R. Varshney and Homa Alemzadeh. 2017. On the Safety of Machine Learning: Cyber-Physical Systems, Decision Sciences, and Data Products. *Big Data* 5, 3 (2017), 246–255.

[54] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*. 5998–6008.

[55] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *6th International Conference on Learning Representations, ICLR*.

[56] Martin J Wainwright, Michael I Jordan, et al. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning* 1, 1–2 (2008), 1–305.

[57] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & Cross Network for Ad Click Predictions. In *Proceedings of the SIGKDD'17*. ACM.

[58] Wei Wang, Meihui Zhang, Gang Chen, H. V. Jagadish, Beng Chin Ooi, and Kian-Lee Tan. 2016. Database Meets Deep Learning: Challenges and Opportunities. *SIGMOD Rec.* 45, 2 (2016), 17–22.

[59] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional Factorization Machines: Learning the Weight of Feature Interactions via Attention Networks. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI*.

[60] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. 2019. Adversarial Examples: Attacks and Defenses for Deep Learning. *IEEE Transactions on Neural Networks and Learning Systems*. 30, 9 (2019), 2805–2824.

[61] Kaiping Zheng, Shaofeng Cai, Horng Ruey Chua, Wei Wang, Kee Yuan Ngiam, and Beng Chin Ooi. 2020. TRACER: A Framework for Facilitating Accurate and Interpretable Analytics for High Stakes Applications. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD*. ACM.

[62] Kaiping Zheng, Gang Chen, Melanie Herschel, Kee Yuan Ngiam, Beng Chin Ooi, and Jinyang Gao. 2021. PACE: Learning Effective Task Decomposition for Human-in-the-loop Healthcare Delivery. In *Proceedings of the 2021 ACM SIGMOD International Conference on Management of Data*.

[63] Kaiping Zheng, Jinyang Gao, Kee Yuan Ngiam, Beng Chin Ooi, and Wei Luen James Yip. 2017. Resolving the Bias in Electronic Medical Records. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Halifax, NS, Canada) *(KDD '17)*. 2171–2180.

[64] Kaiping Zheng, Wei Wang, Jinyang Gao, Kee Yuan Ngiam, Beng Chin Ooi, and Wei Luen James Yip. 2017. Capturing Feature-Level Irregularity in Disease Progression Modeling. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (Singapore, Singapore) *(CIKM '17)*. 1579–1588.