MemoNet:Memorizing Representations of All Cross Features Efficiently via Multi-Hash Codebook Network for CTR Prediction

Pengtao Zhang
Sina Weibo
pengtao1@staff.weibo.com

Junlin Zhang
Sina Weibo
junlin6@staff.weibo.com

Abstract

New findings in natural language processing(NLP) demonstrate that the strong memorization capability contributes a lot to the success of large language models.This inspires us to explicitly bring an independent memory mechanism into CTR ranking model to learn and memorize all cross features' representations.In this paper,we propose multi-Hash Codebook NETwork(HCNet) as the memory mechanism for efficiently learning and memorizing representations of all cross features in CTR tasks.HCNet uses multi-hash codebook as the main memory place and the whole memory procedure consists of three phases: multi-hash addressing,memory restoring and feature shrinking.HCNet can be regarded as a general module and can be incorporated into any current deep CTR model.We also propose a new CTR model named MemoNet which combines HCNet with a DNN backbone.Extensive experimental results on three public datasets show that MemoNet reaches superior performance over state-of-the-art approaches and validate the effectiveness of HCNet as a strong memory module.Besides, MemoNet shows the prominent feature of big models in NLP,which means we can enlarge the size of codebook in HCNet to sustainably obtain performance gains.Our work demonstrates the importance and feasibility of learning and memorizing representations of all cross features ,which sheds light on a new promising research direction.

CCS Concepts: • **Information systems** → **Recommender systems**.

Keywords: Recommender System;Click-Through Rate

ACM Reference Format:
Pengtao Zhang and Junlin Zhang. 2022. MemoNet:Memorizing Representations of All Cross Features Efficiently via Multi-Hash Codebook Network for CTR Prediction. In *Proceedings of (PrePrint).* ACM, New York, NY, USA, 13 pages. https://doi.org/XXXXXXX.XXXXXXX

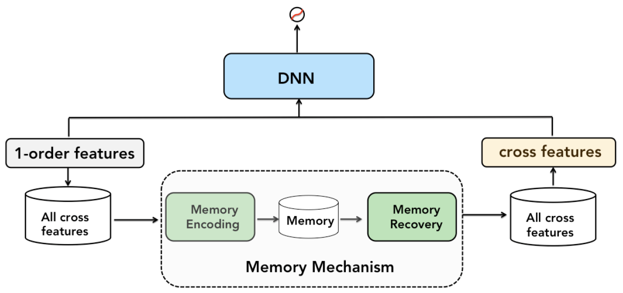

Figure 1. Learning and memorizing representations of all cross features through independent memory mechanism

1 Introduction

Click-through rate (CTR) prediction plays important role in personalized advertising and recommender system, and it aims to predict the probability of a user clicking on the recommended items. Many models[8, 9, 21, 24, 29, 31, 33]have been proposed to resolve this problem.However,researches[13, 16, 30] show that increasing model parameter of modern deep CTR models has little influence on model performance,which is contrary to phenomenon shown in lots of large language models[2, 23, 34] in NLP.

The past several years have witnessed the great success of big models[2, 6, 22, 23, 34] in natural language processing with ever larger model size. Notable models include the 345 million parameter BERT[6] , the 1.5 billion parameter GPT-2[22] , the 11 billion parameter T5[23] and the 175 billion parameter GPT-3[2].Given ample training data,these big models show a great advantage that model performance can be boosted constantly by increasing the number of model parameters,which is regarded as the most prominent feature of big models.

In order to explain the success of large language models,many recent works[3, 27, 32, 37] discuss the memorization and generalization ability of these models.Some works[11, 19, 28] show that big models in NLP have great capabilities to memorize factual world knowledge in their vast amount of parameters during training, which is a crucial component of their improved generalization.

These new findings in NLP inspire us to ask the following two questions: (1)What will happen if we introduce a specially designed memory mechanism into CTR model to efficiently learn and memorize important knowledge in CTR tasks such as cross features?(2)Can we boost model performance constantly by enlarging the size of this memory just as the phenomenon shown by big models in NLP ?

To tackle these problems, we propose to mimic the strong memory capability of big models in NLP by explicitly introducing an independent memory mechanism into CTR ranking model to learn and memorize all cross features' representations,as shown in Figure 1.The memory mechanism is composed of three modules:the 'memory module' is the main place to store all cross features' representations and 'memory encoding module' is used to encode the representation, while 'memory recovery module' tries to retrieval each cross feature's representation for use.

We argue in this paper that an independent memory mechanism for learning and memorizing representations of all cross features reflects the thought of clear division of labour for each component in modern DNN ranking model.The memory system is in charge of memorizing knowledges in all cross features and the DNN part is only responsible for generalization of all features,including 1-order,2-order and high order features,which greatly reduces DNN part's learning difficulty and boosts model performance.It is worth noting that, when comparing with other SOTA CTR approaches which explicitly model feature interactions through sub-network such as xDeepFM[16], DCN[29],DCN v2[30] and FiBiNet[13],learning and memorizing all cross features through an independent memory mechanism instead of modeling cross features by a special DNN sub-network shows a new and different research direction.

Though Wide & Deep Learning[5] also jointly trains wide linear models and deep neural networks to combine the benefits of memorization and generalization.There are big differences between our proposed approach and Wide & Deep Learning[5].First, the wide part of Wide & Deep Learning[5] is a linear model without complex representation ability while we proposed a wholly new memory system with strong memory capability.Second, Wide & Deep Learning[5] only models small proportion of feature interactions and expertise feature engineering is still needed on the input to the wide part , which means that the cross-product transformation also requires to be manually designed.However, we propose to learn and memorize representations of **all the cross features** with limited resource **automatically**,and which is a hard problem to resolve.

To make this idea more specific, in this paper we propose multi-Hash Codebook NETwork(HCNet) as the abovementioned memory mechanism for learning and memorizing representations of all cross features in CTR tasks.In HCNet,multi-hash codebook is used as the place to store cross feature's representation,which consists of large amount of codewords as the basic memory unit. Ideally,codeword accurately represents cross feature if we can allocate one codeword for each cross feature.However,it's impracticable because of the combinatorial explosion problem of high order features. For example,suppose we have 1 million 1-order features,which is just nearly **1**% of the feature number in real world CTR scene,there are about $\mathbf{10}^{12}$ 2-order cross features,letting alone 3-order or even higher order feature interaction.Therefore,it's impossible to allocate a codeword for each cross feature. On the contrary, we want to use as little codeword as possible while exactly memorizing each cross feature's information at the same time in this work.Therefore,it's unavoidable for one codeword to store large amount of cross feature's information.That is to say,if we just use 1 million codewords to record all $\mathbf{10}^{12}$ 2-order cross features,one codeword need to encode nearly one million different cross feature's information in theory,and we aim to restore the exact representation for each specific cross feature from it. The whole procedure looks like information encryption and decryption ,that's the reason why we name the basic memory unit 'codeword' and the main memory place 'codebook'.

For better representation, we use multi-hash functions to segment the representation of cross feature into m chunks and each chunk represents part of the complete representation. In order to efficiently store and retrieval representation of any feature interaction,the entire procedure of HCNet could be divided into three phases: multi-hash addressing,memory restoring and feature shrinking. Multi-hash addressing phase aims to obtain all cross features in an input instance and locate addresses of all chunks in codebook for each cross feature.Memory restoring stage recovers the exact representation for any cross feature through 'linear memory restoring' or 'attentive memory restoring' method, and feature shrinking phase aims to compress representations of many cross features in input instance into a thin embedding layer to reduce the model parameters.

An analog with human brain's memory is that 'multi-hash codebook' is the memory region in brain and HCNet is the whole memorizing mechanism of brain for storing and retrieving information through memory region.Note here multi-hash codebook is a rather concise memory,though we can design codebook into a more complex structure such as hierarchy form.Simple structure is adopted in this paper to verify the feasibility of learning and memorizing representations of all feature interactions with limited resource and we will leave more complex structure as future work.

As a general module to efficiently memorize cross features,HCNet can be incorporated into any current deep CTR models.DNN model is the simplest deep CTR model and we use it as a backbone to combine it with HCNet to boost the model performance,which is called 'MemoNet' in this paper.

Extensive experiments on three real-world public datasets show that MemoNet leads to significant performance improvements compared to state-of-the-art CTR methods.Besides, MemoNet shows the prominent feature of big model,that is to say, model can absorb more knowledge from data when we increase model parameter through memorizing. This demonstrates we can just enlarge the codeword number of codebook in HCNet to sustainably obtain performance gains.As far as we know, MemoNet is the first CTR model with this kind of ability.

The main contributions of our works are concluded as follows:

- To the best of our knowledge,we are the first to propose and verify the feasibility of learning and memorizing all cross features with limited resource,which sheds light on a new promising research direction.
- We propose a novel HCNet as a specific memory mechanism and it's a general module to be incorporated into any current deep CTR model to greatly boost model performance.
- We plug HCNet into DNN models to form 'MemoNet',which is the first CTR model to show the prominent feature of big model as large language models in NLP show .
- The significant improvements on three real-world benchmarks confirm the effectiveness of HCNet in CTR ranking systems.

2 Related Works

2.1 Shallow CTR Methods

Factorization Machines (FMs) [24] and Field-aware Factorization Machines (FFMs) [14] are two of the most successful CTR models. Other linear FM-based models are proposed, such as CoFM [12], FwFM [21] and importance-aware FM [20]. However, these models show limited effectiveness in modeling high-order latent patterns or learning quality feature representations.

2.2 Deep CTR Methods Modeling Feature Interactions

Many deep learning based CTR models[8, 13, 16, 26, 29–31], have been proposed in recent years and how to effectively model feature interactions is key to them.A multilayer perceptron (MLP) can implicitly model any order feature interactions for recommendation[9, 33, 36], but implicit modeling is ineffective in learning feature interactions[1, 25].

While early stage deep CTR models process feature interactions in implicit way[9, 33, 36], most recent works explicitly model pairwise or high order feature interactions by subnetwork[8, 13, 16, 26, 29, 30]. For example,DeepFM[8] utilizes FM to capture 2-order feature interactions.Deep & Cross Network (DCN)[29] and DCN V2 [30] efficiently capture feature interactions of bounded degrees in an explicit fashion. Similarly, xDeepFM [16] models high-order feature interactions by proposing a novel Compressed Interaction Network (CIN). AutoInt[26] designs a multi-head self-attentive neural network with residual connections to explicitly model the feature interactions.FiBiNet [13] and FiBiNet++[35] introduce bilinear feature interaction function while MaskNet [31] proposes instance-guided mask to capture high order cross features.Many reported results [13, 16, 29, 30] show that explicitly modeling high-order interaction outperforms approaches in implicit ways. This indicates the importance of high order feature interaction in CTR tasks.

However, majority of real-life CTR ranking systems still need humanly designed cross features for better performance,which demonstrates inefficiency of deep networks modeling feature interactions explicitly.An independent memory mechanism for memorizing representation of cross features helps MLP focus on feature generalization instead of modeling feature interaction,which make learning easier. We will verify this in experiment part of this paper that memorizing cross features is a much efficient approach to learn feature interaction compared with these methods,which also helps free people from onerous feature engineering work.

2.3 Deep CTR Methods Mining Feature Interactions

Another line of research on pairwise and high order features aims to mine some useful feature interactions in multiple stages: small proportion of feature interactions are mined by manual feature selection or AutoML in the first stage and are introduced into CTR model as new features in the second stage. For example,Wide & Deep Learning[5] jointly trains wide linear models and deep neural networks to combine the benefits of memorization and generalization.However, expertise feature engineering is still needed on the input to the wide part. To alleviate manual efforts in feature engineering, AutoFIS[18] and AutoGroup[17] use AutoML to seek small proportion of useful high-order feature interactions to train CTR model in two or three stages.

There are two shortcomings in these multi-stage mining methods:First,most real world ranking systems support online learning ,and one difficulty of these approaches is that they can't instantly capture new beneficial feature interactions.In addition,these models miss large amount of long-tail useful cross features,which have great impact on model performance as a whole.Different from existing studies, our proposed approach can be deployed in online learning systems and capture long-tail beneficial cross features efficiently.

3 Preliminaries

Deep learning models are widely used in industrial recommendation systems,such as Wide&Deep[5], DCN[29],DeepFM[8] and FiBiNet[13]. Among them, DNN model is almost the simplest one and is always used as a sub-component in most current DNN ranking systems[5, 8, 13, 16, 29, 30]. It contains three components:feature embedding,MLP and prediction layer.In this section, we introduce the DNN model and the optimization objective of CTR prediction task.

3.1 Feature Embeddding

As all we know, features in CTR tasks usually can be segregated into the following two groups:

1. Categorical features. This type of feature is common and the one-hot representation may produce very sparse features. We map one-hot representation to dense, low-dimensional embedding vectors suitable for complex transformation. We can obtain feature embedding $\mathbf{v}_i$ for one-hot vector $\mathbf{x}_i$ via:
$$\mathbf{v}_i = \mathbf{x}_i \mathbf{W}_e \tag{1}$$
where $\mathbf{W}_e \in \mathbb{R}^{d \times n}$ is the embedding matrix of n features and d is the dimension of field embedding.
2. Numerical features. We map the feature field into an embedding vector as follows:
$$\mathbf{v}_i = \mathbf{x}_i \mathbf{e}_i \tag{2}$$
where $\mathbf{e}_i \in \mathbb{R}^{1 \times d}$ is an embedding vector for field i with size d, and $\mathbf{x}_i$ is a scalar value which means the actual value of that numerical feature. Some big numerical values will dominate the parameter update procedure during model training and we normalized $\mathbf{x}_i$ into [0,1] via min-max normalization technique before multiplying it into $\mathbf{e}_i$ to avoid this:
$$\mathbf{x}_i = \frac{\mathbf{x}_i - \mathbf{x}_{min}}{\mathbf{x}_{max} - \mathbf{x}_{min}} \tag{3}$$

3.2 MLP and Prediction Layer

To learn high-order feature interactions, multiple feed-forward layers are stacked on the concatenation of dense features represented as $\mathbf{H}_0 = concat\{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_f\}$. Then,the feed forward process of MLP is:

$$\mathbf{H}_l = ReLU(\mathbf{W}_l \mathbf{H}_{l-1} + \beta_l) \tag{4}$$

where l is the depth and ReLU is the activation function. $\mathbf{W}_l, \beta_l, \mathbf{H}_l$ are weighting matrix, bias and output of the l-th layer.

The prediction layer is put on the last layer of multiple feed-forward networks,and the model' s output is:

$$\hat{y} = \delta(\mathbf{w}_0 + \sum_{i=1}^{n} \mathbf{w}_i \mathbf{x}_i) \tag{5}$$

where $\hat{y} \in (0, 1)$ is the predicted value of CTR, $\delta(\cdot)$ is the sigmoid function, n is the size of feed-forward layer, $\mathbf{x}_i$ is the bit value of feed-forward layer and $\mathbf{w}_i$ is the learned weight for each bit value.

3.3 Optimization Objective

For binary classifications, the loss function of CTR prediction is the log loss:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \tag{6}$$

where N is the total number of training instances, y_i is the ground truth of i-th instance and $\hat{y}_i$ is the predicted CTR.

4 Methodology

In this section, we first describe multi-hash codebook in detail ,which is the place to memorize representations of feature interactions. Then, we take the 2-order pairwise interaction as example to show the technical details of the proposed multi-Hash Codebook NETwork(HCNet) for memorizing cross features efficiently.Next,we show how to memorize higher order feature interactions by HCNet.After that,we demonstrate our proposed MemoNet which embeds HCNet into DNN backbone for CTR prediction.Finally, we discuss how to identify key interaction fields to reduce number of cross features.

4.1 Multi-Hash Codebook

Our works aims to learn and memorize all feature interactions with limited memory resource and codebook is the place where we store cross feature's representation.Codebook consists of large amount of codewords which is the basic memory unit.Formally,we define codebook as parameter matrix $\mathbf{C} \in \mathbb{R}^{n \times l}$ with size of **n** rows and **l** columns to be learned during model training.Each row of matrix **C** is one codeword which is a vector with size of **l**.We leverage the row number of matrix **C** as the index or address of the corresponding codeword in the codebook.

For a feature interaction $\mathbf{x}^{(i,j)}$ coming from cross of feature $\mathbf{x}_i$ and $\mathbf{x}_j$, we can allocate a random codeword with address $\mathbf{a}^{(i,j)}$ for it through a general hash function $\mathbf{H}(\cdot)$ which projects input to a large number as follows:

$$\mathbf{a}^{(i,j)} = \mathbf{H}(\mathbf{x}^{(\mathbf{i},\mathbf{j})}) \quad MOD \quad \|\mathbf{C}\| \tag{7}$$

where $\|\mathbf{C}\| = n$ means size of codebook and MOD is a remainder operator. We can use codeword embedding to memorize the cross feature $\mathbf{x}^{(i,j)}$.

Because of the combinatory explosion problems for cross features, 100 thousands or even 1 million cross features may be projected randomly in one codeword,which leads codeword embedding to a mixed representation and it's hard to distinguish them from each other.To alleviate this problem,we leverage multiple hash functions to encode cross

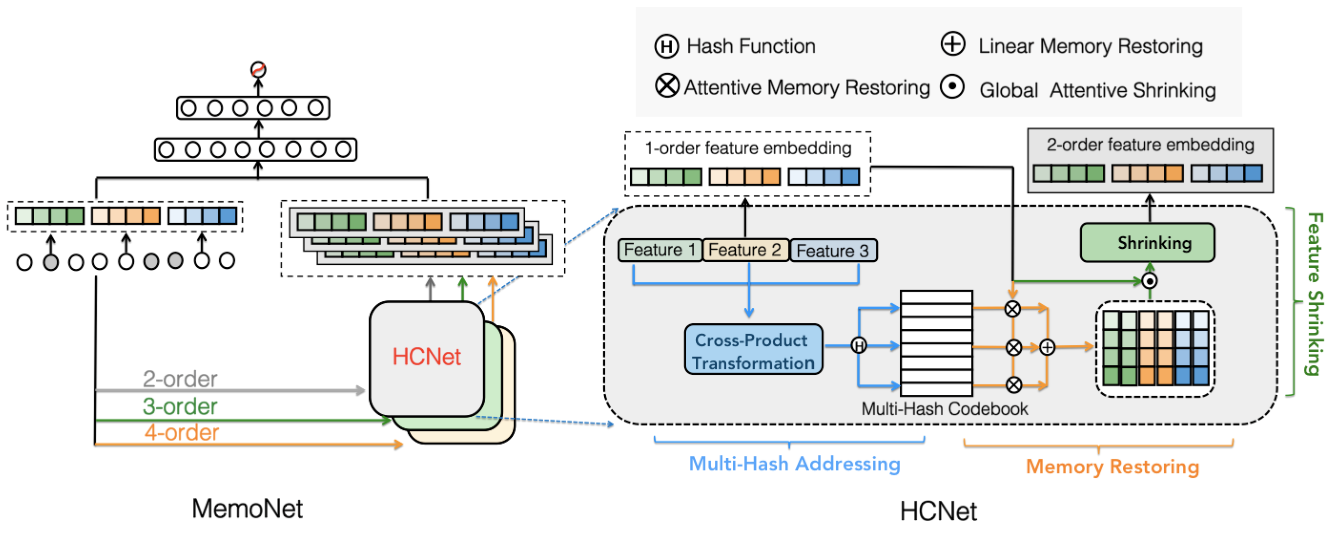

Figure 2. The overall architectures of MemoNet and HCNet

features as follows:

$$\mathbf{A}^{(i,j)} = \left\{\mathbf{H}_t(\mathbf{x}^{(i,j)}) \quad MOD \quad \|\mathbf{C}\|\right\}_{t=1}^{m} \tag{8}$$

where $H_t(\cdot)$ denotes the $t-th$ hash function and $\mathbf{A}^{(i,j)}$ is an index set indicating m codeword addresses in codebook for feature $\mathbf{x}^{(i,j)}$.

The codeword embedding matrix $\mathbf{C}^{(i,j)} \in \mathbb{R}^{m\times l}$ of feature $\mathbf{x}^{(i,j)}$ can be extracted according to location index set $\mathbf{A}^{(i,j)}$ as follows:

$$\mathbf{C}^{(i,j)} = [\mathbf{C}_1^{(i,j)}, \mathbf{C}_2^{(i,j)}, ..., \mathbf{C}_m^{(i,j)}] \in \mathbb{R}^{m\times l} \tag{9}$$

where $\mathbf{C}_t^{(i,j)} \in \mathbb{R}^{1\times l}$ denotes the codeword embedding indicated by index produced by the $t-$th hash function $\mathbf{H}_t(\mathbf{x}^{(i,j)})$ and l is embedding size of codeword.We call the codebook with multiple hash functions 'multi-hash codebook' in this paper,which is the main memory component of HCNet.

Though codeword seems similar with bin or bucket[4] in form,there are two main differences:First,bin usually doesn't contain so large amount of collision items like codeword does,that will bring serious damage to model performance; Second,all items in one bin will share same representation in applications while we try to recover the exact representation of one specific item in our works,and that's the most challenging task.

It's obvious that 'multi-hash codebook' segments the representation of cross feature into m chunks and each chunk indicating by different hash function represents part of the complete information. This helps distinguish cross feature from each other in same codeword because the probability of sharing same representation exponentially decreases with the increase of hash function number.

Aiming to recover exact representation for each cross feature,multi-hash codebook is not enough because each chunk is still the mixed representation.We propose other approaches to resolve this problem and will describe them in detail in 'memory restoring' part of section 4.2.

4.2 HCNet

The overall framework of HCNet is shown in Figure 2.Note the above-mentioned 'multi-hash codebook' is also a component of HCNet,which is main memory place.In order to efficiently store and retrieval embedding of any feature interaction,the entire procedure of HCNet could be divided into three phases: multi-hash addressing,memory restoring and feature shrinking.

Multi-Hash Addressing. This stage is designed to obtain all cross features in an input instance and locate addresses of all chunks in codebook for each cross feature.Specifically,we first produce all cross features in input instance via cross product transformation.Suppose there are f fields in input instance, we obtain an new feature set $\mathcal{E}_2^{all}$ with size $f \times (f-1)$ after cross product transformation for this instance:

$$\mathcal{E}_2^{all} = \bigcup_{i=1}^{f} \mathcal{E}_2^i, \quad where \quad \mathcal{E}_2^i = \left\{\mathbf{x}^{(i,j)}\right\}_{1\leqslant j\leqslant f \quad \wedge \quad i\neq j} \tag{10}$$

Here $\mathbf{x}^{(i,j)}$ means a new 2-order feature by cross of feature $\mathbf{x}_i$ and $\mathbf{x}_j$,and $\mathcal{E}_2^i$ is the feature set with size $(f-1)$ which contains all cross features derived from $\mathbf{x}_i$.

Then,we produce a global unique id for each new feature $\mathbf{x}^{(i,j)}$ to distinguish them from each other. For feature $\mathbf{x}_i$ and $\mathbf{x}_j$,we concatenate two unique internal index numbers as an unique id for new feature $\mathbf{x}^{(i,j)}$ as follows:

$$\mathbf{id}_2^{(i,j)} = \mathbf{id}_2^{(j,i)} = concat[\mathbf{id}_i, \mathbf{id}_j] \tag{11}$$

where cross feature $\mathbf{x}^{(i,j)}$ and $\mathbf{x}^{(j,i)}$ share same id because the input order doesn't matter for CTR prediction.If $\mathbf{x}_i \in \mathbb{R}$

is a numerical feature,the unique $\mathbf{id}_i$ is formed by $\mathbf{id}_i = concat[field-id, \phi_k(\mathbf{x}_i)]$,where $field-id$ is internal index number of the field which $\mathbf{x}_i$ belongs to and $\phi_k(\cdot)$ is a truncation function keeping k decimal places for float number.We concatenate two numbers for the uniqueness of the feature id. We find size of k has influence on model performance and the optimal setting is $k = 5$.

After obtaining a unique $\mathbf{id}_2^{(i,j)}$ for a 2-order new feature $\mathbf{x}^{(i,j)}$,we leverage hash function $H(\cdot)$ to locate its codeword addresses in codebook as follows:

$$\mathbf{A}_2^{(i,j)} = \left\{\mathbf{H}_t(\mathbf{id}_2^{(i,j)}) \quad MOD \quad \|\mathbf{C}\|\right\}_{t=1}^{m} \tag{12}$$

In this way,we can locate the codeword embeddings matrix $\mathbf{C}^{(i,j)}$ for any pairwise feature $\mathbf{x}^{(i,j)}$.HCNet tunes the codeword embedding while training to learn the representation and readout the codeword embedding via multi-hash addressing in online reference stage.

Memory Restoring. Memory restoring stage aims to recover the exact representation for any cross feature.While 'multi-hash codebook' helps distinguish cross feature from each other in same codeword,it's not good enough because each chunk is still mixed representation. To better recover each cross feature's embedding,we propose two methods in this paper.

A straightforward way to recover feature $\mathbf{x}^{(i,j)}$'s accurate representation is first to flatten the two dimensional codeword embedding matrix $\mathbf{C}^{(i,j)} \in \mathbb{R}^{m\times l}$ to one dimensional vector $\mathbf{e}^{(i,j)}$ with size of $1\times ml$,and then project it into a new embedding space by a MLP layer:

$$\mathbf{v}^{(i,j)} = \phi_1(\mathbf{e}^{(i,j)}\mathbf{W}_1) \in \mathbb{R}^{1\times d} \tag{13}$$

where $\mathbf{W}_1 \in \mathbb{R}^{ml\times d}$ is parameter matrix of MLP layer and $\phi_1(\cdot)$ is an identity function without non-linear transformation.The new mapping based on chunks of codeword embedding extracts useful information encoded in chunks and aggregates an unique representation for cross feature.We find the non-linear transformation $\phi(\cdot)$ brings negative effect on model performance because too early complex mutual interaction among codeword embedding chunks weakens informative signals contained in each codeword. Cross feature's embedding can be restored in this way and we call this method "Linear Memory Restoring (LMR)".Note we set size of the recovered cross feature d,which is same with the size of 1-order feature embedding.

In order to better restore the memorized embedding for cross feature $\mathbf{x}^{(i,j)}$,another our proposed method uses 1-order embedding of feature $\mathbf{x}_i$ and $\mathbf{x}_j$ to guild the representation recovery.We think this will make the memorizing and learning easier. Therefore,we design an attention subnetwork based on the corresponding feature embedding $\mathbf{v}_i$ and $\mathbf{v}_j$.Specifically, we first project $\mathbf{Z} = concat[\mathbf{v}_i, \mathbf{v}_j] \in \mathbb{R}^{1\times 2d}$ as input to obtain attention mask $\mathbf{I}$ as follows:

$$\mathbf{I} = \phi_3\left(\phi_2\left(\mathbf{Z}\mathbf{W}_2\right)\mathbf{W}_3\right) \in \mathbb{R}^{1\times ml} \tag{14}$$

where $\mathbf{W}_2 \in \mathbb{R}^{2d\times s}$ and $\mathbf{W}_3 \in \mathbb{R}^{s\times ml}$ are learning matrix of two MLP layers, $\phi_2(\cdot)$ is $ReLu(\cdot)$ and $\phi_3(\cdot)$ is identity function.Then, attention mask $\mathbf{I}$ is reshaped into m groups:

$$\mathbf{I}^r = [\mathbf{I}_1, \mathbf{I}_2, ...\mathbf{I}_m] \in \mathbb{R}^{m\times l} \tag{15}$$

where $\mathbf{I}_i \in \mathbb{R}^{1\times l}$ is the corresponding attention map for i-th codeword embedding.We leverage attention matrix $\mathbf{I}^r$ to recover the embedding by filtering out its information from each codeword,which mixes many cross feature's information.The procedure is computed as follows:

$$\begin{aligned}\mathbf{C}_I^{(i,j)} &= \mathbf{I}^r \otimes \mathbf{C}^{(i,j)} \in \mathbb{R}^{m\times l} \\ &= [\mathbf{I}_1 \otimes \mathbf{C}_1^{(i,j)}, \mathbf{I}_2 \otimes \mathbf{C}_2^{(i,j)}, ..., \mathbf{I}_m \otimes \mathbf{C}_m^{(i,j)}]\end{aligned} \tag{16}$$

where $\otimes$ is an element-wise multiplication between two vectors. The following procedure is similar with LMR and we flatten the attentive matrix $\mathbf{C}_I^{(i,j)}$ and map it into a new embedding space via MLP according to formula 13 to better restore cross feature's embedding $\mathbf{v}^{(i,j)}$. We call this method "Attentive Memory Restoring(AMR)" in this paper.

We find AMR outperforms LMR when codeword number is relatively small while LMR performs better if we continue to increase codeword's number. This indicates 1-order feature embedding indeed helps recover cross feature's representation because small codeword number means much more mixed chunk representations in one codeword.We will discuss this in Section 5.3.

Feature Shrinking. In this section we describe our proposed feature shrinking approaches. Cross features should be compressed in this stage due to the following two reasons:First,large amount of cross feature's embedding brings noise because of its high sparse property. Second,there are $f\times(f-1)$ new 2-order cross features in an input instance and embedding layer will be too wide if we don't compress them,let along 3-order or even higher order feature interactions.

In this stage,we aim to compress cross features according to the following rule:Given a feature set $\mathcal{E}_2^i = \left\{\mathbf{x}^{(i,j)}\right\}_{1\leqslant j\leqslant f}$ which contains all 2-order cross features derived by feature $\mathbf{x}_i$, we leverage a weighted sum function $\delta(\cdot)$ to project all corresponding embedding into $\mathbf{v}_2^i \in \mathbb{R}^{1\times d}$,where $\mathbf{v}_2^i$ is a compressed feature embedding as representation of all useful cross features derived by $\mathbf{x}_i$.Formally, we have following function $\delta(\cdot)$:

$$\mathbf{v}_2^i = \sum_{j=1}^{f} \mathbf{a}_j\mathbf{v}^{(i,j)}, j \neq i \tag{17}$$

Therefore, we need another attentive function $\psi(\cdot)$ to evaluate the importance of each new feature derived by $\mathbf{x}_i$.

In this paper,we propose three approaches to produce attentive weights.One straightforward method is sum pooling as $\mathbf{v}_2^i = \sum_{j=1}^{f}\mathbf{v}^{(i,j)}$,which can be regarded as a special kind of attention function giving all cross features same weight.We call this method 'Sum Pooling Shrinking(SPS)'.

The second approach leverages all the 1-order feature embedding to help select useful cross features.We think 1-order feature is not so sparse as high order features and contains more information,that helps reduce noisy cross feature's negative effect. Formally, we first concatenate embedding of all 1-order features as $\mathbf{V} = concat[\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_f] \in \mathbb{R}^{1\times fd}$.Then,$\mathbf{V}$ is input into an attention network to produce vector-wise attention score for each pairwise feature in feature set $\mathcal{E}_2^{all}$ as follows:

$$\mathbf{R} = \phi_5\left(\phi_4\left(\mathbf{V}\mathbf{W}_4\right)\mathbf{W}_5\right) \in \mathbb{R}^{1\times f(f-1)} \tag{18}$$

where $\mathbf{W}_4 \in \mathbb{R}^{fd\times s}$ and $\mathbf{W}_5 \in \mathbb{R}^{s\times f(f-1)}$ are parameter matrix of two MLP layers, $\phi_4(\cdot)$ is $ReLu(\cdot)$,and $\phi_5(\cdot)$ is an identity function without non-linear transformation.Here $\mathbf{R} \in \mathbb{R}^{1\times f(f-1)}$ is the attention map with size of $f \times (f-1)$,which contains weight for each feature in set $\mathcal{E}_2^{all}$.Then,we can compress features according to formula 17 after obtaining $\mathbf{R}$. We call this approach 'Global Attentive Shrinking(GAS)' in this paper.

The third approach for HCNet to shrink cross features shares the same procedure of obtaining attention map $\mathbf{R} \in \mathbb{R}^{1\times f(f-1)}$ with GAS.However, for features in set $\mathcal{E}_2^i = \left\{\mathbf{x}^{(i,j)}\right\}$ which contains all 2-order cross features derived by feature $\mathbf{x}_i$, we use softmax normalization on their corresponding attention scores in $\mathbf{R}$ in order to amplify the impact of the most beneficial features in $\mathcal{E}_2^i = \left\{\mathbf{x}^{(i,j)}\right\}$ .Formally,we have:

$$\hat{\mathbf{a}}_\mathbf{j} = Softmax(\mathbf{a}_j) = \frac{\mathbf{e}^{\mathbf{a}_j/\tau}}{\sum_{c=1}^{f}\mathbf{e}^{\mathbf{a}_c}} \tag{19}$$

where $\tau = 0.9$ is temperature parameter and we find 0.9 is an optimal value. After attention score's normalization,features are compressed according to formula 17 and we name this approach 'Local Attentive Shrinking(LAS)'.

We adopt GAS as feature shrinking component in HCNet because our experiment shows GAS outperforms other two approaches and we will compare them in detail in Section 4.4.

Due to the feature shrinking,we can compress all 2-order features of an input instance from $1 \times f(f-1)d$ size layer into a thin embedding layer $\mathbf{V}_2$ with size of $1 \times fd$:

$$\mathbf{V}_2 = concat[\mathbf{v}_2^1, \mathbf{v}_2^2, ..., \mathbf{v}_2^f] \in \mathbb{R}^{1\times fd} \tag{20}$$

4.3 MemoNet

HCNet can be regarded as a general module to efficiently memorize pairwise or high order cross features.Therefore, we incorporate it into DNN model described in section 3 to boost its performance,which is the simplest deep CTR model.We call the HCNet with DNN backbone 'MemoNet' in this paper and the overall structure is shown in Figure 2.

As discussed in section 2.2,modern complex deep CTR models are inclined to design sub-network to capture cross features such as CIN in xDeepFM[16] and bi-linear interaction in FiBiNet[13].However, MemoNet use HCNet to memorize cross features with limited resources and the DNN part is only responsible for generalization of all features,including 1-order,2-order and high order features. This kind of clear division of labour for each component in MemoNet greatly reduces the learning difficulty and boosts model performance. Besides,as main memory in HCNet, size of multi-hash codebook can be gradually enlarged to store more useful knowledge in cross features and further boost model's performance just as big models in NLP show.We will verify these in experiment part of this paper.

Though MemoNet leverages DNN as backbone of the model, HCNet can be used as a general plug-in module in other CTR models. We explore the compatibility of our proposed HCNet and integrate DeepFM,xDeepFM,DCN v2 and MaskNet with HCNet in Section 5.5.

4.4 Memorizing High Order Features

We can learn and memorize representation of high order feature just as 2-order HCNet does,which also consists of three phases:multi-hash addressing,memory restoring and feature shrinking.We find model performance decreases if high order HCNet shares same codebook with 2-order HCNet.The possible reason is maybe that high order features faces much serious combination explosion problem and it needs new space to store the cross features.Therefore,we need to allocate new multi-hash codebook as main memory for high order HCNet.

Suppose we want to memorize the k-th order cross features by k-order HCNet.Formally, we first produce all k-order features in input instance through cross product transformation and obtain k-order feature set $\mathcal{E}_k^{all}$ for this instance:

$$\mathcal{E}_k^{all} = \bigcup_{i=1}^{f}\mathcal{E}_k^i, \quad where \quad \mathcal{E}_k^i = \left\{\mathbf{x}^{(i,j_1,j_2...j_{(k-1)})}\right\}_{1\leqslant j_t\leqslant f} \tag{21}$$

where $\mathbf{x}^{(i,j_1,j_2...j_{(k-1)})}$ denotes a new produced k-order feature and $\mathcal{E}_k^i$ is the k-order feature subset which contains all k-order cross features derived from feature $\mathbf{x}_i$.

Then,a global unique id for each new feature $\mathbf{x}^{(i,j_1,j_2...j_{(k-1)})}$ is produced by concatenating unique internal index numbers as follows:

$$\mathbf{id}_k^{(i,j_1,j_2...j_{(k-1)})} = concat[\mathbf{id}_i, \mathbf{id}_{j_1}, \mathbf{id}_{j_2}, ..., \mathbf{id}_{j_{(k-1)}}] \tag{22}$$

After obtaining a global unique id for each k-order feature, the following procedure is same with that in multi-hash addressing and memory restoring phase of 2-order HCNet.In feature shrinking phase,we also use GAS to compress cross features in set $\mathcal{E}_k^i$ to project all corresponding embedding into $\mathbf{v}_k^i \in \mathbb{R}^{1\times d}$,where $\mathbf{v}_k^i$ is a compressed feature embedding as representation of all useful k-order features derived by $\mathbf{x}_i$.Finally,we can compress all k-order features of an input

Table 1. Overall performance of different models(MC means million codewords and xH denotes hash function number= x)

		Avazu			**KDD12**			**Criteo**		
Model		AUC(%) ↑	Logloss ↓	Paras.	AUC(%) ↑	Logloss ↓	Paras.	AUC(%) ↑	Logloss ↓	Paras.
Shallow	FM	78.17	0.3809	79M	77.65	0.1583	60M	78.97	0.4607	11M
Mining	Wide&Deep	78.49	0.3788	79M	79.33	0.1542	60M	80.53	0.4462	11M
Modeling	DeepFM	78.64	0.3769	79M	79.40	0.1538	60M	80.58	0.4457	11M
	xDeepFM	78.88	0.3747	81M	79.51	0.1534	62M	80.64	0.4450	15M
	DCN	78.68	0.3767	78M	79.58	0.1531	55M	80.73	0.4441	10M
	FiBiNet	79.12	0.3730	87M	79.52	0.1533	61M	80.73	0.4441	17M
	AutoInt+	78.62	0.3758	78M	79.69	0.1529	55M	80.78	0.4438	10M
	MaskNet	78.94	0.3740	85M	79.89	0.1521	56M	81.07	0.4410	13M
	DCN v2	78.98	0.3738	81M	79.66	0.1530	55M	80.88	0.4430	11M
Backbone	DNN	78.67	0.3756	78M	79.54	0.1533	55M	80.73	0.4440	10M
Ours (Small/1MC)	MemoNet-L	**79.58/9H**	**0.3704**	128M	**80.60/7H**	**0.1508**	60M/0.5MC	81.28/3H	0.4390	20M
	Improve.	+0.91	-0.0052	+50M	+1.06	-0.0025	+5M	+0.55	-0.0050	+10M
	MemoNet-A	79.52/8H	0.3707	128M	80.52/8H	0.1516	60M/0.5MC	**81.30/3H**	**0.4387**	20M
	Improve.	+0.85	-0.0049	+50M	+0.98	-0.0017	+5M	+0.57	-0.0053	+10M
Ours (Big/10MC)	MemoNet-L	**79.87/9H**	**0.3684**	578M	-	-	-	**81.39/4H**	**0.4380**	110M
	Improve.	+1.20	-0.0072	+500M	-	-	-	+0.66	-0.0060	+100M
	MemoNet-A	79.73/9H	0.3693	578M	-	-	-	81.38/3H	0.4381	110M
	Improve.	+1.06	-0.0063	+500M	-	-	-	+0.65	-0.0059	+100M

instance into a thin embedding layer $\mathbf{V_k}$ as follows :

$$\mathbf{V_k} = concat[\mathbf{v}_k^1, \mathbf{v}_k^2, ..., \mathbf{v}_k^f] \in \mathbb{R}^{1\times fd} \tag{23}$$

4.5 Key Interaction Field

Given dataset,we find only small fraction of fields are important for feature interaction,which contribute majority of the performance gain in HCNet. We name this kind of field 'key interaction field'(KIF). Therefore,we can only use features in these KIFs to interact with any other feature in input instance,which will greatly reduce the combination number of cross feature in HCNet .In order to identify these KIFs,we propose to use score function $\mathbf{s}_{F_k}(\cdot)$ to rank field importance of feature interaction and the top scored fields can be regarded as KIFs.

In this paper,we propose two approaches to compute the field importance.A straightforward method is to rank field according to the feature number it contains.More features it contains ,more important it is. Formally,we have the following field importance score for th-**k** field:

$$\mathbf{s}_{F_k} = \|F_k\| \tag{24}$$

where $\|F_k\|$ means the number of features which belong to field F_k.We call this method 'feature number ranking'(FNR).

The second method is named 'field attention ranking'(FAR) because we can compute attention score for one field by accumulating the GAS attention score of features belonging to this field.The higher attention score one field has,more important it is.Specifically,after training HCNet,we can leverage a small validation dataset to find out which field is vital by accumulating attention score of the feature in one field.We can compute the field importance score $\mathbf{s}_{F_k}$ for th-**k** field as follows:

$$\mathbf{s}_{F_k} = \sum_{i=1}^{n}\sum_{j=1}^{f} \mathbf{a}^i_{(k,j)}, j \neq k \tag{25}$$

where n denotes the instance number of validation set ,f is field number and $\mathbf{a}^i_{(k,j)}$ is the attention score of th-(k, j) cross feature in th-i instance computed by GAS. We will show the effectiveness of the proposed methods to find out KIFs in Section 5.6.

5 Experimental Results

We conduct extensive experiments on three real-world benchmarks to comprehensively evaluate our proposed method.

5.1 Experiment Setup

Datasets. The following three datasets are used in our experiments:

1. **Avazu[1] Dataset:** The Avazu dataset consists of several days of ad click- through data which is ordered chronologically. For each click data, there are 23 fields which indicate elements of a single ad impression.
2. **KDD12[2] Dataset:** KDD12 dataset aims to predict the click-through rate of ads and the instances are derived from session logs of the Tencent proprietary search

[1]Avazu http://www.kaggle.com/c/avazu-ctr-prediction
[2]KDD12 https://www.kaggle.com/c/kddcup2012-track2

engine.There are 13 fields spanning from user id to ad position for a clicked data.

3. **Criteo[3] Dataset:** As a very famous public real world display ad dataset with each ad display information and corresponding user click feedback, Criteo data set is widely used in many CTR model evaluation. There are 26 anonymous categorical fields and 13 continuous feature fields in Criteo data set.

We remove the infrequent features and replace them with a same value "unknown", where threshold is set to 4,5,10 for Avazu,KDD12,Criteo respectively. We randomly split instances by 8:1:1 for training , validation and test while Table 2 lists the statistics of the evaluation datasets.

Table 2. Statistics of the Evaluation Datasets

Datasets	#Instances	#fields	#features
Avazu	40.4M	23	9.4M
KDD12	149.6M	13	54.6M
Criteo	45.8M	39	33.7M

Evaluation Metrics. AUC (Area Under ROC) and Logloss (binary cross-entropy loss) are used as the evaluation metrics in our experiments. These metric are very popular for binary classification tasks.

Models for Comparison. We compare the performance of the FM[24], Wide&Deep[5],DNN[36], DeepFM[8], DCN[29], AutoInt[26],DCN V2[30],xDeepFM[16],FiBiNet[13] and the MaskNet[31] models as baselines and all of which are discussed in Section 2. Results of some models such as the LR[10],FwFM[21],NFM[9] , AFM[33] and CCPM[7] are not presented in this paper, because more recent models like FiBiNET[13] and DCN-V2[30] have outperformed these methods significantly as experiments in FuxiCTR[38] shows.

Implementation Details. We implement all the models with Tensorflow in our experiments. For optimization method, we use the Adam[15] with a mini-batch size of 1024. We find different learning rate has great influence on some baseline's performance.Therefore, both 0.0001 and 0.001 learning rate are verified and the best results are reported as final performance for all baselines. We make the dimension of 1-order feature embedding for all models to be a fixed value of 10 for Criteo, 50 for Avazu and 10 for KDD12 dataset. For models with DNN part, the depth of hidden layers is set to 3, the number of neurons per layer is 400, all activation function are ReLU. In HCNet,unless otherwise specified,we use 2 hash functions and 1M codewords(0.5M codewords on KDD12 dataset) as our default setting and keep size of codeword embedding same with 1-order feature embedding. For other models, we take the optimal settings from the original papers.

[3]Criteo http://labs.criteo.com/downloads/download-terabyte-click-logs/

5.2 Performance Comparison

Table 1 shows the performance of different SOTA baselines and MemoNet. For MemoNet,two groups of experiment are designed:the first group is small MemoNet which has codebook with 1 million codeword(0.5 million codeword on KDD12 dataset which is optimal codebook size). The second group is big MemoNet containing codebook with 10 millions codeword.For experiments in each group,we fix other settings and tune the number of hash function for best performance model.

The experiments for MemoNet and the baseline models are repeated 5 times by changing the random seeds and the averaged results are reported for fair comparison.The best results are in bold, and the best baseline results are underlined. From the experimental results, we mainly have the following observations:

1. Among all the baselines,FiBiNet[13] is best performance model on Avazu dataset and MaskNet[31] outperforms other baselines on KDD12 and Criteo. However,experiments show that there is no dominant model performing best on all benchmarks,which indicates model performance depends on properties of dataset. Compared with DNN model,which is a strong baseline on all three datasets,only MaskNet[31] and DCN v2[30] stably outperform DNN on all benchmarks.In addition,AutoInt+[26] and xDeepFM[16] perform well on one or two datasets.
2. Compared with other baselines, two small MemoNets (MemoNet-L with linear memory restoring and MemoNet-A with attentive memory restoring approach) both achieve the best results on all benchmarks and outperform other SOTA models with a large margin.The best performance small MemoNet models show a huge superiority in performance and outperform DNN backbone by +0.91, +1.06, and +0.57 absolute value in terms of AUC on three datasets, respectively.This indicates that a specifically designed memorizing mechanism for cross features indeed make DNN part focus on feature generalization and reduce the learning difficulty.On the other hand,HCNet respectively contains 50M,5M and 10M parameters to memorize cross features on three datasets,which is applicable in most real life CTR scenes.The experimental results demonstrate the feasibility of memorizing all cross features with limited resources to greatly boost model performance.
3. As for the big MemoNet,which contains 500M and 100M parameters in HCNet on Avazu and Criteo datasets respectively, we find the model performance continues to increase compared with small MemoNet. Besides,best performance MemoNets outperform best

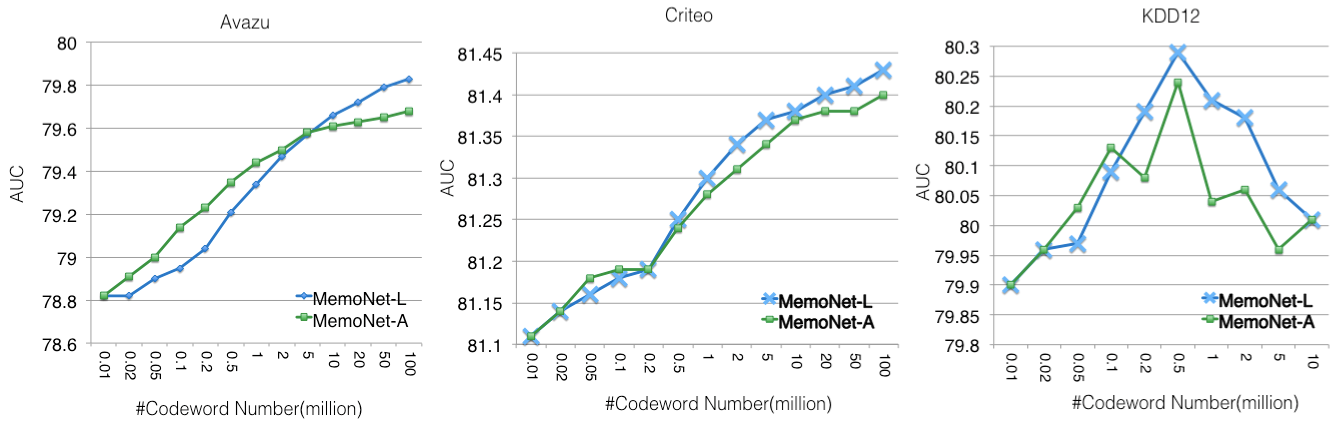

Figure 3. Effect of different codeword number in HCNet

baselines by +0.72, +0.71, and +0.32 absolute value in terms of AUC on three datasets, respectively.It is worth noting that, when comparing with best baseline, an improvement of AUC at 0.1 level is rather hard in real-life CTR tasks.

5.3 Effect of Codeword Number

In this section, we conduct extensive experiments on three datasets to explore the effects of various numbers of codeword in codebook of HCNet on performance,and show the different characteristic of MemoNet-L and MemoNet-A model.

We gradually increase codeword number from 0.01M to 100M on Avazu and Criteo(the max codeword number is set to 10M on kDD12 because 0.5M is optimal number) to watch the performance change. Figure 3 reports the experimental results in MemoNet-L and MemoNet-A and we have the following findings:

Finding 1: **MemoNet achieves big model's ability of absorbing more knowledge from data when we increase model parameter through memorizing.**we can see from Figure 3 that both model(MemoNet-L and MemoNet-A) 's performances continue to increase when we enlarge the codeword number on Avazu and Criteo datasets.The procedure is stopped when the max codeword number reaches 100M because of our limited hardware resource,and it seems the performance will further increase if we continue to enlarge codeword number. It demonstrates that MemoNet indeed obtains the big model's ability of absorbing more knowledge from data when we increase model parameter through memorizing.We argue that MemoNet is the first CTR model to have this kind of ability as big models in NLP show. However, the optimal codeword number is 0.5M on KDD12 dataset. This indicates KDD12 dataset has much less useful cross features compared with Avazu and Criteo, and small HCNet can effectively memorize all effective cross features with rather limited resource.If the dataset contains large amount of long-tail useful cross features such as Avazu, we can just enlarge the codeword number of codebook in HCNet to sustainably obtain performance gains.This denotes memorizing cross features efficiently just like HCNet does is a very promising new research direction for CTR prediction.

Finding 2: **MemoNet-A outperforms MemoNet-L when codebook is small.** It can be seen from Figure 3 that MemoNet-A performs better than MemoNet-L model when codeword number is relatively small and it will go into reverse once codeword number reaches specific number.On KDD12 and Criteo,the turning point number is 0.2M and it's 5M on Avazu dataset.This denotes the attentive memory restoring helps recover cross feature's representation because there are more cross features sharing same codeword embedding when codeword number is small.

5.4 Effect of Hash Function Number

This section studies the performance sensitivity when setting different numbers of hash function in HCNet,as shown in Figure 4 .We find the performance sensitivity of hash function number depends on datasets.Firstly, we observe that performance of MemoNet-L on criteo dataset fluctuates within limited range when changing hash function numbers.However, model performance on Avazu will increase continuously with the increase of hash function number.As for the model performance on KDD12,it increases continuously when the hash function number is less than 5 and begin to fluctuate after that.The possible reason is that Avazu dataset contains much more useful long-tail cross features compared with other two datasets.

5.5 Plugging HCNet into SOTA Models

As a concise and effective component, we argue that HCNet can be easily incorporated into any deep CTR model to boost the performance. To verify this, extensive experiments are conducted on four SOTA models: DeepFM,xDeepFM,DCN v2 and MaskNet.Table 3 show the results which demonstrate

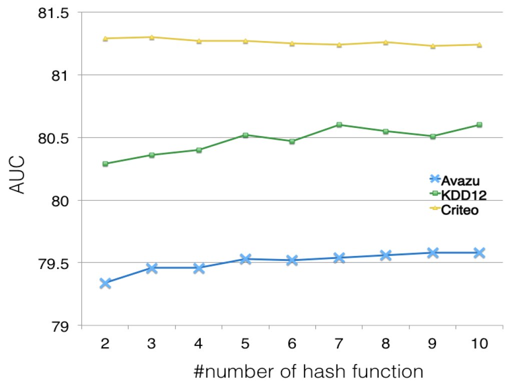

Figure 4. Effect of different hash numbers in HCNet

Table 3. Performance Gains (AUC) of SOTA models after incorporating 2-order HCNet

	Avazu	KDD12	Criteo
Model	AUC(%)	AUC(%)	AUC(%)
DeepFM	78.64	79.40	80.58
DeepFM$_{HCNet}$	79.29	80.25	81.19
Δ	**+0.65**	**+0.85**	**+0.61**
xDeepFM	78.88	79.51	80.64
xDeepFM$_{HCNet}$	79.49	80.34	81.22
Δ	**+0.61**	**+0.83**	**+0.58**
DCN v2	78.98	79.66	80.88
DCN v2$_{HCNet}$	79.42	80.37	81.36
Δ	**+0.44**	**+0.71**	**+0.48**
MaskNet	78.94	79.89	81.07
MaskNet$_{HCNet}$	<u>79.57</u>	<u>80.50</u>	<u>81.40</u>
Δ	**+0.63**	**+0.61**	**+0.33**
DNN	78.67	79.54	80.73
MemoNet	79.44	80.29	81.30
Δ	**+0.77**	**+0.75**	**+0.57**

that the enhanced models significantly outperform the original methods with a large margin.It demonstrates that the proposed solution can effectively make the deep models learn cross features easily through memorizing. Among the enhanced baselines,MaskNet performs best and is the only model outperforming MemoNet with small codebook on all three datasets,which indicates its structure is better at fusing the cross feature's information.

5.6 Effect of Methods to Identify KIFs

In this section, two groups of experiments are conducted to show the effectiveness of our proposed 'feature number ranking'(FNR) and 'field attention ranking'(FAR) methods to identify the 'key interaction field' in order to greatly reduce the feature interaction number in HCNet.First,we rank field according to FNR and FAR scores respectively.Then,top scored fields are gradually added into HCNet to watch the change of model performance.For all three datasets, we select four fields to be added each time and Figure 5 shows the results.We can see from results that:(1)Both the FNR and FAR methods can successfully identify most beneficial fields.Take FAR as example,if we select top 4 ranked fields as KIFs,the performance gain ratio of HCNet will be 93.90%,74.58%,and 65.79% on Avazu,Criteo and KDD12 datasets, respectively.Here performance gain ratio denotes the ratio between performance gain of HCNet with top 4 KIFs compared with DNN and that gain of HCNet using all fields as KIFs.This shows we can leverage a little fraction of fields to obtain majority of performance gain,which will greatly reduce combination number between features.(2)When we compare FNR with FAR, experimental results show FAR outperforms FNR on Avazu and Criteo while FNR performs better on KDD12.

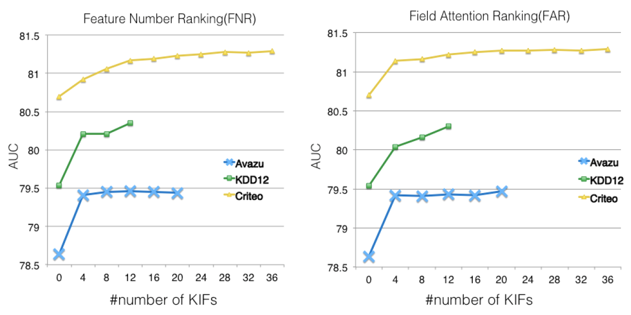

Figure 5. Effect of FNR and FAR

5.7 Effect of Higher Order HCNet

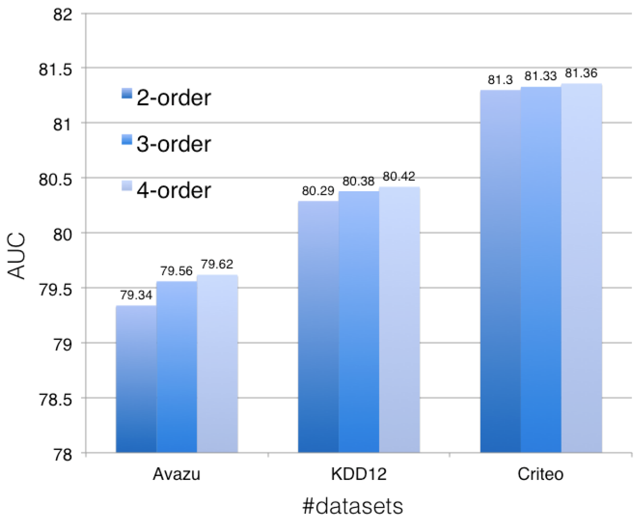

Figure 6. Model performance of high order HCNet

In this section, we evaluate how the performance changes when we plug high order HCNets into MemoNet-L model.Note we allocate new storage resource with size of 1M codewords in codebook of high order HCNet. Their performances on three benchmarks are presented in Figure 6 , from which we can see that:model performance consistently increases when we add higher order HCNet into MemoNet-L on all three datasets.This indicates high order features are beneficial to boost model performance.However,compared with 3-order HCNet,performance gains significantly degraded when 4-order HCNet is added.This implies it's good enough to bring 2-order or 3-order HCNet into MemoNet while more higher order cross feature is unnecessary considering tradeoff between its gain and cost.

5.8 Feature Shrinking Methods Comparison

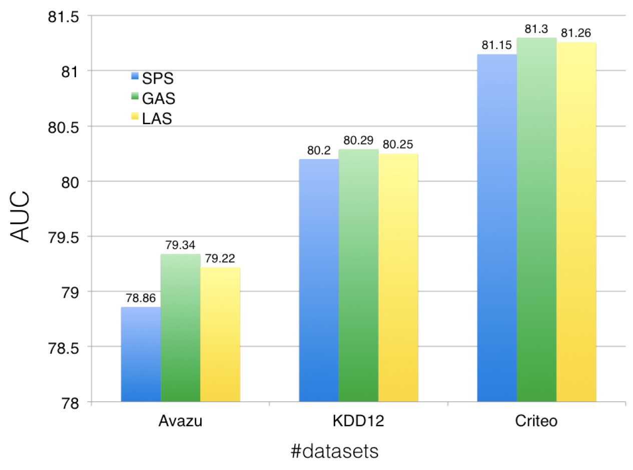

Figure 7. Comparison of shrinking methods

As introduced in Section 4.2,we proposed three feature shrinking methods:sum pooling shrinking(SPS),global attentive shrinking(GAS) and local attentive shrinking(LAS).In order to explore the impact of different feature shrinking approaches,we conduct extensive experiments on three datasets to compare the performance and the experiment results are shown in Figure 7.We can find that GAS achieves the best performance on all datasets. Besides, LAS performs better than SPS.That indicates 1-order features helps identify useful cross features through attention operation.A possible reason why GAS outperforms LAS is that the softmax operation in LAS may wrongly amplify the negative effect of some uninformative cross features.

6 Conclusion and Future Work

We propose HCNet as memory mechanism for efficiently learning and memorizing representations of all cross features in CTR tasks.We also propose MemoNet model by combining HCNet with a DNN backbone.Extensive experimental results on three public datasets show that MemoNet outperforms SOTA models.In future wrok,we will explore more complex codebook structure.

References

[1] 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*. 46–54.

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html

[3] Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2022. Quantifying memorization across neural language models. *arXiv preprint arXiv:2202.07646* (2022).

[4] Olivier Chapelle, Eren Manavoglu, and Romer Rosales. 2014. Simple and scalable response prediction for display advertising. *ACM Transactions on Intelligent Systems and Technology (TIST)* 5, 4 (2014), 1–34.

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

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, 4171–4186. https://doi.org/10.18653/v1/n19-1423

[7] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional Sequence to Sequence Learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017 (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 1243–1252. http://proceedings.mlr.press/v70/gehring17a.html

[8] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

[9] 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, Shinjuku, Tokyo, Japan, August 7-11, 2017*, Noriko Kando, Tetsuya Sakai, Hideo Joho, Hang Li, Arjen P. de Vries, and Ryen W. White (Eds.). ACM, 355–364. https://doi.org/10.1145/3077136.3080777

[10] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, et al. 2014. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*.

ACM, 1–9.

[11] Benjamin Heinzerling and Kentaro Inui. 2020. Language models as knowledge bases: On entity representations, storage capacity, and paraphrased queries. *arXiv preprint arXiv:2008.09036* (2020).

[12] Liangjie Hong, Aziz S Doumith, and Brian D Davison. 2013. Co-factorization machines: modeling user interests and predicting individual decisions in twitter. In *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM, 557–566.

[13] Tongwen Huang, Zhiqi Zhang, and Junlin Zhang. 2019. FiBiNET: combining feature importance and bilinear feature interaction for click-through rate prediction. In *Proceedings of the 13th ACM Conference on Recommender Systems, RecSys 2019, Copenhagen, Denmark, September 16-20, 2019*. ACM, 169–177. https://doi.org/10.1145/3298689.3347043

[14] Yuchin 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*. ACM, 43–50.

[15] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6980

[16] 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*. ACM, 1754–1763.

[17] Bin Liu, Niannan Xue, Huifeng Guo, Ruiming Tang, Stefanos Zafeiriou, Xiuqiang He, and Zhenguo Li. 2020. AutoGroup: Automatic Feature Grouping for Modelling Explicit High-Order Feature Interactions in CTR Prediction. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, Jimmy X. Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (Eds.). ACM, 199–208. https://doi.org/10.1145/3397271.3401082

[18] Bin Liu, Chenxu Zhu, Guilin Li, Weinan Zhang, Jincai Lai, Ruiming Tang, Xiuqiang He, Zhenguo Li, and Yong Yu. 2020. AutoFIS: Automatic Feature Interaction Selection in Factorization Models for Click-Through Rate Prediction. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash (Eds.). ACM, 2636–2645. https://doi.org/10.1145/3394486.3403314

[19] Inbal Magar and Roy Schwartz. 2022. Data Contamination: From Memorization to Exploitation. *arXiv preprint arXiv:2203.08242* (2022).

[20] Richard J Oentaryo, Ee-Peng Lim, Jia-Wei Low, David Lo, and Michael Finegold. 2014. Predicting response in mobile advertising with hierarchical importance-aware factorization machine. In *Proceedings of the 7th ACM international conference on Web search and data mining*. ACM, 123–132.

[21] Junwei Pan, Jian Xu, Alfonso Lobos Ruiz, Wenliang Zhao, Shengjun Pan, Yu Sun, and Quan Lu. 2018. Field-weighted factorization machines for click-through rate prediction in display advertising. In *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 1349–1357.

[22] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).

[23] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* 21, 140 (2020), 1–67.

[24] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International Conference on Data Mining*. IEEE, 995–1000.

[25] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. 2020. Neural collaborative filtering vs. matrix factorization revisited. In *Fourteenth ACM conference on recommender systems*. 240–248.

[26] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2019. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1161–1170.

[27] Kushal Tirumala, Aram H Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. 2022. Memorization Without Overfitting: Analyzing the Training Dynamics of Large Language Models. *arXiv preprint arXiv:2205.10770* (2022).

[28] Peifeng Wang, Filip Ilievski, Muhao Chen, and Xiang Ren. 2021. Do Language Models Perform Generalizable Commonsense Inference? *arXiv preprint arXiv:2106.11533* (2021).

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

[30] Ruoxi Wang, Rakesh Shivanna, Derek Zhiyuan Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed H. Chi. 2021. DCN V2: Improved Deep & Cross Network and Practical Lessons for Web-scale Learning to Rank Systems. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*. 1785–1797.

[31] Zhiqiang Wang, Qingyun She, and Junlin Zhang. 2021. MaskNet: Introducing Feature-Wise Multiplication to CTR Ranking Models by Instance-Guided Mask. In *Proceedings of DLP-KDD 2021*.

[32] Albert Webson and Ellie Pavlick. 2021. Do Prompt-Based Models Really Understand the Meaning of their Prompts? *arXiv preprint arXiv:2109.01247* (2021).

[33] 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. *arXiv preprint arXiv:1708.04617* (2017).

[34] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems* 32 (2019).

[35] Pengtao Zhang and Junlin Zhang. 2022. FiBiNet++:Improving FiBiNet by Greatly Reducing Model Size for CTR Prediction. https://doi.org/10.48550/ARXIV.2209.05016

[36] Weinan Zhang, Tianming Du, and Jun Wang. 2016. Deep learning over multi-field categorical data. In *European conference on information retrieval*. Springer, 45–57.

[37] Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. 2020. Modifying memories in transformer models. *arXiv preprint arXiv:2012.00363* (2020).

[38] Jieming Zhu, Jinyang Liu, Shuai Yang, Qi Zhang, and Xiuqiang He. 2020. FuxiCTR: An Open Benchmark for Click-Through Rate Prediction. *ArXiv* abs/2009.05794 (2020).