electronics

MDPI

Article

Interaction-Aware Structured Pruning for Click-Through Rate Prediction Models with Data-Adaptive Noise Gating

Beomhee Han and Chang Ouk Kim *

Department of Industrial Engineering, Yonsei University, Seoul 03722, Republic of Korea; firefox@yonsei.ac.kr
* Correspondence: kimco@yonsei.ac.kr

Abstract

Click-through rate (CTR) prediction models in real-time bidding (RTB) systems must remain accurate under strict latency budgets (about 20 ms for model inference). Explicit interaction architectures such as Deep & Cross Network V2 (DCN V2) combine large embedding tables with a cross-layer cost that grows quadratically with the input dimension. We propose *Interaction-Aware Structured Pruning* (IASP), which scores embedding dimensions by combining a Taylor-inspired importance score (magnitude and gradient) with each dimension's participation in the cross network; a data-adaptive gate, whose threshold τ is a percentile of the interaction scores, restricts the synergy boost to dimensions with strong cross-layer participation. On the Avazu and Criteo datasets, IASP retains or improves the area under the ROC curve (AUC) at up to 99% embedding sparsity for DCN and DCN V2 and outperforms magnitude- and gradient-based criteria at 90–95% sparsity within a shared pruning pipeline. Physically removing the lowest-ranked dimensions (one dimension per field retained) reduces the memory footprint by about 16× and yields up to a 6.6× single-thread CPU speedup for DCN V2 when 200 candidates are ranked per request, whereas the speedup for single-candidate requests is marginal.

Keywords: click-through rate prediction; deep and cross networks; model compression; noise gating; RTB systems; structured pruning

1. Introduction

In the era of big data, real-time recommendation systems and Click-Through Rate (CTR) prediction have become the primary revenue drivers for global digital advertising. These systems operate in high-throughput environments, such as real-time bidding (RTB), where billions of transactions occur daily. The ultimate goal is to rank candidate items in real-time within a latency limit of 100 ms for the entire OpenRTB cycle. Deducting network transit and feature extraction times, the stringent Service Level Agreement (SLA) leaves less than 20 ms for the deep learning model's pure inference phase [1,2].

Modern CTR models have evolved from shallow Factorization Machines (FMs) [3] to sophisticated deep learning architectures such as Deep & Cross Network (DCN) [4] and DCN V2 [5]. Attention mechanisms and sequential modeling have further advanced CTR architectures (e.g., DIN [6], DIEN [7], AutoInt [8]). Among these architectures, explicit interaction models such as DCN V2 excel at modeling high-order feature interactions. Although these models achieve remarkable prediction accuracy, their complexity—specifically the $O(d^2)$ matrix-based interactions in DCN V2 [5] (where d is the dimension of the concatenated input embedding)—incurs a substantial computational cost. Furthermore, owing to high-cardinality ID features in CTR domains, over 99% of the model parameters

Check for updates

Academic Editor: Xinshun Xu

Received: 13 September 2026
Revised: 21 October 2026
Accepted: 26 October 2026
Published: 30 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

are heavily concentrated in embedding tables. This creates the well-known "memory wall" problem in recommendation systems, where memory bandwidth severely limits inference throughput on standard hardware [9,10]. Consequently, high redundancy emerges, as the majority of features reside in the long-tail region, contributing to memory inefficiency and potential overfitting.

To bridge the gap between inference efficiency and feature interaction preservation, we propose Interaction-Aware Structured Pruning (IASP). IASP evaluates the importance of each feature dimension by capturing its baseline capacity (via a Taylor-inspired combination of weight magnitude and gradient sensitivity) and its structural synergy within the Cross Network [5]. We further introduce a percentile-based gating mechanism (τ) that restricts the synergy amplification to dimensions with strong cross-layer participation, so that weakly participating dimensions are not amplified.

The main contributions of this paper are summarized as follows:

- **Novel Interaction-Aware Pruning Framework:** We introduce a dimension-wise structured pruning method that explicitly accounts for the feature interaction mechanisms of CTR models, mitigating the loss of critical synergistic information during extreme compression.
- **Data-Adaptive Noise Gating Mechanism:** We formulate a threshold-based gating function whose threshold is recomputed as a percentile of the interaction scores at each pruning step, so that only dimensions with strong cross-layer participation receive a synergy boost.
- **Significant SLA-Compliant Acceleration:** By physically reducing the architecture, IASP achieves a theoretical quadratic speedup (reducing the complexity from $O(d^2)$ to $O(k^2)$, where $k \ll d$ is the input dimension after pruning) on complex matrix-based interaction models like DCN V2 [5], yielding up to a 6.6× single-thread CPU speedup when ranking 200 candidates per request and widening the margin under the 20 ms RTB latency constraint on standard CPU environments.

The scope of this study is the importance criterion: IASP is compared with magnitude-, gradient-, and Taylor-based criteria within a shared pruning pipeline, and we do not claim superiority over search-based or threshold-learning compression methods, which are discussed in Sections 2.2 and 4.1.

2. Related Work

The methodology of our framework lies at the intersection of explicit feature interaction modeling and deep neural network compression.

2.1. Evolution of CTR Prediction Models

Early CTR prediction models primarily relied on linear combinations or shallow structures like FM [3], which benefit from lightweight computation but exhibit limited representational capacity for complex non-linear patterns. To capture high-order non-linear interactions, deep-learning-based hybrid structures such as Wide & Deep [11], DeepFM [12], and xDeepFM [13] were proposed. However, their deep components model these interactions implicitly and act as black boxes, making interaction modeling difficult to interpret.

To overcome these structural limitations, DCN explicitly modeled high-order feature interactions using cross layers parameterized by weight vectors [4], while Attentional Factorization Machines (AFMs) learned interaction weights via attention networks [14]. DCN V2, the successor of DCN, extended this architecture by substituting the vector parameters with matrix multiplications to substantially enhance interaction expressiveness [5].

To formalize this matrix-based interaction mechanism, the l-th cross layer in DCN V2 is mathematically expressed as

$$\mathbf{x}_{l+1} = \mathbf{x}_0 \odot (\mathbf{W}_l \mathbf{x}_l + \mathbf{b}_l) + \mathbf{x}_l, \quad (1)$$

where $\mathbf{x}_l, \mathbf{x}_{l+1} \in \mathbb{R}^d$ represent the hidden features at layer l and $l+1$, respectively; $\mathbf{x}_0 \in \mathbb{R}^d$ is the base embedded input; $\odot$ denotes the element-wise (Hadamard) product; $\mathbf{W}_l \in \mathbb{R}^{d \times d}$ is the dense layer weight matrix; and $\mathbf{b}_l \in \mathbb{R}^d$ is the bias vector. While this explicit matrix-based modeling achieves strong predictive accuracy, the $\mathbf{W}_l \mathbf{x}_l$ operation fundamentally dictates a quadratic computational complexity of $O(d^2)$ per layer. In real-world RTB environments with large embedding dimensions, this limitation can incur substantial latency overheads in online systems.

2.2. Limitations of Conventional Compression Methods

To alleviate the inference burden of deep networks, various model compression paradigms have been explored. Quantization [15] lowers numeric precision but retains the $O(d^2)$ structure of the cross layers, and knowledge distillation [16] increases training overhead and complicates production pipelines. Table 1 compares representative embedding compression and pruning methods in terms of their pruning unit, additional procedures, and deployable output structure.

Table 1. Taxonomy of CTR Model Compression Methods based on Structural Output and Hardware Efficiency.

Method	Pruning Unit	Extra Procedure	Output Structure	CPU Speedup (w/o Sparse Kernels)	Uses Interaction Stats
AutoDim [17]	Field-wise dimension	Architecture search	Field-specific widths	Partial	No
PEP [18]	Individual parameter	Learnable thresholds + retraining	Feature-specific sparse	No	No
OptEmbed [19]	Feature row + field dim.	Supernet + evolutionary search	Pruned rows, field widths	Partial	No
Mag./Grad./Taylor [20]	Parameter	None	Masked	No (unless mapped)	No
IASP (Ours)	**Dimension (scored per param.)**	**Iterative pruning and rewinding + tuning of $\alpha, \beta, \gamma, \delta, p$**	**Uniform-width dense**	**Yes**	**Yes**

The proposed method is shown in bold.

While unstructured weight pruning [21] reduces the number of nonzero parameters [22], it introduces irregular sparse memory access and rarely yields actual inference speedups on general-purpose CPUs without specialized sparse kernels. On the other hand, common structured pruning methods [23] (e.g., channel slimming [24]) are primarily developed for convolutional networks.

To specifically target CTR models, recent studies have proposed field-aware embedding dimension search (AutoDim) [17], Plug-in Embedding Pruning (PEP) [18], and compositional hashing [25]. While these methodologies pioneered embedding dimension reduction, they introduce additional optimization procedures: AutoDim relies on an architecture search whose cost grows with the search space, while PEP learns pruning thresholds jointly with the model and produces fine-grained, feature-specific sparsity that reduces storage but does not change tensor shapes. These additional procedures make them less convenient in RTB production environments, where models are frequently retrained.

Subsequent studies introduced dynamic embedding allocation (e.g., OptEmbed [19]) and automated feature architecture selection (e.g., AutoFAS [26]) to balance memory limits

and search efficiency. More recently, the research focus has shifted toward unified compression paradigms to bypass these search overheads. For instance, unified low-rank decomposition frameworks [27], model-agnostic quantization mechanisms leveraging popularity-weighted regularization [28], lightweight distillation-based graph neural networks for recommendation [29], and learnable sparse projection-based embedding compression [30] have been introduced. A recent survey [31] provides a comprehensive review of embedding compression techniques. While they effectively reduce memory footprints, most of them compress embedding tables independently of the downstream interaction network and therefore do not reflect how embedding dimensions participate in explicit interaction layers such as the DCN cross layers.

Recent studies have also examined the overfitting behavior of deep CTR models [32] and proposed feature and embedding-dimension selection methods such as AutoFIS [33], AutoField [34], AdaFS [35], and SSEDS [36]. These methods select features, interactions, or dimensions within the embedding or interaction layer, whereas IASP ranks embedding dimensions by combining their own importance with their participation in the explicit cross network, targeting dense explicit-interaction architectures.

Beyond recommender systems, related ideas have been explored in general-purpose learning. The kNN-Graph model [37] learns sample-specific neighborhood sizes and shifts the cost of neighbor selection from inference to a precomputation phase to reduce inference latency, and robust quantum feature selection [38] selects informative features under sparse regularization using quantum sparse optimization circuits. Like robust feature selection, IASP retains a compact set of informative input dimensions, and, like kNN-Graph, it moves computation from inference to an offline phase. Unlike these methods, it operates on the learned embedding dimensions of a deep CTR model and scores each dimension using the weights and gradients of the downstream cross network.

3. Proposed Method: Interaction-Aware Structured Pruning

In this section, we present the technical details of the proposed IASP framework. First, we introduce the overall iterative execution pipeline (Section 3.1). Within this pipeline, the core pruning mechanism operates via three sequential phases: computing a Taylor-inspired baseline importance for each parameter (Section 3.2), integrating structural synergy with data-adaptive noise gating (Section 3.3), and generating pruning masks via scalable sub-sampled quantiles and converting them into a dimension-level structure (Section 3.4). Finally, we detail the automated iterative pruning and rewinding protocol designed to maximize the representation capacity of the compressed network (Section 3.5).

3.1. Overall Framework Pipeline

As illustrated in Figure 1, to identify and prune redundant dimensions within embedding tables while accounting for the explicit cross-layer structure, IASP monitors both static parameter states and dynamic gradient trajectories. The execution flow directly targets the embedding layer and the subsequent interaction functions. Specifically, the IASP module operates as a control mechanism: it extracts both the weights and gradients from the embedding layer and the subsequent interaction backbone (e.g., Cross Network) to evaluate dimension-wise importance. After computing the structural synergy and applying the noise gating threshold, it generates pruning masks over the embedding parameters, which are applied during iterative retraining. After the final cycle, the redundant embedding dimensions and the corresponding cross-layer rows and columns are physically removed, which in turn reduces the input width of the coupled Deep Network.

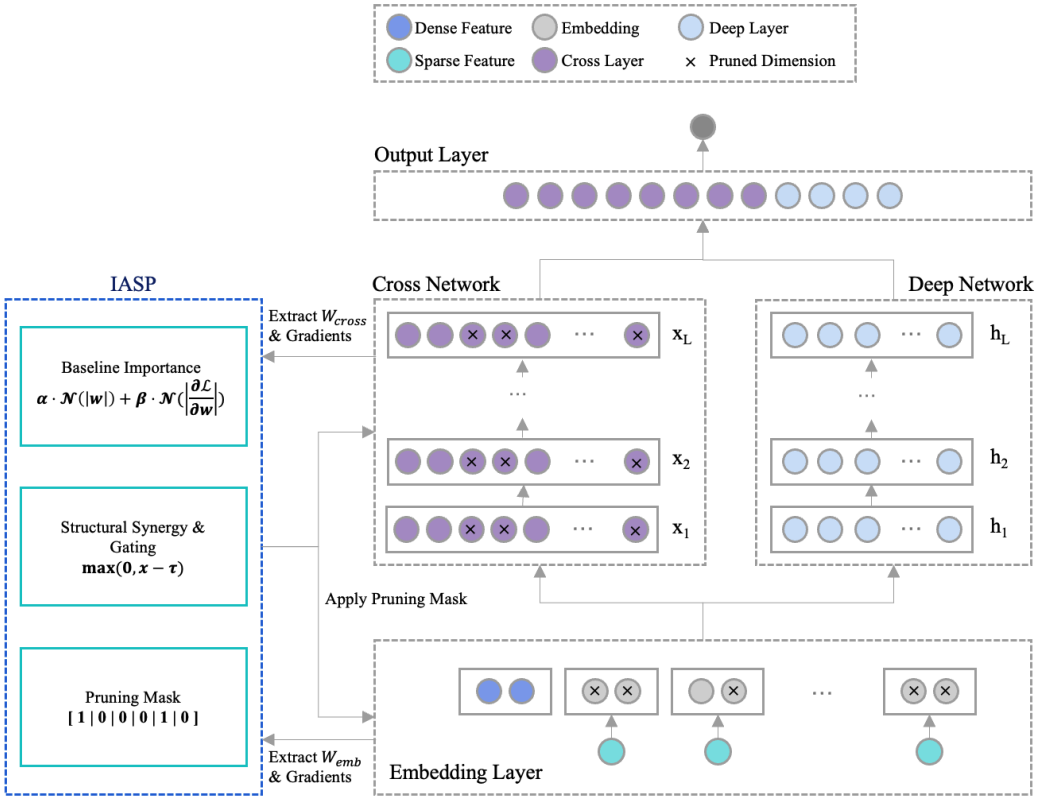


Figure 1. An illustration of the proposed IASP framework applied to a representative DCN architecture. Acting as an external module, the IASP module extracts parameter weights and gradients from both the Embedding Layer and the Cross Network. By integrating baseline importance with structural synergy and applying a data-adaptive noise gating mechanism, IASP generates pruning masks. After the final pruning cycle, the lowest-ranked dimensions (denoted by × marks) are physically removed, which in turn reduces the input width of the coupled Deep Network, compressing the architecture while retaining dimensions with strong interaction participation.

This pipeline is iteratively executed following the Lottery Ticket Hypothesis [22], where the model weights are periodically rewound to their initial configurations θ_0 after mask generation, allowing the surviving sub-network to adaptively fine-tune its continuous representation mappings.

3.2. Taylor-Inspired Baseline Importance

Traditional model compression methodologies predominantly rely on static weight magnitude $|w|$, assuming that smaller parameters contribute minimally to the output. However, in CTR domains with heavy power-law distributions, low-frequency embedding indices may possess small absolute magnitudes yet yield substantial gradient adjustments during backpropagation.

To mathematically formulate this, we define the optimization objective as minimizing the absolute change in the loss function $\mathcal{L}$ incurred by removing a parameter w (equivalent to setting $\Delta w = w$). Expanding the non-linear loss function $\mathcal{L}(w - \Delta w)$ around the local neighborhood of w using a second-order Taylor series yields

$$\mathcal{L}(w - \Delta w) \approx \mathcal{L}(w) - \frac{\partial \mathcal{L}}{\partial w}\Delta w + \frac{1}{2}\Delta w^T \mathbf{H} \Delta w, \tag{2}$$

where $\mathbf{H}$ denotes the Hessian matrix containing second-order partial derivatives. Rearranging the formulation for the structural delta $|\Delta\mathcal{L}|$ results in

$$|\Delta\mathcal{L}| \approx \left|\frac{\partial\mathcal{L}}{\partial w}w - \frac{1}{2}w^T\mathbf{H}w\right|. \tag{3}$$

Computing the full Hessian matrix $\mathbf{H}$ for recommendation architectures with hundreds of millions of parameters introduces an intractable computational complexity of $O(N^2)$, where N is the number of parameters, which is computationally prohibitive on commercial hardware clusters. Consequently, we utilize a first-order approximation (i.e., the Taylor product $\frac{\partial\mathcal{L}}{\partial w}w$). This approximation neglects second-order curvature, which may not be negligible in the highly non-convex landscape of extremely sparse CTR models; incorporating curvature information (e.g., diagonal Fisher or Hessian approximations) is a natural extension of the proposed criterion. First-order Taylor criteria [20] use this product directly, whereas Optimal Brain Damage [39] relies on a diagonal approximation of the second-order term. However, applying the multiplicative form to evaluate dimension importance in highly sparse CTR domains raises two concerns:

- **The Cancellation Effect:** If the unadjusted Taylor products were algebraically summed across an entire embedding dimension, positive and negative gradient-weight products within the same dimension would offset each other. Consequently, a highly active dimension catalyzing numerous dynamic interactions could erroneously compute to a near-zero overall score, leading to the unintended pruning of a vital feature. This concern applies when signed products are aggregated over a dimension; the parameter-level absolute product evaluated in Section 4.5 is not affected by it.
- **Sensitivity to Heavy-Tailed Gradients:** Due to the long-tail distribution of CTR data, gradient estimates from a finite calibration sample often contain large spikes. Multiplying the weight by such an estimate can make the scores, and hence the ranking, sensitive to a few large gradient values.

To address these concerns, we adopt an additive formulation. Theoretically, a monotonic logarithmic transformation could decouple this product (i.e., $\log(|a \cdot b|) = \log(|a|) + \log(|b|)$) while preserving the ranking order of parameter importance. However, applying a pure logarithm in highly sparse networks introduces undefined values ($\log(0) \to -\infty$) for dormant parameters and yields unbounded scores that cannot be placed on a common scale with the subsequent synergy term.

To resolve this instability, we propose a heuristic surrogate. Instead of a direct logarithmic replacement, we map both the structural state and the dynamic sensitivity into a unified $[0, 1]$ space using a bounded monotonic mapping: global min–max normalization $\mathcal{N}(\cdot)$. We formally define the intrinsic baseline importance score $\mathcal{B}(w)$ as

$$\mathcal{B}(w) = \alpha \cdot \mathcal{N}(|w|) + \beta \cdot \mathcal{N}\left(\left|\frac{\partial\mathcal{L}}{\partial w}\right|\right), \tag{4}$$

where α and β are control coefficients that balance the static structural capacity and the dynamic optimization sensitivity. Here, $\mathcal{N}(\cdot)$ is mapped as

$$\mathcal{N}(x) = \frac{x - \min(x)}{\max(x) - \min(x) + \epsilon}. \tag{5}$$

By substituting the unbounded logarithm with a generalized normalized addition ($\epsilon = 10^{-12}$ is injected to prevent division-by-zero), this heuristic bounds the scores to $[0, \alpha + \beta]$ and provides a common scale for the subsequent synergy term. Note that this surrogate does not, in general, preserve the ranking induced by the Taylor product; it is a heuristic whose effect we evaluate empirically against the conventional Taylor criterion in Section 4.5.

3.3. Interaction-Aware Gating and Structural Synergy

While the baseline approximation preserves independent parameter properties, it treats features as isolated entities, which does not fully capture the mechanics of modern explicit interaction architectures. To compensate for this, IASP hooks into the interaction modules (e.g., a dense projection layer or the explicit Cross Network in DCN V2) to capture structural co-evolution statistics. We define the unified structural synergy score $\mathcal{S}(i)$ for the i-th embedding dimension as

$$\begin{aligned}\mathcal{S}(i) = \mathcal{B}\big(\mathbf{W}_{emb}^{(i)}\big) \\ \times \Big(1.0 + \mathcal{G}\big(\gamma \cdot \mathcal{N}(I_W^{(i)})\big) + \mathcal{G}\big(\delta \cdot \mathcal{N}(I_G^{(i)})\big)\Big),\end{aligned} \tag{6}$$

where $I_W^{(i)}$ and $I_G^{(i)}$ are the L_1-norms of the cross-layer weights and of their gradients connected to the i-th embedding dimension, and γ and δ are control hyperparameters governing the interaction influence. The baseline term $\mathcal{B}(\mathbf{W}_{emb}^{(i)})$ is obtained by averaging $\mathcal{B}(w)$ over the parameters belonging to the i-th dimension. Formally,

$$I_W^{(i)} = \sum_{l=1}^{L}\sum_{j=1}^{d}\left|\mathbf{W}_{l,ij}\right|, \quad I_G^{(i)} = \sum_{l=1}^{L}\sum_{j=1}^{d}\left|\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{l,ij}}\right|, \tag{7}$$

where $\mathbf{W}_l$ is the weight matrix of the l-th cross layer, L is the number of cross layers ($L = 3$ in our experiments), and j runs over the d entries of its i-th row. For DCN, whose cross layers use weight vectors $\mathbf{w}_l \in \mathbb{R}^d$, the inner sum reduces to the single entry $|\mathbf{w}_{l,i}|$.

Crucially, $\mathcal{G}(x)$ acts as a data-adaptive gating function (a one-sided shrinkage operator), formulated as

$$\mathcal{G}(x) = \max(0, x - \tau). \tag{8}$$

We refer to $\mathcal{G}$ as a noise gate by analogy with signal processing, in which a noise gate passes only components above a threshold; the term describes this thresholding operation and does not imply that the gated dimensions are noise.

To ensure the framework is genuinely "data-adaptive" (i.e., responsive to varying data distributions and training steps), the threshold τ is not a static constant. Instead, τ is calculated at each mask generation step as the p-th percentile of the combined interaction scores $\gamma \cdot \mathcal{N}(I_W^{(i)}) + \delta \cdot \mathcal{N}(I_G^{(i)})$ across embedding dimensions. This effectively means that for $p = 70$, dimensions in the bottom 70% of the combined interaction scores receive no amplification, and only dimensions in the top 30% can receive a synergistic multiplier boost, thereby reallocating the parameter budget toward dimensions with strong interaction participation.

Because $\mathcal{G}(x) \geq 0$, the multiplier $(1.0 + \mathcal{G}(\cdot))$ is always at least 1.0. If a dimension's interaction affinity drops below the dynamic threshold τ, $\mathcal{G}(x)$ returns 0. This does not forcibly zero out the parameter immediately; rather, it deprives the dimension of "synergy amplification," leaving its score strictly at its baseline $\mathcal{B}(\cdot) \times 1.0$. Conversely, dimensions with strong interaction participation receive larger multipliers, which are bounded by $1 + \gamma + \delta$.

3.4. Scalable Mask Generation via Sub-Sampled Quantiles

Importance is evaluated per embedding parameter: each parameter w in the i-th embedding dimension receives the score $\mathcal{B}(w) \cdot \big(1 + \mathcal{G}(\gamma \cdot \mathcal{N}(I_W^{(i)})) + \mathcal{G}(\delta \cdot \mathcal{N}(I_G^{(i)}))\big)$, i.e., its baseline importance scaled by the gated synergy of the dimension it belongs to. Let $\mathbf{V}_{score}$ denote the vector of these parameter-level scores. Because the embedding tables of industrial datasets such as Criteo contain hundreds of millions of parameters, computing an exact global quantile over $\mathbf{V}_{score}$ is memory-intensive. When $|\mathbf{V}_{score}|$ exceeds $\Omega_{max} = 5{,}000{,}000$, we draw a uniformly random subset:

$$\tilde{\mathbf{V}}_{score} = \text{Sample}(\mathbf{V}_{score}, \Omega_{max}), \tag{9}$$

and approximate the pruning threshold for the current target sparsity κ_t (defined in Section 3.5) as

$$\mathcal{T}_{thresh} = \text{Quantile}(\tilde{\mathbf{V}}_{score}, \kappa_t). \tag{10}$$

Parameters whose scores fall below $\mathcal{T}_{thresh}$ are masked, and this mask is used during iterative retraining (Section 3.5). After the final cycle, the mask is converted into a deployable structure by retaining, in each field, the K dimensions with the highest $\mathcal{S}(i)$, where $K = 4, 2$, and 1 for 90%, 95%, and 99% target sparsity (Section 5.1). The resulting input dimension is $k = F \cdot K + N_{num}$, where F is the number of categorical fields and N_{num} is the number of numerical features, which are kept unpruned ($N_{num} = 13$ for Criteo and 0 for Avazu); the corresponding columns of the embedding tables and the rows and columns of each cross-layer matrix $\mathbf{W}_l$ (the corresponding entries of $\mathbf{w}_l$ for DCN) are then physically removed, reducing the input dimension from d to k and the cross-layer cost from $O(d^2)$ to $O(k^2)$.

3.5. Iterative Pruning and Rewinding Protocol

To maximize the learning capacity of compressed topologies, IASP adheres to an automated schedule governed by an exponential decay trajectory. Pruning starts from a model trained for one warm-up epoch. Given a target sparsity κ_{target} and total pruning cycles T, the sparsity metric for step $t \in \{1, \ldots, T\}$ is configured as

$$\kappa_t = 1 - (1 - \kappa_{target})^{\frac{t}{T}}. \tag{11}$$

At each step, statistics are extracted over 150 calibration mini-batches via forward and full backward execution hooks in PyTorch (version 2.2.2). After the masks are updated, the weights are rewound to their initial values θ_0, and the masked network is fine-tuned for one epoch.

The mask generation step adds little to the training cost. At each cycle, it consists of element-wise operations over the embedding parameters, a percentile over the d combined interaction scores ($d = 368$ for Avazu and 429 for Criteo) to obtain τ, and a quantile over at most Ω_{max} sampled parameter scores, and it is executed only $T = 10$ times in total. The dominant cost is the calibration pass over 150 mini-batches, which corresponds to approximately 3.8% and 3.4% of the iterations in one training epoch on Avazu and Criteo, respectively (about 3950 and 4480 iterations per epoch at a batch size of 8192).

4. Experimental Results

4.1. Experimental Setup

To evaluate the proposed IASP framework, we conducted experiments on two industry-standard CTR prediction benchmarks: the Avazu and Criteo datasets. The Avazu dataset comprises over 40 million samples with 23 categorical fields, representing a highly sparse mobile advertising environment where long-tail distributions dominate. The Criteo dataset contains over 45 million samples across 39 fields (13 numerical, 26 categorical), providing a dense-sparse mixed signal environment. We applied IASP to both DCN and DCN V2 architectures, comparing its efficacy against conventional magnitude-based and gradient-based pruning methods under extreme sparsity conditions (90–99%). The magnitude- and gradient-based criteria score each embedding parameter by $|w|$ and by $|\partial \mathcal{L}/\partial w|$ averaged over the calibration batches, respectively. Predictive performance is measured by the area under the receiver operating characteristic curve (AUC) and the LogLoss (binary cross-entropy) on the test split. The conventional first-order Taylor criterion [20] is additionally compared within the same pipeline in the ablation study (Section 4.5). Unless otherwise noted, results are from a single run with seed 42. To assess run-to-run variability, we repeated the experiment for DCN V2 on Avazu, the architecture–dataset pair with the

largest differences among criteria, at 95% sparsity with two additional seeds (43 and 44), retraining the dense baseline and all pruned models for each seed (Section 4.2).

Dense baselines are reported at their best validation checkpoint, so the observed gains are not attributable to under-trained baselines. In contrast, the pruned models strictly followed the automated iterative pruning and rewinding protocol (Section 3.5) with a fixed computational budget (one warm-up epoch followed by 10 cycles × 1 fine-tuning epoch). All evaluations were conducted using identical data splits (8:1:1 for training, validation, and testing). CTR-specific embedding compression methods such as AutoDim and PEP were not included as experimental baselines. Our study compares importance criteria within a single pruning pipeline—identical mask granularity, sparsity schedule, weight rewinding, and training budget—so that differences in accuracy reflect the scoring criterion (including its tuned coefficients) rather than differences in the pipeline. These methods instead rely on their own architecture-search or threshold-learning procedures (Table 1); including them would conflate the scoring criterion with a different optimization procedure. The low-rank compression framework [27] was likewise not included. Although it has been applied to DCN V2, it compresses each embedding table with a PCA-based projection, restores the original embedding width through an additional linear layer before the feature interaction module, and decomposes the MLP layers rather than the cross layers. The input dimension and the $O(d^2)$ cost of the cross layers therefore remain unchanged, and the framework could be combined with IASP rather than replacing it. Similarly, the quantization-based framework [28] replaces each embedding with product-quantization codes and shared codebooks; this reduces storage but does not reduce the input dimension of the cross layers, and it could be applied to the pruned embedding tables of IASP as a complementary step, while the methods in [29,30] target graph neural network recommenders and embedding-based retrieval, respectively, rather than ranking models with explicit cross layers.

For the IASP framework, the hyperparameters (α, β, γ, δ, and p) were systematically optimized via the Tree-structured Parzen Estimator (TPE) algorithm using the Optuna framework. The full list of optimal hyperparameters (Table A1) and detailed reproducibility settings (e.g., search ranges, batch size) are provided in Appendix A.

The gating percentile p was selected by Optuna from $\{p_{50}, \ldots, p_{90}\}$ (Appendix A), and τ is recomputed as the p-th percentile of the combined interaction scores across embedding dimensions at each pruning step.

4.2. Performance and Stability Analysis

As shown in Figure 2, IASP achieves higher AUC than the magnitude- and gradient-based criteria at 90% and 95% sparsity in all four settings, with the largest margins for DCN V2 on Avazu, and remains comparable at 99% sparsity. In this run (seed 42), the conventional criteria on DCN V2 (Avazu) remain near or below the dense baseline at 90–95% sparsity, whereas IASP stays above it at all sparsity levels. Because all criteria share the same pruning pipeline, this difference reflects the importance criterion (with its tuned coefficients) rather than the pipeline. Removing a dimension in DCN V2 also removes a row and a column of each cross-layer matrix $\mathbf{W}_l$, which motivates the interaction-aware terms; Section 4.5 examines how much each component of the IASP score contributes. On Criteo, all criteria exceed the dense baseline, and the advantage of IASP over the better conventional criterion is smaller (0.0011–0.0032 in AUC at 90–95% sparsity); because these are single runs, such small differences should be interpreted with caution.

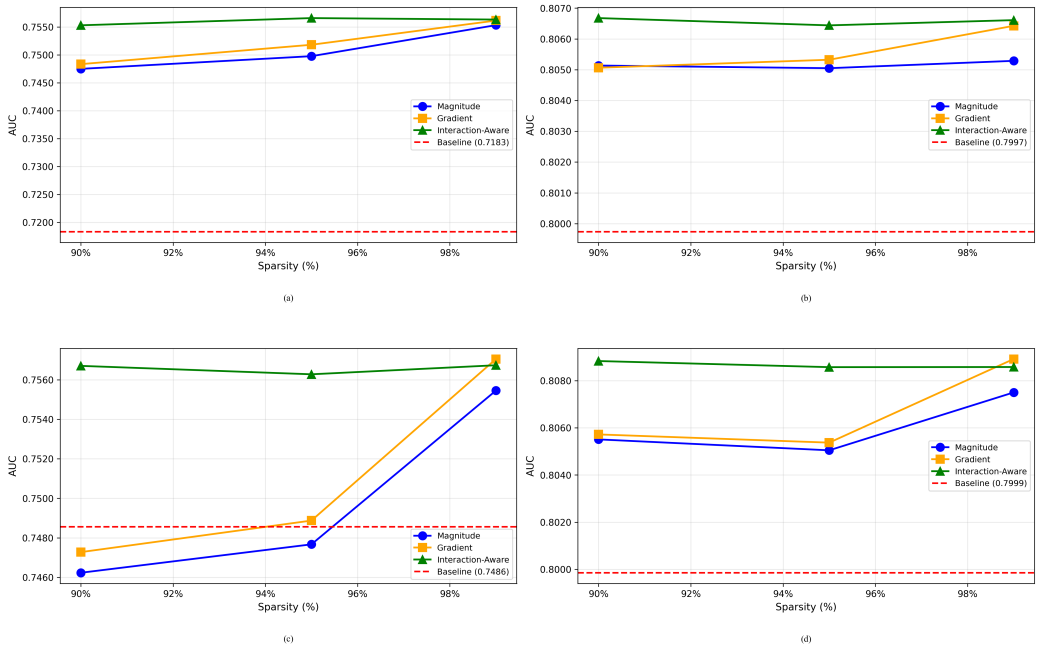


Figure 2. Model prediction performance (AUC) vs. sparsity level. (**a**) DCN on Avazu. (**b**) DCN on Criteo. (**c**) DCN V2 on Avazu. (**d**) DCN V2 on Criteo. IASP maintains its AUC from 90% sparsity onward. Results are from a single run (seed 42); results over three seeds for DCN V2 on Avazu at 95% sparsity are given in Table 2. Exact values are given in Table A2.

Table 2. Test AUC and LogLoss over three random seeds (DCN V2, Avazu, 95% sparsity). For each seed, the dense baseline and all pruned models were trained with that seed. SD denotes the standard deviation over seeds. Means, SDs, and the paired statistics reported in the text were computed from unrounded values.

Model	Seed 42	Seed 43	Seed 44	Mean ± SD
AUC (↑)				
Dense (Baseline)	0.7486	0.7460	0.7498	0.7481 ± 0.0019
Magnitude	0.7477	0.7522	0.7516	0.7505 ± 0.0025
Gradient	0.7489	0.7545	0.7509	0.7514 ± 0.0028
IASP	**0.7563**	**0.7574**	**0.7568**	**0.7568 ± 0.0005**
LogLoss (↓)				
Dense (Baseline)	0.3826	0.3840	0.3811	0.3826 ± 0.0015
Magnitude	0.3870	0.3833	0.3845	0.3850 ± 0.0019
Gradient	0.3858	0.3804	0.3844	0.3836 ± 0.0028
IASP	**0.3784**	**0.3783**	**0.3779**	**0.3782 ± 0.0003**

↑ indicates that higher is better, and ↓ indicates that lower is better. Bold indicates the best value in each column.

To assess run-to-run variability, Table 2 reports DCN V2 on Avazu at 95% sparsity over three seeds; for each seed, the dense baseline and all pruned models were retrained, and each pruned model is compared with the dense model of the same seed. The dense baseline itself varied across seeds (AUC 0.7460–0.7498), and the margins of the magnitude- and gradient-based criteria over it were inconsistent: the magnitude-based model fell below it in seed 42, and the gradient-based model exceeded it by only 0.0003 in that seed (AUC differences from −0.0009 to +0.0085). In contrast, IASP exceeded the dense baseline in every seed (+0.0070 to +0.0114) and obtained the highest AUC and the lowest LogLoss in all three seeds, with the smallest seed-to-seed standard deviation (0.0005 in AUC and 0.0003 in LogLoss, compared with 0.0019–0.0028 and 0.0015–0.0028 for the other models). The paired AUC improvements of IASP were 0.0087 ± 0.0024, 0.0063 ± 0.0020, and 0.0054 ± 0.0023 over

the dense, magnitude-based, and gradient-based models, respectively (mean ± SD over seeds; paired t-tests, $p = 0.024$, 0.032, and 0.055). Three seeds, however, provide limited statistical power.

4.3. Regularization Effect

As shown in Figure 3, IASP achieved a lower test LogLoss than the dense baseline at all sparsity levels in all four settings. The magnitude- and gradient-based criteria also reduced the loss in most settings, but on DCN V2 (Avazu) their LogLoss exceeded the dense baseline at 90–95% sparsity in the run shown in Figure 3 and fell below it only at 99%, whereas IASP reached a low loss already at 90% sparsity. Across the three seeds at 95% sparsity (Table 2), the LogLoss of IASP was below that of the dense baseline in every seed (by 0.0032–0.0057), whereas the magnitude- and gradient-based criteria exceeded it in two of the three seeds. These results are consistent with a regularization effect of removing long-tail embedding capacity.

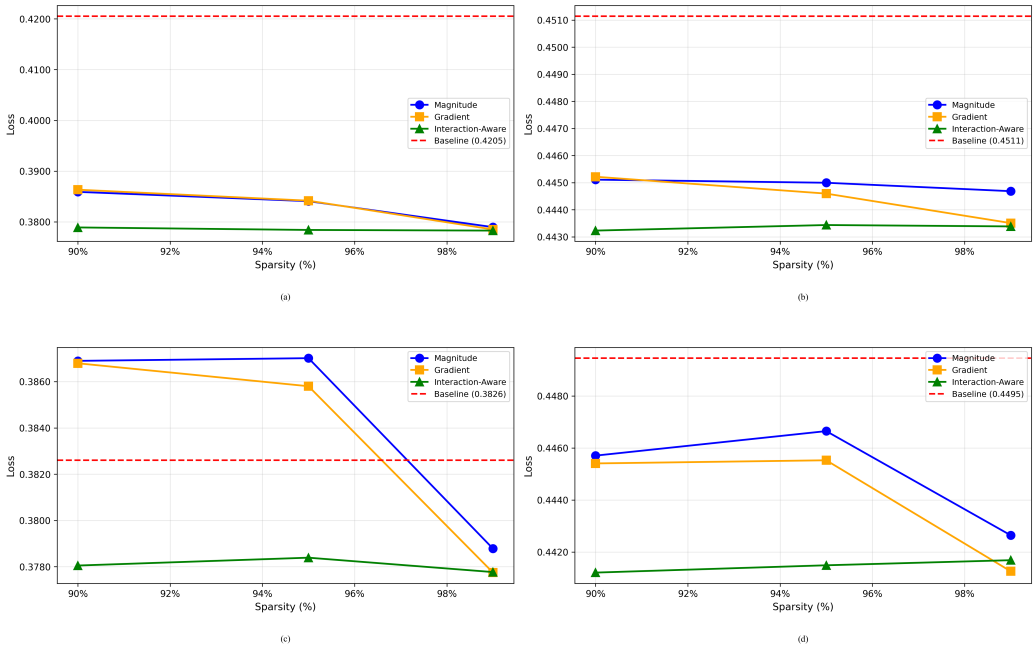


Figure 3. Test LogLoss vs. sparsity level. (**a**) DCN on Avazu. (**b**) DCN on Criteo. (**c**) DCN V2 on Avazu. (**d**) DCN V2 on Criteo. IASP attains a lower LogLoss than the dense baseline at all sparsity levels, consistent with a regularization effect. Results are from a single run (seed 42); see Table 2 for results over three seeds. Exact values are given in Table A2.

4.4. Qualitative Analysis: Feature Survival at Extreme Sparsity

To examine how the criteria allocate the retained parameters, we analyzed the composition of surviving parameters under the 99% pruning mask (Figure 4). On Avazu, magnitude-based pruning allocated a large share of the retained parameters to *device_id*, a field with 2,686,408 unique values whose most frequent value accounts for 82.5% of samples. IASP assigned a smaller share to *device_id* and larger shares to *device_ip* and *device_model*. On Criteo, magnitude-based pruning concentrated the budget on a single field (*C12*), whereas IASP distributed it more evenly across several top fields (e.g., *C03*, *C16*, *C24*). These compositions describe how each criterion prioritizes fields; they do not by themselves establish that the reallocation causes the observed AUC differences.

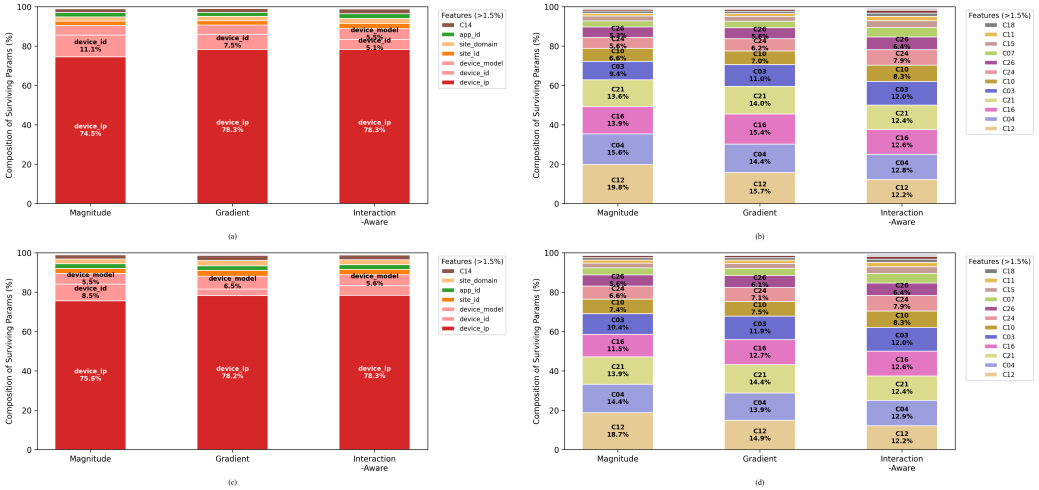


Figure 4. Composition of surviving parameters under the 99% pruning mask. (**a**) DCN on Avazu. (**b**) DCN on Criteo. (**c**) DCN V2 on Avazu. (**d**) DCN V2 on Criteo. Compared with magnitude-based pruning, IASP assigns a smaller share to *device_id* on Avazu and distributes the budget more evenly across top fields on Criteo.

4.5. Ablation Study and Component Analysis

To quantify the contribution of each component and to compare the additive baseline importance with the conventional first-order Taylor criterion [20], we conducted an ablation study on DCN V2 with the Avazu dataset at 95% sparsity, the setting also used for the multi-seed analysis (Table 2). All variants share the pruning pipeline of Section 3.5 (mask granularity, sparsity schedule, weight rewinding, calibration batches, and training budget) and differ only in the scoring function. The Taylor variant scores each embedding parameter by $|w| \cdot |\partial \mathcal{L}/\partial w|$, with the gradient magnitude averaged over the calibration batches. The Taylor + Synergy & Gating variant multiplies this score by the gated synergy factor in Equation (6), using the γ, δ, and p of the full IASP framework (Table A1) without retuning. The dense baseline in Table 3 is the seed-42 dense model of Table 2. Each pruned configuration was run once with seed 42 in ablation runs conducted separately from the main experiments. Because some CUDA kernels used by PyTorch rely on atomic floating-point additions whose order is not fixed, training is not bitwise reproducible even with a fixed seed, and small numerical differences can change which parameters fall below the pruning threshold in later cycles; consequently, the absolute values of the pruned models differ from those of the main runs in Table 2 (e.g., AUC 0.7537 vs. 0.7563 for the full IASP framework). The variants are therefore compared within Table 3.

As shown in Table 3, all pruned variants improved on the dense baseline in both AUC ($+0.0044$ to $+0.0051$) and LogLoss (-0.0018 to -0.0026), whereas the differences among the variants were much smaller (at most 0.0007 in AUC and 0.0008 in LogLoss). The additive base and the Taylor criterion performed comparably (AUC 0.7536 vs. 0.7532; LogLoss 0.3806 vs. 0.3802). Adding synergy and gating to the Taylor score slightly increased the AUC (0.7535) but also increased the LogLoss (0.3808); since γ, δ, and p were tuned for the additive base, this variant shows only how these terms transfer without retuning. On the additive base, adding synergy without gating ($\tau = 0$) lowered the AUC to 0.7530, which is consistent with the motivation for gating, and the full IASP framework obtained the highest AUC (0.7537) and the lowest LogLoss (0.3800) among the variants.

These single-run differences are of the same order as the seed-to-seed standard deviation of IASP in Table 2 (0.0005 in AUC) and smaller than the AUC difference between the main and ablation runs of the full IASP framework with the same seed (0.0026). The latter difference also indicates that the three-seed standard deviation may underestimate

run-to-run variability, so we do not interpret the differences among the variants as evidence that one formulation is strictly superior. Rather, Table 3 indicates that the additive surrogate does not reduce accuracy relative to the Taylor criterion and that most of the gain over the dense model is already obtained with the base score alone, with the synergy and gating terms adding a smaller increment. We retain the additive form because it yields bounded scores and an explicit, tunable balance between magnitude and gradient information; the selected balance differs substantially across settings (e.g., $\beta = 0.0073$ for DCN V2 on Avazu versus $\beta = 0.7963$ for DCN V2 on Criteo; Table A1), which the standard product $|w| \cdot |g|$ does not provide.

Table 3. Ablation of the importance score and IASP components (DCN V2, Avazu, 95% sparsity). Test AUC and LogLoss. The dense baseline is the seed-42 dense model in Table 2; each pruned configuration was run once with seed 42 in ablation runs conducted separately from the main experiments. Due to non-deterministic GPU operations, the absolute values of the pruned models differ from those in Table 2, and the variants should be compared within this table. $|g|$ denotes the magnitude of $\partial\mathcal{L}/\partial w$ averaged over the calibration batches, and $\mathcal{N}(\cdot)$ is the min–max normalization in Equation (5). Synergy and gating use γ, δ, and p from Table A1.

Configuration	Formulation	AUC	LogLoss
Dense (Baseline)	-	0.7486	0.3826
Conventional Taylor [20]	Multiplicative: $\|w\| \cdot \|g\|$	0.7532	0.3802
Taylor + Synergy & Gating	Multiplicative: $\|w\| \cdot \|g\|$	0.7535	0.3808
Additive base only	Additive: $\alpha \cdot \mathcal{N}(\|w\|) + \beta \cdot \mathcal{N}(\|g\|)$	0.7536	0.3806
Additive + Synergy ($\tau = 0$)	Additive: $\alpha \cdot \mathcal{N}(\|w\|) + \beta \cdot \mathcal{N}(\|g\|)$	0.7530	0.3805
Full IASP	**Additive: $\boldsymbol{\alpha \cdot \mathcal{N}(\|w\|) + \beta \cdot \mathcal{N}(\|g\|)}$**	**0.7537**	**0.3800**

The full IASP framework, which obtained the highest AUC and the lowest LogLoss, is shown in bold.

5. Real-World RTB Deployment Analysis

A critical requirement for modern CTR prediction models is the ability to operate within the strict 20 ms SLA of OpenRTB environments [1]. To validate the practical engineering utility of the proposed framework, we measured inference latency on a high-end desktop processor (Intel Core i9-10980XE, 24.75 MB L3 cache) to emulate a production inference setting. Although single-thread latency is largely determined by per-core execution, cache hierarchies and memory bandwidth differ between this processor and server-class CPUs (e.g., Intel Xeon, AMD EPYC), so absolute latencies may vary across deployment hardware. Inference was restricted to a single CPU thread to approximate per-request processing in RTB servers and to examine whether the SLA can be met without GPU accelerators.

5.1. Theoretical Complexity and Inference Speedup

By physically reconfiguring the network architecture, the proposed IASP framework yields non-trivial latency reductions. Theoretically, since the cross-layer computational complexity of the original DCN scales linearly ($O(d)$) with respect to the feature dimension d, it achieves a linear speedup [4]. Conversely, DCN V2 involves heavy matrix–vector multiplications scaling quadratically ($O(d^2)$) [5], thereby benefiting from a quadratic speedup under structured dimension reduction.

In our implementation, the target sparsity is mapped to a uniform per-field embedding width for deployment: at 90%, 95%, and 99% target sparsity, each field retains its top-4, top-2, and top-1 dimensions (out of 16) ranked by the IASP score, and the corresponding rows and columns of each cross-layer matrix $\mathbf{W}_l$ are removed. These widths retain 25%, 12.5%, and 6.25% of the embedding parameters, respectively; the deployed models are therefore less sparse than the corresponding masked models. On Avazu (23 fields), the

input dimension is thus reduced from $d = 368$ to $k = 92$, 46, and 23; on Criteo (26 categorical fields × 16 plus 13 numerical features), it is reduced from $d = 429$ to $k = 117$, 65, and 39. Because over 99% of the parameters reside in the embedding tables, reducing the per-field width from 16 to 1 yields the approximately 16× memory reduction reported in Table 4 and reduces the cross-layer cost from $O(d^2)$ to $O(k^2)$.

Note that predictive accuracy (Figures 2 and 3) is evaluated on the models pruned at the target sparsity, whereas inference efficiency (Table 4) is measured on the corresponding uniform-width models extracted using the same importance scores.

To examine SLA compliance under a realistic final-ranking workload, Table 4 reports the inference latency, speedup factors, and memory footprints across both datasets and architectures. In industrial RTB pipelines, the final ranking phase typically evaluates a candidate pool of 100 to 300 items returned by the upstream matching stage. To reflect this setting, we set the inference batch size to 200. All efficiency measurements use the models at 99% target sparsity; single-request latency (batch size 1) is analyzed in Section 5.2.

Under this strict single-thread CPU execution, the dense DCN V2 model on the Avazu dataset satisfies the 20 ms SLA limit with a latency of 6.222 ms. By structurally reducing the parameter space, the uniform-width IASP model (99%, 1 dim/field) compresses the memory requirement to 36.24 MB (a 16-fold reduction) while cutting the overall latency down to 1.002 ms. While the $O(d^2) \rightarrow O(k^2)$ reduction specifically describes cross-layer arithmetic, our end-to-end deployment measurements provide empirical evidence of an actual inference speedup of 6.2× on Avazu and 6.6× on Criteo for the matrix-based DCN V2, compared with 2.4–2.5× for DCN, consistent with reduced computation and memory traffic. Note that masked models retain the original tensor shapes regardless of the importance criterion and therefore show no latency or memory reduction (Table 4); efficiency gains arise only after physical extraction.

Table 4. Comprehensive Inference Performance Across Architectures and Datasets (CPU Single-Thread, Batch = 200, 99% Target Sparsity).

Architecture & Dataset	Model State	Latency (ms)	Speedup	Memory (MB)
DCN (Avazu)	Dense (Baseline)	2.607	1.00×	577.27
	Masked (Mag./Grad.)	2.597	1.00×	577.27 *
	IASP (99%, 1 dim/field)	**1.031**	**2.5×**	**36.23**
DCN (Criteo)	Dense (Baseline)	2.781	1.00×	2061.30
	Masked (Mag./Grad.)	2.772	1.00×	2061.30 *
	IASP (99%, 1 dim/field)	**1.159**	**2.4×**	**129.00**
DCN V2 (Avazu)	Dense (Baseline)	6.222	1.00×	578.95
	Masked (Mag./Grad.)	6.217	1.00×	578.95 *
	IASP (99%, 1 dim/field)	**1.002**	**6.2×**	**36.24**
DCN V2 (Criteo)	Dense (Baseline)	7.629	1.00×	2063.41
	Masked (Mag./Grad.)	7.634	1.00×	2063.41 *
	IASP (99%, 1 dim/field)	**1.150**	**6.6×**	**129.01**

* Masked models are measured in the dense tensor layout used for serving. Rows for the uniform-width IASP models are shown in bold.

5.2. Single-Request Latency

To complement the batch-level measurement, Table 5 reports single-request inference (batch size 1) for the same models as in Table 4 at 99% target sparsity, on the same processor with single-thread execution, including the 95th- and 99th-percentile (P95 and P99) latencies. After 50 warm-up requests, test samples were scored one at a time over 5 sessions of 1000 runs each. All models had stable mean latencies of 0.40–0.49 ms and P99 latencies of at most 0.55 ms, remaining well below the 20 ms SLA limit. At this batch size, however,

the speedup of the uniform-width IASP models over the dense models was marginal (0.99–1.13×), compared with 2.4–6.6× at batch size 200 (Table 4). The dense DCN and DCN V2 models on Avazu also differed by only 0.010 ms in mean latency (0.427 vs. 0.437 ms), whereas they differed by 3.6 ms at batch size 200. This suggests that single-request latency is dominated by costs that do not scale with the embedding width, such as PyTorch framework overhead and per-field embedding lookups, rather than by cross-layer matrix arithmetic. The computational benefit of IASP therefore becomes apparent when each request scores a larger candidate pool, as is typical in the final ranking stage of RTB pipelines: based on the batch-200 latencies in Table 4, one core can score approximately 199,600 candidates per second (200/1.002 ms) with the IASP model for DCN V2 on Avazu, compared with approximately 32,100 (200/6.222 ms) with the dense model.

Table 5. Single-request inference performance (CPU single-thread, batch size 1, 99% target sparsity). Per-request latency is reported as its mean and its 95th and 99th percentiles (P95 and P99); QPS is the number of requests processed per second on one core; speedup is the ratio of the mean latency of the dense model to that of each model. Rows for the uniform-width IASP models are shown in bold.

Architecture & Dataset	Model State	Latency (ms)			QPS	Speedup
		Mean	P95	P99		
DCN (Avazu)	Dense (Baseline)	0.427	0.444	0.463	2342	1.00×
	Masked (Mag./Grad.)	0.425	0.433	0.452	2354	1.00×
	IASP (99%, 1 dim/field)	**0.430**	**0.441**	**0.486**	**2325**	**0.99×**
DCN (Criteo)	Dense (Baseline)	0.462	0.469	0.510	2164	1.00×
	Masked (Mag./Grad.)	0.465	0.471	0.495	2150	0.99×
	IASP (99%, 1 dim/field)	**0.451**	**0.457**	**0.479**	**2217**	**1.02×**
DCN V2 (Avazu)	Dense (Baseline)	0.437	0.451	0.514	2290	1.00×
	Masked (Mag./Grad.)	0.435	0.453	0.468	2296	1.00×
	IASP (99%, 1 dim/field)	**0.397**	**0.403**	**0.426**	**2522**	**1.10×**
DCN V2 (Criteo)	Dense (Baseline)	0.488	0.495	0.548	2049	1.00×
	Masked (Mag./Grad.)	0.486	0.492	0.525	2056	1.00×
	IASP (99%, 1 dim/field)	**0.433**	**0.438**	**0.461**	**2311**	**1.13×**

5.3. Cache Efficiency and Throughput

Beyond pure arithmetic speed, memory bandwidth remains a primary bottleneck (the von Neumann bottleneck) during embedding lookups in production environments. For DCN V2, the uniform-width IASP model (99%, 1 dim/field) reduces the memory footprint from 578.95 MB to 36.24 MB on Avazu and from 2063.41 MB to 129.01 MB on Criteo. Although the Avazu model still exceeds the 24.75 MB L3 cache of the tested processor, it is brought to the same order of magnitude as the L3 caches of server-class CPUs, which can raise the hit rate for frequently accessed embedding rows and reduce DRAM traffic; for Criteo, the expected benefit lies mainly in reduced memory traffic. Although we did not measure cache-miss rates directly, a reduction in DRAM traffic would be consistent with the lower batch-level latencies in Table 4. The smaller footprint also allows more model replicas to be served concurrently on a server node, which can increase the sustainable queries per second (QPS) per node.

6. Conclusions

In this paper, we proposed IASP, a novel model compression framework specifically designed to address the stringent latency constraints (SLA $<$ 20 ms) and memory bottlenecks of RTB systems. Unlike conventional pruning criteria that score parameters without regard to feature interactions, IASP combines a Taylor-inspired importance score (weight magnitude and gradient sensitivity) with cross-layer interaction statistics, and its data-

adaptive gating mechanism restricts the synergy amplification to dimensions with strong cross-layer participation. Evaluations on the Avazu and Criteo datasets show that IASP can reach sparsity levels of up to 99% while maintaining predictive accuracy, consistent with a regularization effect against overfitting. For DCN V2 on Avazu at 95% sparsity, IASP's advantage over the magnitude- and gradient-based criteria was consistent across three random seeds, with IASP obtaining the highest AUC and the lowest LogLoss in every seed. An ablation on DCN V2 (Avazu) further showed that the additive importance score performs comparably to the conventional Taylor criterion and that the full framework with gating obtained the highest AUC and the lowest LogLoss among the tested variants, although the differences among the variants were small.

Moreover, by physically removing embedding dimensions rather than merely masking weights, the uniform-width IASP models reduce the memory footprint by approximately 16×, which brings the model size closer to CPU cache scales and is expected to reduce memory traffic. They achieve single-thread CPU inference speedups of 6.2–6.6× for DCN V2 (2.4–2.5× for DCN) when ranking 200 candidates per request, whereas the speedup for single-candidate requests is marginal (0.99–1.13×). These results suggest that heavy CTR models can be served on standard CPU servers with higher throughput and lower cost. This study has several limitations. First, the importance score relies on a first-order approximation and neglects second-order curvature (Section 3.2). Second, results over multiple seeds are reported only for DCN V2 on Avazu at 95% sparsity; the other configurations, including the ablation study, are single runs owing to computational constraints. Third, this study compares importance criteria within a shared pruning pipeline and does not include search-based or threshold-learning methods such as AutoDim and PEP as experimental baselines. Finally, predictive accuracy was evaluated on the masked models, whereas efficiency was measured on the corresponding uniform-width models on a single desktop-class CPU with batch sizes of 1 and 200. Future work will extend IASP to field-wise adaptive widths that allocate different numbers of dimensions to each field, curvature-aware importance estimates, attention-based interaction architectures (e.g., DIN, AutoInt), and online pruning for non-stationary streaming data, as well as compare IASP with learned-threshold and search-based methods (e.g., PEP, AutoDim) under a unified protocol that accounts for both accuracy and deployable inference cost.

Author Contributions: Conceptualization, B.H. and C.O.K.; methodology, B.H.; software, B.H.; validation, B.H.; formal analysis, B.H.; investigation, B.H.; writing—original draft preparation, B.H.; writing—review and editing, C.O.K.; supervision, C.O.K.; funding acquisition, C.O.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by National Research Foundation of Korea (NRF) grants funded by the Korean government (MSIT) (No. RS-2025-23523836 and No. RS-2025-00513417).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets analyzed in this study are publicly available: Avazu at https://www.kaggle.com/c/avazu-ctr-prediction (accessed on 12 September 2026) and Criteo at https://www.kaggle.com/c/criteo-display-ad-challenge (accessed on 12 September 2026).

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Reproducibility Details and Optimal Hyperparameters

Unless otherwise stated, results are from a single run with seed 42; the multi-seed comparison in Section 4.2 (Table 2) uses seeds 42, 43, and 44, with the dense baseline and all pruned models retrained for each seed. Hyperparameters were selected by maximizing

validation AUC on the validation split, and all reported metrics are computed on the test split. The IASP hyperparameters were optimized with the Optuna TPE sampler over $\alpha, \beta, \gamma, \delta \sim \text{Uniform}[0.0, 1.0]$ and the gating percentile $p \sim \text{Categorical}[p_{50}, p_{60}, p_{70}, p_{80}, p_{90}]$, with 20 trials per dataset and architecture. All models used an embedding dimension of 16, three cross layers, and a three-layer deep network, and they were trained with the Adam optimizer (learning rate 0.001, weight decay 10^{-5}, batch size 8192). Dense baselines were trained with early stopping (patience of 5 epochs) and reported at their best validation checkpoint. For pruning, each model was first trained for one warm-up epoch; the protocol then ran for $T = 10$ cycles with one fine-tuning epoch per cycle, and importance statistics were accumulated over 150 calibration mini-batches at each cycle. Avazu was split chronologically using its hour field, and Criteo, which provides no timestamp, was split by its original record order (8:1:1).

Notably, for DCN V2 on Avazu, Optuna selected a small gradient weight ($\beta = 0.0073$) relative to the magnitude weight ($\alpha = 0.5733$), suggesting that raw gradients are less informative under extreme sparsity in this setting, whereas $\beta \approx 0.80$ was selected for DCN V2 on Criteo. The appropriate balance between the two signals is therefore data-dependent.

Table A1. Hyperparameters Selected by Optuna.

Architecture & Dataset	α	β	γ	δ	p
DCN (Avazu)	0.8273	0.6012	0.7789	0.0179	p_{70}
DCN (Criteo)	0.3116	0.5994	0.3387	0.0468	p_{70}
DCN V2 (Avazu)	0.5733	0.0073	0.2639	0.2988	p_{70}
DCN V2 (Criteo)	0.6735	0.7963	0.7841	0.8679	p_{70}

Table A2. Test AUC and LogLoss underlying Figures 2 and 3 (single run, seed 42). Bold indicates the best value within each setting and column.

Setting	Model	AUC (↑)			LogLoss (↓)		
		90%	95%	99%	90%	95%	99%
DCN (Avazu)	Dense		0.7183			0.4205	
	Magnitude	0.7475	0.7498	0.7553	0.3859	0.3841	0.3789
	Gradient	0.7484	0.7518	0.7562	0.3863	0.3842	0.3785
	IASP	**0.7553**	**0.7566**	**0.7563**	**0.3789**	**0.3784**	**0.3783**
DCN (Criteo)	Dense		0.7997			0.4511	
	Magnitude	0.8051	0.8050	0.8053	0.4451	0.4450	0.4447
	Gradient	0.8051	0.8053	0.8064	0.4452	0.4446	0.4435
	IASP	**0.8067**	**0.8064**	**0.8066**	**0.4432**	**0.4434**	**0.4434**
DCN V2 (Avazu)	Dense		0.7486			0.3826	
	Magnitude	0.7462	0.7477	0.7555	0.3869	0.3870	0.3788
	Gradient	0.7473	0.7489	**0.7571**	0.3868	0.3858	**0.3777**
	IASP	**0.7567**	**0.7563**	0.7567	**0.3781**	**0.3784**	0.3778
DCN V2 (Criteo)	Dense		0.7999			0.4495	
	Magnitude	0.8055	0.8050	0.8075	0.4457	0.4466	0.4426
	Gradient	0.8057	0.8054	**0.8089**	0.4454	0.4455	**0.4413**
	IASP	**0.8088**	**0.8086**	0.8086	**0.4412**	**0.4415**	0.4417

↑ indicates that higher is better, and ↓ indicates that lower is better.

References

1. Wang, J.; Zhang, W.; Yuan, S. Display Advertising with Real-Time Bidding (RTB) and Behavioural Targeting. *Found. Trends Inf. Retr.* **2017**, *11*, 297–435. [CrossRef]

2. Jiang, B.; Deng, C.; Yi, H.; Hu, Z.; Zhou, G.; Zheng, Y.; Huang, S.; Guo, X.; Wang, D.; Song, Y.; et al. XDL: An Industrial Deep Learning Framework for High-dimensional Sparse Data. In *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data (DLP-KDD)*; Association for Computing Machinery: New York, NY, USA, 2019.
3. Rendle, S. Factorization Machines. In *2010 IEEE International Conference on Data Mining*; IEEE: Piscataway, NJ, USA, 2010.
4. Wang, R.; Fu, B.; Fu, G.; Wang, M. Deep & Cross Network for Ad Click Predictions. In *Proceedings of the ADKDD*; Association for Computing Machinery: New York, NY, USA, 2017.
5. Wang, R.; Shivanna, R.; Cheng, D.; Jain, S.; Lin, D.; Hong, L.; Chi, E. DCN V2: Improved Deep & Cross Network and Practical Lessons for Web-scale Learning to Rank Systems. In *Proceedings of the Web Conference 2021*; Association for Computing Machinery: New York, NY, USA, 2021.
6. Zhou, G.; Zhu, X.; Song, C.; Fan, Y.; Zhu, H.; Ma, X.; Yan, Y.; Jin, J.; Li, H.; Gai, K. Deep Interest Network for Click-Through Rate Prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*; Association for Computing Machinery: New York, NY, USA, 2018.
7. Zhou, G.; Mou, N.; Fan, Y.; Pi, Q.; Bian, W.; Zhou, C.; Zhu, X.; Gai, K. Deep Interest Evolution Network for Click-Through Rate Prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*; Association for the Advancement of Artificial Intelligence: Washington, DC, USA, 2019.
8. Song, W.; Shi, C.; Xiao, Z.; Duan, Z.; Xu, Y.; Zhang, M.; Tang, J. AutoInt: Automatic Feature Interaction Learning via Self-Attentive Neural Networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*; Association for Computing Machinery: New York, NY, USA, 2019.
9. Gupta, U.; Wu, C.J.; Wang, X.; Naumov, M.; Reagen, B.; Brooks, D.; Cottel, B.; Hazelwood, K.; Hempstead, M.; Jia, B.; et al. The Architectural Implications of Facebook's DNN-based Personalized Recommendation. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*; IEEE: Piscataway, NJ, USA, 2020.
10. Naumov, M.; Mudigere, D.; Shi, H.J.M.; Huang, J.; Sundaraman, N.; Park, J.; Wang, X.; Gupta, U.; Wu, C.-J.; Azzolini, A.G.; et al. Deep Learning Recommendation Model for Personalization and Recommendation Systems. *arXiv* **2019**, arXiv:1906.00091.
11. Cheng, H.T.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.S.; Chai, W.; Ispir, M.; et al. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS)*; Association for Computing Machinery: New York, NY, USA, 2016.
12. Guo, H.; Tang, R.; Ye, Y.; Li, Z.; He, X. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. In *IJCAI'17: Proceedings of the 26th International Joint Conference on Artificial Intelligence*; AAAI Press: Washington, DC, USA, 2017.
13. Lian, J.; Zhou, X.; Zhang, F.; Chen, Z.; Xie, X.; Sun, G. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*; Association for Computing Machinery: New York, NY, USA, 2018.
14. Xiao, J.; Ye, H.; He, X.; Zhang, H.; Wu, F.; Chua, T.S. Attentional Factorization Machines: Learning the Weight of Feature Interactions via Attention Networks. In *IJCAI'17: Proceedings of the 26th International Joint Conference on Artificial Intelligence*; AAAI Press: Washington, DC, USA, 2017.
15. Gholami, A.; Kim, S.; Dong, Z.; Yao, Z.; Mahoney, M.W.; Keutzer, K. A Survey of Quantization Methods for Efficient Neural Network Inference. *arXiv* **2021**, arXiv:2103.13630.
16. Hinton, G.; Vinyals, O.; Dean, J. Distilling the Knowledge in a Neural Network. *arXiv* **2015**, arXiv:1503.02531.
17. Zhao, X.; Liu, H.; Liu, H.; Tang, J.; Guo, W.; Shi, J.; Wang, S.; Gao, H.; Long, B. AutoDim: Field-aware Embedding Dimension Search in Recommender Systems. In *Proceedings of the Web Conference 2021*; Association for Computing Machinery: New York, NY, USA, 2021.
18. Liu, S.; Gao, C.; Chen, Y.; Jin, D.; Li, Y. Learnable Embedding Sizes for Recommender Systems. In Proceedings of the ICLR, Virtual Event, Austria, 3–7 May 2021.
19. Lyu, F.; Tang, X.; Zhu, H.; Guo, H.; Zhang, Y.; Tang, R.; Liu, X. OptEmbed: Learning Optimal Embedding Table for Click-through Rate Prediction. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*; Association for Computing Machinery: New York, NY, USA, 2022.
20. Molchanov, P.; Tyree, S.; Karras, T.; Aila, T.; Kautz, J. Pruning Convolutional Neural Networks for Resource Efficient Inference. In Proceedings of the ICLR, Toulon, France, 24–26 April 2017.
21. Han, S.; Mao, H.; Dally, W.J. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. In Proceedings of the ICLR, San Juan, Puerto Rico, 2–4 May 2016.
22. Frankle, J.; Carbin, M. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In Proceedings of the ICLR, New Orleans, LA, USA, 6–9 May 2019.
23. Wen, W.; Wu, C.; Wang, Y.; Chen, Y.; Li, H. Learning Structured Sparsity in Deep Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Barcelona, Spain, 5–10 December 2016.
24. Liu, Z.; Li, J.; Shen, Z.; Huang, G.; Yan, S.; Zhang, C. Learning Efficient Convolutional Networks through Network Slimming. In *2017 IEEE International Conference on Computer Vision (ICCV)*; IEEE: Piscataway, NJ, USA, 2017.

25. Shi, H.-J.M.; Mudigere, D.; Naumov, M.; Yang, J. Compositional Embeddings Using Complementary Partitions for Memory-Efficient Recommendation Systems. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*; Association for Computing Machinery: New York, NY, USA, 2020.
26. Li, X.; Zhou, X.; Xiao, Y.; Huang, P.; Chen, D.; Chen, S.; Xian, Y. AutoFAS: Automatic Feature and Architecture Selection for Pre-Ranking System. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*; Association for Computing Machinery: New York, NY, USA, 2022.
27. Yu, H.; Fu, M.; Ding, J.; Zhou, Y.; Wu, J. Unified Low-rank Compression Framework for Click-through Rate Prediction. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*; Association for Computing Machinery: New York, NY, USA, 2024.
28. Wang, K.; Wang, H.; Song, K.; Guo, W.; Cheng, K.; Li, Z.; Liu, Y.; Lian, D.; Chen, E. A Universal Framework for Compressing Embeddings in CTR Prediction. *arXiv* **2025**, arXiv:2502.15355.
29. Chen, G.; Xia, L.; Huang, C. LightGNN: Simple Graph Neural Network for Recommendation. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*; Association for Computing Machinery: New York, NY, USA, 2025; pp. 549–558.
30. Kasalický, P.; Spišák, M.; Vančura, V.; Bohuněk, D.; Alves, R.; Kordík, P. The Future is Sparse: Embedding Compression for Scalable Retrieval in Recommender Systems. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*; Association for Computing Machinery: New York, NY, USA, 2025.
31. Li, S.; Guo, H.; Tang, X.; Tang, R.; Hou, L.; Li, R.; Zhang, R. Embedding Compression in Recommender Systems: A Survey. *ACM Comput. Surv.* **2024**, *56*, 1–38. [CrossRef]
32. Zhang, Z.-Y.; Sheng, X.R.; Zhang, Y.; Jiang, B.; Han, S.; Deng, H.; Zheng, B. Towards Understanding the Overfitting Phenomenon of Deep Click-Through Rate Prediction Models. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*; Association for Computing Machinery: New York, NY, USA, 2022.
33. Liu, B.; Zhu, C.; Li, G.; Zhang, W.; Lai, J.; Tang, R.; He, X.; Li, Z.; Yu, Y. AutoFIS: Automatic Feature Interaction Selection in Factorization Models for Click-Through Rate Prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*; Association for Computing Machinery: New York, NY, USA, 2020.
34. Wang, Y.; Zhao, X.; Xu, T.; Wu, X. AutoField: Automating Feature Selection in Deep Recommender Systems. In *Proceedings of the ACM Web Conference 2022*; Association for Computing Machinery: New York, NY, USA, 2022.
35. Lin, W.; Zhao, X.; Wang, Y.; Xu, T.; Wu, X. AdaFS: Adaptive Feature Selection in Deep Recommender System. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*; Association for Computing Machinery: New York, NY, USA, 2022.
36. Qu, L.; Ye, Y.; Tang, N.; Zhang, L.; Shi, Y.; Yin, H. Single-shot Embedding Dimension Search in Recommender System. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*; Association for Computing Machinery: New York, NY, USA, 2022.
37. Li, J.; Xu, H.; Zhang, S. Adaptive kNN graph model. *Nat. Commun.* **2026**, *17*, 7491. [CrossRef] [PubMed]
38. Li, J.; Song, J.; Shi, J.; Xu, H.; Yu, H.; Chen, G.; Zhang, S. Robust quantum feature selection with sparse optimization circuit. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2025**, *44*, 2613–2626. [CrossRef]
39. LeCun, Y.; Denker, J.; Solla, S. Optimal Brain Damage. In *Advances in Neural Information Processing Systems (NeurIPS)*; Morgan Kaufmann: San Mateo, CA, USA, 1989.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.