MICE: Minimal Interaction Cross-Encoders for efficient Re-ranking

Anonymous ACL submission

Abstract

In Information Retrieval (IR), cross-encoders deliver state-of-the-art ranking effectiveness but have a high inference cost, limiting their use to second-stage re-rankers. Prior work has addressed this bottleneck from two largely separate directions: accelerating cross-encoder inference through attention sparsification, or improving first-stage retrieval effectiveness to alleviate the need of a re-ranker, using more complex models, e.g. late-interactions. In this work, we bridge these two directions through an in-depth analysis of cross-encoder internal mechanisms. By identifying and removing superfluous interactions, we derive MICE (Minimal Interaction Cross-Encoders), a new cross-encoder architecture that retains effectiveness while reducing computational overhead. Extensive evaluations on both in-domain and out-of-domain datasets demonstrate that MICE matches or exceeds its cross-encoder counterpart in effectiveness, while reducing FLOPs down to 2.5 times.[1]

1 Introduction

Since the advent of transformers (Vaswani et al., 2017), many neural Information Retrieval (IR) architectures have been proposed, from representation-based bi-encoders to interaction-based cross-encoders, with dense or sparse, single- or multi-vector representations. Although cross-encoders offer state-of-the-art ranking performance, they are computationally prohibitive when applied exhaustively to large corpora (Yates et al., 2021; Nogueira and Cho, 2019). This bottleneck has maintained the prevalent *retrieve-and-rerank* paradigm, where a fast initial retriever (Karpukhin et al., 2020; Robertson et al., 1994) filters the corpus down to a candidate pool that is then re-ranked by a cross-encoder (Nogueira and Cho, 2019).

[1]Anomymized code is made available at https://anonymous.4open.science/r/cross-encoders-7EB8

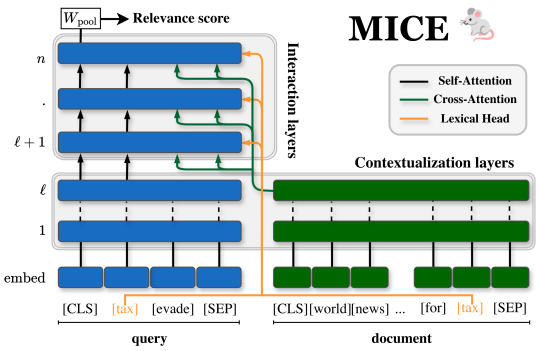

Figure 1: MICE architecture. Keeps the strict minimum interactions in a cross-encoder to maintain effectiveness.

However, the *retrieve-and-rerank* paradigm introduces fundamental limitations. First, re-ranking can only be as good as its input: documents missed by the first-stage retriever cannot be recovered. Second, the re-ranking step is still affected by the low efficiency of cross-encoders.

The first limitation is addressed by architectures that improve effectiveness while remaining efficient enough for full-corpus search. A representative class of models are late-interaction models, such as ColBERT (Khattab and Zaharia, 2020), that keep a token level representation of documents and queries (like cross-encoders), but delay (only contextualized representations) and simplify (MaxSim) their interactions. Late-interaction models remain less effective than cross-encoders. The second limitation has been approached by cutting cross-encoder inference cost by architectural optimizations such as sparse attention (Schlatt et al., 2024), or by restricting the candidate pool via pruning / cascade strategies (Meng et al., 2024; Campagnano et al., 2025). Still, such methods remain computationally heavier than late-interaction models.

Our work connects these two lines of research by deriving an efficient, late-interaction-style re-ranker directly from a conventional cross-encoder. Specifically, we pursue two main goals. First, we propose a principled masking strategy that strips un-

necessary cross-encoder interactions identified by prior interpretability studies (Lu et al., 2025; Zhan et al., 2020), reducing inference cost to below that of ColBERT while preserving ranking effectiveness (Section 3). Second, we use these findings to reshape the cross-encoder architecture into a more efficient but as effective re-ranking model. We call this new architecture MICE (Minimal-Interaction Cross-Encoder, described in Section 4 and depicted in Figure 1). We demonstrate that MICE does not degrade the ranking performance compared to the initial cross-encoder, while being up to $2.5\times$ more efficient, on two distinct backbones, BERT (Devlin et al., 2019) and the more recent ModernBERT (Warner et al., 2024).

We address the following research questions:

RQ 1 How many interactions can be removed to improve the efficiency of a cross-encoder, while maintaining its effectiveness?

RQ 2 Can we design a more efficient architecture, while maintaining cross-encoder effectiveness?

2 Related Works

More efficient ranking paradigms Cross-encoder models (Nogueira and Cho, 2019) are highly effective, but their poor efficiency has led IR practitioners to explore more efficient alternatives. Bi-encoders encode queries and documents separately into single vectors (Karpukhin et al., 2020). This enables offline document indexing, making bi-encoders very efficient first-stage retrievers, at the cost of reduced effectiveness, particularly in out-of-domain (OOD) scenarios (Rosa et al., 2022; Thakur et al., 2021), as the model cannot explicitly capture query-document interactions. Learned Sparse Retrieval models, such as SPLADE (Formal et al., 2021), improve the effectiveness of dense bi-encoders while retaining their efficiency due to their sparse nature and inverted indexes.

In contrast with bi-encoders, late-interaction models, such as ColBERT (Khattab and Zaharia, 2020), encode queries and passages into multiple vectors (one per token) and aggregate the score of each query token by computing the maximum similarity (*MaxSim* operator) with a document token. This yields greater effectiveness at the cost of increased storage, partially addressed by follow-up works (Santhanam et al., 2022b,a).

Several hybrid architectures have been proposed to bridge the efficiency-effectiveness gap between bi-encoders and cross-encoders. Poly-encoders (Humeau et al., 2020) summarize documents into M context vectors to reduce self-attention cost, but only match cross-encoder effectiveness at high M values ($M = 360$, exceeding the average MS MARCO document length).

Mid-fusion transformers (Tan and Bansal, 2019) encode each input stream independently in the lower layers before fusing their representations in the upper layers, an idea later adapted to text pairs by DeFormer (Cao et al., 2020) to enable offline passage encoding. PreTTR (MacAvaney et al., 2020) apply this idea to IR, while MORES (Gao et al., 2020) further disentangle document and query processing into two distinct encoders before combining them with an interaction module. This design enables offline document encoding, but hybrid architectures struggle to match the effectiveness of cross-encoders.

Instead of modifying the cross-encoder paradigm, several works have aimed at improving the efficiency of re-rankers, without altering their fundamental architecture.

Improving the efficiency of cross-encoders A first line of research targets the self-attention mechanism (Lin et al., 2017), which underpins transformer representations but slows inference. General approaches such as linear attention (Wang et al., 2020; Wu et al., 2021) have been proposed to reduce its complexity, yet they generally transfer poorly to IR, where attention plays a central role in detecting semantic and lexical matches between query and document tokens (Lu et al., 2025). Sparse attention models such as Longformer (Beltagy et al., 2020) and BigBird (Zaheer et al., 2020) restrict token interactions to local windows, under the assumption that not all pairwise interactions are necessary. (Schlatt et al., 2024) successfully applied this principle to cross-encoders with limited effectiveness loss. A second line of research focuses on reducing model size through knowledge distillation (Hinton et al., 2015) or pruning (Frankle and Carbin, 2019; Campos et al., 2023), with applications to IR rankers (Lei et al., 2025; Schlatt et al., 2025).

Rather than optimizing existing cross-encoders or replacing them entirely, our methodology progressively strips cross-encoders of unnecessary interactions to improve their efficiency (see section 3). Pushed to its limit, this process naturally leads to a new architecture, namely MICE (see Section 4).

3 Towards minimal interaction cross-encoders

In this section, we first study which interactions are truly necessary within a cross-encoder by analyzing the impact of masking interactions between input segments: [CLS], query Q, document D, and [SEP] tokens.

3.1 Background

Cross-encoders classify a query-document (Q, D) couple. Their input is typically composed of the sequence of tokens: [CLS] $q_1 \dots q_n$ [SEP1] $d_1 \dots d_m$ [SEP2] (n query tokens and m document tokens). In this work we focus on encoder-only cross-encoders, as this architecture has been shown to be the best choice for sequence classification task (Weller et al., 2025). The relevance score for the document *w.r.t.* the query is predicted by applying a classification head to the token representation [CLS] of the last transformer layer L.

Cross-encoders use self-attention to capture *interaction* signals between two tokens a and b. In IR, these interactions have a very specific meaning when token a belongs to the query and token b to the document (or reciprocally). By moving information from token a and comparing it with what is already encoded in token b (its identity, the semantics of its context, etc.), self-attention is key to detecting *matching signals* between queries and documents.

For that reason, studies attempting to reverse-engineer the inner working of cross-encoders particularly focus on interpreting the interactions between the different input parts through the self-attention. For instance, Lu et al. (2025) show that it allows cross-encoders to detect, not only exact matching signals – what a lexical retriever like BM25 (Robertson et al., 1994) would do – but also semantic matching signals. Zhan et al. (2020) show that the relevance prediction process inside cross-encoders is decomposed into multiple consecutive stages. In the first layers, the model contextualizes query and document tokens. At this stage, query-document interactions only play a minor role, but once their semantics have been properly encoded, the model starts using query-document interaction signals. Lu et al. (2025) further provide empirical evidence that matching signals are captured by the self-attention, in so-called *matching heads*, and then aggregated inside the query tokens by *contextual query representation heads*. Ultimately, *relevance scoring heads* scan query tokens to retrieve relevance information and to encode it in the [CLS] representation for the final prediction. Their findings suggest that information does not flow freely between input parts inside cross-encoders, but instead roughly flows from the document tokens towards the query tokens (after contextualization), and then towards the [CLS] (Zhan et al., 2020). In the following, we denote input parts as $X, Y \in \{$[CLS], Q, [SEP1], D, [SEP2]$\}$ and $Y \leftarrow X$ (resp. $Y \not\leftarrow X$) a transfer of information (resp. blocking the transfer) from a token in X *to* a token in Y through the self-attention mechanism (or equivalently that Y *attends to* X).

3.2 Our approach

We measure the importance of a given interaction with causal analysis: by evaluating the effect of blocking it on the model's effectiveness. In practice, we prevent interactions by *masking* the corresponding block in the self-attention weight matrices, i.e., setting its logits to $-\infty$ before *Softmax*, effectively blocking any information transfer. Following prior description of information flow between input parts in cross-encoders (Zhan et al., 2020; Lu et al., 2025), we consider four cumulative masking steps, all summarized in Figure 2. For instance, Mask 2 blocks information transfers from the query Q to the document D ($D \not\leftarrow Q$), while also applying Mask 0 and Mask 1.

Masking Step 0 We block all interactions towards [SEP] and from [CLS] to other input parts, while dedicating [SEP1] and [SEP2] as attention sinks respectively for Q and D. This step is expected to have little to no impact on effectiveness, as it merely reduces noise received by query and document tokens. *Mask 0:* [SEP] $\not\leftarrow \{$[CLS], $Q, D\}$, $\{Q,$ [SEP], $D\} \not\leftarrow$ [CLS], $Q \not\leftarrow$[SEP2] *and* $D \not\leftarrow$[SEP1].

Masking Step 1 We block the flow of information from the document to [CLS] ([CLS] $\not\leftarrow \{D,$[SEP2]$\}$), motivated by evidence that relevance signals are stored in query tokens before being passed to [CLS] (Lu et al., 2025). This step is expected to have only marginal impact on effectiveness. *Mask 1: Mask 0 and* [CLS] $\not\leftarrow \{D,$[SEP2]$\}$.

Masking Step 2 We mask the query-to-document flow ($D \not\leftarrow Q$) across all layers, with limited expected effectiveness drop. This is a first step towards separating query and document contextualization, enabling offline document encoding. *Mask*

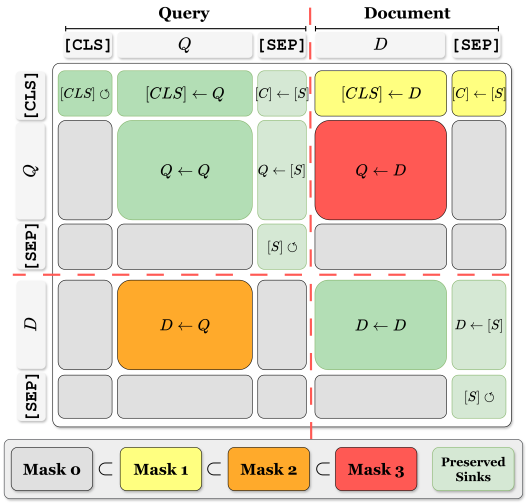

Figure 2: Masking approach. Interactions between input parts ([CLS], Q, D, [SEP]) are blocked using cumulative masking. Colors indicate the step where masking begins, ending with Mask 3 in a complete $Q \not\leftrightarrow D$ separation (block-diagonal structure). Green blocks denote permanently preserved interactions and attention sinks.

2: Mask 1 and $D \not\leftarrow Q$

Masking Step 3 We additionally block document-to-query interactions ($Q \not\leftarrow D$) across the first ℓ^* layers, fully separating query and document contextualization in the early layers. ℓ^* is the highest value that preserves the base model's effectiveness. *Mask 3: Mask 2 and $Q \not\leftarrow D$ (up to layer ℓ^*)*

Appendix B.1 provides further details on the motivation behind these masking steps. With this approach, the set of possible interactions decreases after each step, until we obtain the minimal set required to maintain the original cross-encoder effectiveness.

3.3 Experimental setup

Backbones We consider two distinct backbones: BERT (Devlin et al., 2019), which has been extensively studied (Rogers et al., 2020; Ferrando et al., 2024), and ModernBERT (Warner et al., 2024), a recent update to the original BERT architecture with stronger capabilities.While our approach can be applied to analyze any cross-encoder, we focus here on cross-encoders based on small transformer encoders: MiniLM-v2 (Wang et al., 2021) (simply "MiniLM" in the paper), a compact yet effective model based on BERT (Devlin et al., 2019) optimized via deep self-attention distillation, and Ettin-32M (Weller et al., 2025), based on ModernBERT (Warner et al., 2024). We further detail their configuration in Appendix (Table 3). Small models let us run the extensive training and evaluation of our masking study, which would be prohibitively expensive on larger backbones. Despite the rise of LLMs in IR (Ma et al., 2024), they remain competitive for ranking thanks to their efficiency (Déjean et al., 2024).

Table 1: Re-ranking evaluation results over 1k docs/query from BM25 in nDCG@10 (over 5 seeds) for the masking experiment (Section 3), comparing models fine-tuned with and w/o masking. **Bold** marks the best value per backbone, "_" second best.

		In-domain			Average	
	Re-Ranker	MSM	DL19	DL20	**ID**	**BEIR**
	BM25	23.0	51.2	47.7	42.5	43.0
MiniLM	Sparse CE	44.1	**74.3**	70.7	63.0	44.9
	PreTTR (ℓ4)	44.5	73.8	71.9	63.4	49.2
	Baseline	44.7	73.8	72.9	63.8	44.7
	Mask 0	44.6	73.8	**73.3**	63.9	46.3
	Mask 1	**44.8**	73.8	**73.3**	**64.0**	**50.2**
	Mask 2	44.1	73.1	71.8	63.0	49.0
	Mask 3-ℓ4	43.9	73.0	70.9	62.6	47.2
Ettin-32M	Baseline	**43.1**	70.3	**69.7**	**61.0**	48.3
	Mask 0	42.8	70.0	**69.7**	60.8	47.8
	Mask 1	42.6	**70.8**	69.2	60.9	**48.7**
	Mask 2	42.3	69.6	68.5	60.1	46.7
	Mask 3-ℓ6	41.9	69.4	68.6	59.9	42.8

Baselines We consider two baselines. Sparse CE (Schlatt et al., 2024) that sparsifies cross-encoder attention by blocking document-to-query interactions ($Q \not\leftarrow D$), the direction identified as most important by previous work (Zhan et al., 2020; Lu et al., 2025), yet preserves effectiveness when fine-tuned with this mask, making it a strong reference for validating our design choices. We do not reproduce its sliding-window attention over document tokens, as this is orthogonal to our work and detrimental for small windows. We also compare with PreTTR (MacAvaney et al., 2020), which implements mid-fusion by blocking all query-document interactions in the early layers, differing from our approach in the later layers where it keeps all interactions. We only reproduce its independent query-document contextualization across the first ℓ layers, without the Auto-Encoder compression — reported results thus constitute an upper bound for PreTTR.

3.4 Results and Analysis

We fine-tune pretrained models with the masks on the re-ranking task (detailed setup in Appendix A.3), learning a separate cross-encoder for each mask and for the baselines of Section 3.3. Table 1 reports the average nDCG@10 on 5 ran-

dom seeds across ID — MS MARCO (Bajaj et al., 2016), TREC-DL19 and 20 (Craswell et al., 2020, 2021)— and OOD —BEIR (Thakur et al., 2021)— described in Appendix A.5.

First, we note that fine-tuning these checkpoints with Mask 0, either matches the unmasked baselines' effectiveness (for Ettin) or exceeds them (for MiniLM, especially in OOD: 46.3 vs 44.7 nDCG@10), confirming that the role of the [SEP] tokens is not directly tied to relevance prediction.

For both backbones, Table 1 shows that further masking up to Mask 2 either exceeds or matches the performance of the fine-tuned baselines, both ID and OOD (+5.5 nDCG@10 in average on OOD for Mask 1 on MiniLM). With this masking strategy, MiniLM reaches the best performances, both for ID and OOD. We can draw two conclusions from that: (i) it confirms that [CLS] does not need to attend to the document to receive the appropriate relevance signals (Mask 1). In contrast, OOD results suggest that cross-encoders capture *spurious* correlations when allowing this transfer of information; and (ii) Using Mask 2 ($D \not\leftarrow Q$) across all layers does not harm cross-encoders effectiveness, while it potentially enables substantial gain in efficiency. This also contradicts the masking strategy of Sparse CE (Schlatt et al., 2024) ($Q \not\leftarrow D$) as we observe no improvement on the effectiveness compared to the unmasked MiniLM cross-encoder when applying their sparsification. This strengthens our design choice for simultaneously improving both effectiveness and efficiency.

Figure 3 shows the nDCG@10 averaged on ID collections as a function of the layer ℓ before which query and document tokens can be processed independently, *i.e.*, without interaction (Mask 3). For MiniLM, we observe that it is possible to contextualize the query and document tokens independently, without harming ID effectiveness, up to the layer $\ell^* = 4$ (out of 12 total layers). There are two potential explanations for the ID effectiveness drop after $\ell \geq 5$: (1) the remaining *interaction layers* are not enough to properly capture relevance; (2) after layer $\ell^* = 4$, further contextualizing query and document tokens independently introduces signal detrimental to the IR task. We partially address this question in Section 4, when designing MICE's architecture. For Ettin-32M, the optimal ℓ^* is 6 (out of 10). We refer to ℓ^* as the *first interaction layer* for each model.

Table 1 further reports the detailed results of Mask 3 with both models, using their respective

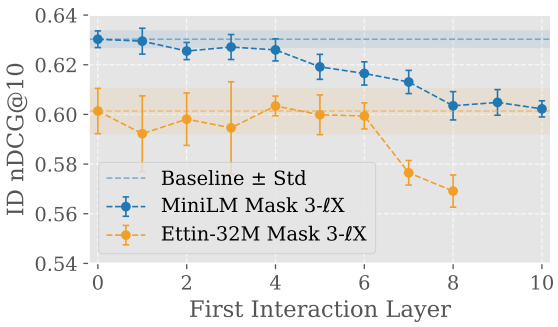

Figure 3: Search of ℓ^* (Mask 3). *All interactions between Q and D* up to a given layer are masked.

layer ℓ^*. We observe a consistent drop in OOD performance compared to using only Mask 2, in particular for Ettin-32M, with almost -4 for nDCG@10. At the same time, we observe very little variation in ID effectiveness between masking steps, and compared to the baseline (see the evolution, per backbone, in the ID column of Table 1). This suggests that our masks do not prevent the models from learning correctly the IR task, but, depending on the base model, may hinder their OOD robustness.

We also report a PreTTR reproduction on MiniLM using $\ell^* = 4$, which performs similarly to Mask 2 and surpasses Mask 3 by 1–2 points on average over ID and OOD. This gap stems from the richer interactions allowed in PreTTR's later layers. Still we expect our additional masks to yield a better efficiency-effectiveness trade-off overall.

Intermediate Conclusions Our experiments with Mask 1 first revealed that masking direct [CLS] $\not\leftarrow D$ interactions consistently improves both ID and OOD effectiveness. Then, Mask 2 confirmed that information can flow only from the document to the query ($Q \leftarrow D$), and blocking $D \not\leftarrow Q$ does not harm the model. Finally, Mask 3 corroborated previous studies on mid-fusion architectures (MacAvaney et al., 2020; Cao et al., 2020), showing that queries and documents can be contextualized independently in the lower layers, without compromising the performance.This section demonstrates that masking targeted interactions in a cross-encoder can enhance its re-ranking performance, addressing RQ 1. This gain is particularly pronounced in OOD, suggesting that our masks act as a form of regularization, preventing the model from overfitting. This observation holds across backbones and sets the stage to more efficient and as effective cross-encoders.

4 MICE - Minimal Interaction Cross Encoder

Building on the insights from Section 3, we propose a novel, streamlined late-interaction style ranker architecture we call MICE (**M**inimal **I**nteraction **C**ross-**E**ncoder). By discarding superfluous interactions, we show that MICE becomes up to 2.5× more efficient than standard cross-encoders, while maintaining competitive re-ranking performance (RQ 2).

4.1 Architecture

Compared to a standard cross-encoder, MICE differs in four principal architectural choices: **(1) Mid-Fusion** first encodes the query and document independently; **(2) Light Cross-Attention** only transfers information from a frozen document representation to the query; **(3) Layer Dropping** reduces the number of interaction layers; **(4) Lexical Head** optionally transfers lexical (exact match) information to upper interaction layers. We detail each of these choices in the following paragraphs, and provide an overview of MICE in Figure 1.

Mid-Fusion First, MICE follows the Mid-Fusion paradigm by encoding query and document tokens independently. We use the optimal fusion layer ℓ^* derived from our Mask 3 analysis (Section 3.4) to set the number of contextualization layers. MORES (Gao et al., 2020) pushes this further by dedicating two entirely separate transformer models to document and query encoding, at the cost of efficiency.

Light Cross-Attention As Mask 2 suggests that $D \not\leftarrow Q$ direction is superfluous, we go further by only computing cross-attention from *frozen document representations* to the query, eliminating expensive self-attention over document tokens. This significantly differs from PreTTR, which retains full query-document attention in its interaction layers, and is close to the interaction module of MORES (Gao et al., 2020) — though our masking analysis provides the first empirical justification. Despite sharing similarities with late-interaction architectures like ColBERT (Khattab and Zaharia, 2020), the cross-attention offers higher expressivity as MICE can capture more granular, non-linear dependencies between query and document tokens. Following MORES, we compute cross-attention before self-attention in the transformer layer, allowing query tokens to gather document information first. We also allow cross-attention transfers from document to `[CLS]` (as opposed to Mask 1), a choice validated by ablations in Section C. Cross-attention is seeded with the original self-attention weights.

Dropping Backbone Top Layers Unique to MICE, we further posit that the final layers of a backbone pretrained on MLM may be non-essential for re-ranking, as specialized for token prediction. Consequently, we experiment dropping them to limit the number of *interaction layers*.

Lexical Head Experiments with Mask 3 showed that separately contextualizing Q and D beyond ℓ^* is detrimental (Fig. 3). Overly semantic representations may hinder *lexical matching signals*. We thus also introduce a novel dedicated attention head in the interaction layers that explicitly leverages exact query-document token matches. We implement it as an additional cross-attention head that replaces the standard query-key dot product with precomputed lexical match scores (for query/document tokens (q_i, d_j), $s_{i,j} = 1$ iif $q_i = d_j$ or $s_{i,i} = 1$ if $\nexists d_j |\ q_i = d_j$ to ensure normalization). It is thus parametrized only by the value/output matrices.

4.2 Experimental Setup

Baselines In addition to comparing MICE variants to their corresponding unmasked cross-encoders (same baselines as in Section 3), we also compare MICE to previous hybrid architectures, i.e., PreTTR (MacAvaney et al., 2020) and MORES (Gao et al., 2020). As MICE resembles late-interaction like models, we further compare it with a reproduction of **ColBERT** (Khattab and Zaharia, 2020). We train such approaches from MiniLM-v2, fine-tuning them with the same setup as our MICE models (described in Appendix A.3, ensuring a fair comparison).

Validation Set Our training setup for MICE slightly differs from this of the masking experiments, as we validate on nano-BEIR (AI, 2025) instead of MS MARCO. We view nano-BEIR as a richer tool to select checkpoints, while it allows us to still use BEIR as an "OOD" benchmark, given the very limited size of its subsets. On the contrary, to avoid biasing the architecture for BEIR, we chose to validate all the design choices of MICE on ID collections only. This also includes the masking experiments and the validation on MS MARCO.

4.3 Results

For MiniLM and Ettin-32M backbones, we first analyze in Figure 4 the impact of the number of

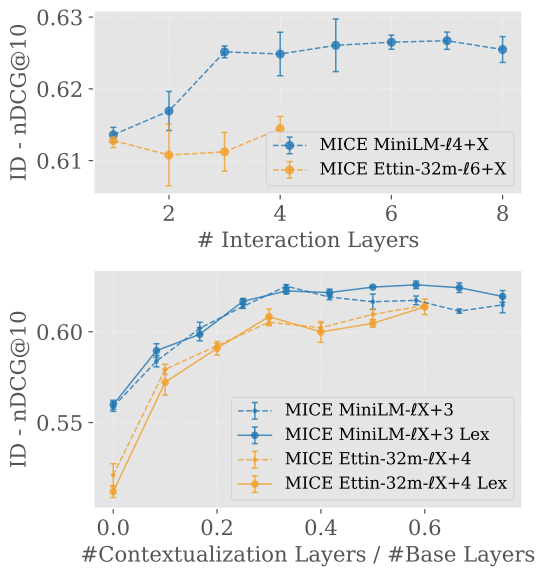

Figure 4: Impact of number of interaction / contextualization layers and lexical head ("Lex" models) in MICE (mean ID effectiveness over 3 seeds).

Table 2: MICE models evaluation of (re-ranking BM25 top 1K - nDCG@10 over 5 seeds). "MICE-ℓX+[Y/all]" models indicates using Y interaction layers (or all otherwise) starting from layer X. ↑ and ↓ marks a statistically significant difference between models and their cross-encoder baseline. **Bold** values marks the best averaged value per backbone, "_" indicates second best. "×" column is the FLOPs-based theoretical speedup compared to the baseline (see Appendix E for more details).

		In-domain (ID)			Average	
Re-Ranker	×	MSM	DL19	DL20	**ID**	**BEIR**
MiniLM-L12 - Baseline		**45.0**	**74.0**	**73.5**	**64.1**	44.0
MORES-ℓ10-d12+2	1.0	43.5↓	72.0↓	70.1↓	61.9↓	48.6↑
PreTTR-ℓ8+4	1.0	<u>43.8</u>↓	71.8↓	70.9↓	62.2↓	46.5↑
ColBERT	1.0	38.2↓	69.1↓	65.2↓	57.5↓	40.6
MICE-ℓ4+3	2.5	43.2↓	<u>73.4</u>	70.9↓	<u>62.5</u>↓	<u>49.7</u>↑
MICE-ℓ4+3 Lex	2.4	43.2↓	73.1	70.5↓	62.3 ↓	<u>49.7</u>↑
MICE-ℓ4+all	1.9	43.6↓	73.3	70.8↓	<u>62.5</u>↓	**50.5**↑
MICE-ℓ8+3	1.4	43.3↓	70.5↓	69.7↓	61.1↓	49.0↑
MICE-ℓ8+3 Lex	1.3	43.5↓	72.7↓	<u>71.2</u>↓	62.4↓	**50.5**↑
Ettin-32M - Baseline		**43.8**	<u>71.5</u>	**71.7**	**62.4**	**48.4**
MICE-ℓ3+4	2.1	42.7↓	70.5↓	68.4↓	60.5↓	<u>47.9</u>↓
MICE-ℓ3+4 Lex	2.0	42.4↓	71.1	69.0↓	60.8↓	47.6
MICE-ℓ6+4	1.3	<u>42.8</u>↓	**71.7**	69.8↓	<u>61.4</u>↓	47.2↓
MICE-ℓ6+4 Lex	1.2	42.6↓	<u>71.5</u>	69.9↓	<u>61.4</u>↓	46.8↓

contextualization and interaction layers kept from the backbone. We report detailed evaluations of MICE variants in Table 2, including a FLOPs-based theoretical speedup over the full cross-encoder (the '×' column, computed per forward pass without pre-computing documents; see Appendix E).

Layer Dropping As shown in Figure 4 (top) and Table 2, dropping late backbone layers does not hurt — and can even benefit — re-ranking effectiveness, both ID and OOD. MiniLM plateaus at 3 interaction layers (MICE-ℓ4+3 outperforms MICE-ℓ4+8 despite using only 7 of 12 layers), while Ettin-32M shows higher variance, making conclusions harder to draw. MICE-all variants show no statistically significant difference from those using fewer interaction layers.

Contextualization layers Figure 4 (bottom) shows ID effectiveness with increasing number of contextualization layers. For both backbones, effectiveness plateaus at ∼half of the backbone layers, even decreasing after $\ell^* = 4$ for MiniLM without lexical heads. It corroborates the optimal contextualization depths identified in Figure 3 ($\ell^* = 4$ for MiniLM, 6 for Ettin-32M). Table 2 provides further evidence, as MICE-MiniLM-ℓ4+3 reaches an nDCG@10 of 49.7 on BEIR (+5.7 points over its cross-encoder counterpart), while underperforming in ID. At the same time, MICE-MiniLM-ℓ4+3 achieves better ID and OOD results than both PreTTR and MORES. Ettin-32M derived MICE models show a more modest trend, slightly underperforming their baseline (47.2 for MICE-ℓ6+4 vs. 48.4).

Lexical Head Deeper layers provide richer semantic signals but at the cost of lexical information, whose absence may explain the effectiveness drop beyond optimal ℓ^* values (Figure 3). Our lexical head is designed to alleviate this trade-off, allowing deeper contextualization without sacrificing lexical matching. The "Lex" variant recovers the effectiveness drop beyond $\ell^* = 4$ for MiniLM (Figure 4), confirming that the lexical head preserves lexical information in deeper variants. No comparable gains are observed for Ettin-32M, highlighting architectural differences between BERT (Devlin et al., 2019) and ModernBERT (Warner et al., 2024). Table 2 confirms this: MICE-MiniLM-ℓ8+3 Lex outperforms its default counterpart in ID and achieves the best OOD results overall (50.5).

Baselines Comparisons Regarding effectiveness, Table 2 shows that our MICE-MiniLM variants can outperform our MiniLM-based reproduction of PreTTR and MORES in OOD. In ID, while MICE-MiniLM-ℓ8+3 slightly underperforms, the addition of the Lexical Head successfully fills the gap. However, MICE's primary advantage lies in its efficiency: MICE-ℓ4+3 is 2.5× more efficient than a full cross-encoder. In contrast, PreTTR and MORES do not reduce computational overhead (FLOPs), as inputs must still traverse all transformer layers. Comparisons with our reproduced

MiniLM-ColBERT on ID and BEIR validate that MICE enables richer interactions.

Appendices C and D detail some ablations justifying the final MICE architecture, as well as a more exhaustive Table 2 with detailed results of the BEIR datasets (Thakur et al., 2021).

In summary, we showed that, despite being up to 2.5× lighter than a standard cross-encoder, MICE preserves most of the ID performance of a standard cross-encoder while improving OOD generalization. The answer to RQ 2 is therefore positive – we successfully leveraged our masking analysis to transpose the effectiveness of a cross-encoder into a novel and more efficient architecture that eliminates superfluous interactions.

4.4 Pareto Frontier with MICE

To show the applicability of MICE beyond MiniLM and Ettin-32M, we now extend our initial study to new backbones. We use $\text{ELECTRA}_{\text{base}}$, a 110M parameter encoder (Clark et al., 2020) to extend MiniLM, and the 68M and 150M versions from the Ettin-suite (Weller et al., 2025). We build MICE models following the guidelines from the Section 4.3: setting ℓ^* to ∼half the backbone's total layers, with 3 interaction layers for BERT-based encoders and 4 for ModernBERT. We report ColBERTv2 results (Santhanam et al., 2022b) instead of our reproduction for better comparison.

Precomputing Representations A key feature of MICE's Mid-Fusion architecture is the ability to pre-compute document representations offline. At inference, MICE only needs to encode the query, gather the document representations, and run both through its small set of interaction layers. We present in Figure 5 an efficiency-effectiveness benchmark for all re-rankers considered in this work, featuring three Pareto frontiers: 1) baselines and standard cross-encoders; 2) the improvement brought by MICE; 3) a further frontier obtained by pre-computing document representations when possible (MICE, PreTTR, MORES, and ColBERTv2).

Figure 5 clearly shows efficiency gains from switching to MICE across all backbones. However, only BERT-based variants (MiniLM-v2 and ELECTRA) improve OOD effectiveness over their cross-encoder baselines and reproductions of PreTTR and MORES. On the Ettin suite, all variants except Ettin-150M show noticeable effectiveness drops, though efficiency gains keep MICE-Ettin variants on the frontier — except for Ettin-68M, as MICE-Ettin-68M (orange triangle) is overtaken by the stronger MICE-MiniLM models (blue triangles).

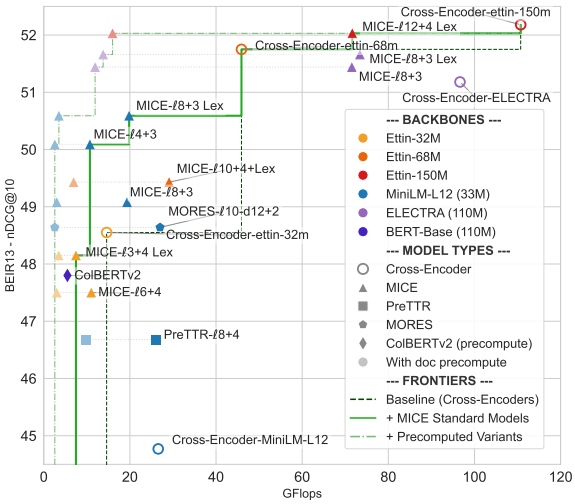

Figure 5: Pareto frontiers of all the models in our study, decomposed by backbones and types.

When pre-computing document representations, all Mid-Fusion models and ColBERT gain substantially in efficiency over traditional cross-encoders (e.g., MICE-Ettin-150M matches the efficiency of Ettin-32M while largely exceeding its OOD effectiveness). Overall, Figure 5 confirms MICE's generalization across architectures and scalability to larger models. While MICE-Ettin-68M slightly underperforms its standard counterpart, it remains more effective than Ettin-32M and can be more efficient via pre-computation, making it a valuable option for low-latency scenarios. Latency comparisons against ColBERTv2, with and without pre-computation, are further reported in Appendix F, confirming these conclusions.

5 Conclusion

In this work, motivated by insights from interpretability (Zhan et al., 2020; Lu et al., 2025) and preliminary experiments, we proposed MICE, a new architecture that starts bridging the gap between lightweight late-interaction models and heavy cross-encoders. By combining mid-fusion, cross-attention, lexical head, and layer pruning, MICE defines a new effectiveness-efficiency trade-off. Our experiments across the BERT and ModernBERT backbones confirm this. MICE reduces the computational overhead up to 2.5× compared to standard cross-encoders, matching late-interaction models like ColBERT while retaining most of cross-encoder ID effectiveness and demonstrating superior generalization abilities.

6 Limitations

In this paper, we propose a new architecture for cross-encoders named MICE. We show on different backbones how MICE can significantly improve efficiency while preserving effectiveness. However, our empirical results suggest that there is no single recipe, applicable to any backbone cross-encoder, to turn it into its MICE counterpart: the optimal configuration depends on the differences between backbone encoders such as BERT and ModernBERT. This limits the generalization of our method, and adapting MICE to a new backbone still requires some per-backbone tuning. Relatedly, our evaluation is restricted to English retrieval (MS MARCO and BEIR) and to two encoder families (BERT and ModernBERT) over a limited range of model sizes; we therefore do not provide evidence that our findings transfer to other languages or to substantially different architectures.

Another limitation concerns our evaluation of the OOD abilities of MICE. As we rely on nano-BEIR for validation when training MICE, our results on BEIR may be biased by the small amount of data used to pick the checkpoints across our random seeds (each nano-BEIR dataset is a subset of 5000 documents and 50 queries of the original dataset). This affects our reported OOD numbers, and we cannot rule out that a different validation set would lead to different selected checkpoints. Two factors limit the impact: all of our design choices about MICE, including the masking experiments, were motivated using ID-only evaluations, and the same checkpoint-selection methodology is applied to our baselines and reproductions, so models are compared on equal footing. A more thorough OOD evaluation, e.g. adding the LoTTE benchmark (Santhanam et al., 2022b), would have increased the computational cost significantly given the scale of our study (3 random seeds per model $\times$ 26 backbones[2] $\times$ 5 more datasets), and we leave it to future work.

In Figure 5, we assumed document representations could be pre-computed and stored uncompressed. This is unrealistic at scale, and we do not measure the storage cost or its impact on latency; optimizing multi-vector storage is a separate line of work, out of the scope of this submission, which aims at proposing a new alternative to cross-encoders that allows pre-computing document representations. MICE would likely benefit from such optimization, but our reported efficiency gains should be read under this no-compression assumption.

Finally, MICE remains a re-ranker: our attempts to push it toward first-stage retrieval were unsuccessful. Removing the self-attention over query tokens in the interaction layers collapsed ranking performance, indicating that intra-query contextualization is critical for effective re-ranking. Reducing the dimensionality of the interaction layers proved less effective than layer pruning, as the compressed layers cannot be initialized from the backbone's pre-trained weights. Combining MICE with PLAID (Santhanam et al., 2022a) to compress its pre-computed document embeddings led to a large drop in performance.

[2]Counting only the ones reported in the Table 6.

References

Sionic AI. 2025. Nano-beir: A multilingual information retrieval benchmark with quality-enhanced queries.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, and 1 others. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *ArXiv*, abs/2004.05150.

Cesare Campagnano, Antonio Mallia, Jack Pertschuk, and Fabrizio Silvestri. 2025. E2rank: Efficient and effective layer-wise reranking. In *European Conference on Information Retrieval*, pages 417–426. Springer.

Daniel Campos, Alexandre Marques, Tuan Nguyen, Mark Kurtz, and ChengXiang Zhai. 2023. Sparse*bert: Sparse models generalize to new tasks and domains. *Preprint*, arXiv:2205.12452.

Qingqing Cao, Harsh Trivedi, Aruna Balasubramanian, and Niranjan Balasubramanian. 2020. DeFormer: Decomposing Pre-trained Transformers for Faster Question Answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4497, Online. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the trec 2020 deep learning track. In *Text REtrieval Conference (TREC)*. TREC.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the trec 2019 deep learning track. In *Text REtrieval Conference (TREC)*. TREC.

Hervé Déjean, Stéphane Clinchant, and Thibault Formal. 2024. A thorough comparison of cross-encoders and llms for reranking splade. *ArXiv*, abs/2403.10407.

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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Javier Ferrando, Gabriele Sarti, Arianna Bisazza, and Marta R. Costa-jussà. 2024. A primer on the inner workings of transformer-based language models. *Preprint*, arXiv:2405.00208.

Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. Splade: Sparse lexical and expansion model for first stage ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2288–2292. ACM.

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Luyu Gao, Zhuyun Dai, and Jamie Callan. 2020. Modularized transfomer-based ranking framework. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4180–4190, Online. Association for Computational Linguistics.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *Preprint*, arXiv:1503.02531.

Sebastian Hofstätter, Sophia Althammer, M. Schröder, Mete Sertkan, and A. Hanbury. 2020. Improving Efficient Neural Ranking Models with Cross-Architecture Knowledge Distillation. *ArXiv*.

Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2020. Poly-encoders: Architectures and pre-training strategies for fast and accurate multi-sentence scoring. In *International Conference on Learning Representations*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *Preprint*, arXiv:2001.08361.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 39–48, New York, NY, USA. Association for Computing Machinery.

Yibin Lei, Shwai He, Ang Li, and Andrew Yates. 2025. Making large language models efficient dense retrievers. *Preprint*, arXiv:2512.20612.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A STRUCTURED SELF-ATTENTIVE SENTENCE EMBEDDING. In *International Conference on Learning Representations*.

Meng Lu, Catherine Chen, and Carsten Eickhoff. 2025. Pathway to relevance: How cross-encoders implement a semantic variant of BM25. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 25525–25547, Suzhou, China. Association for Computational Linguistics.

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2024. Fine-tuning llama for multi-stage text retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 2421–2425, New York, NY, USA. Association for Computing Machinery.

Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020. Efficient Document Re-Ranking for Transformers by Precomputing Term Representations. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, pages 49–58, New York, NY, USA. Association for Computing Machinery.

Chuan Meng, Negar Arabzadeh, Arian Askari, Mohammad Aliannejadi, and Maarten de Rijke. 2024. Ranked list truncation for large language model-based re-ranking. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 141–151, New York, NY, USA. Association for Computing Machinery.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3. In *Proceedings of The Third Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994*, NIST Special Publication, pages 109–126. National Institute of Standards and Technology (NIST).

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Guilherme Rosa, Luiz Bonifacio, Vitor Jeronymo, Hugo Abonizio, Marzieh Fadaee, Roberto Lotufo, and Rodrigo Nogueira. 2022. In defense of cross-encoders for zero-shot retrieval. *Preprint*, arXiv:2212.06121.

Keshav Santhanam, Omar Khattab, Christopher Potts, and Matei Zaharia. 2022a. Plaid: an efficient engine for late interaction retrieval. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 1747–1756.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022b. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3715–3734, Seattle, United States. Association for Computational Linguistics.

Ferdinand Schlatt, Maik Fröbe, and Matthias Hagen. 2024. Investigating the Effects of Sparse Attention on Cross-Encoders. volume 14608, pages 173–190.

Ferdinand Schlatt, Maik Fröbe, Harrisen Scells, Shengyao Zhuang, Bevan Koopman, Guido Zuccon, Benno Stein, Martin Potthast, and Matthias Hagen. 2025. *Rank-DistiLLM: Closing the Effectiveness Gap Between Cross-Encoders and LLMs for Passage Re-ranking*, page 323–334. Springer Nature Switzerland.

Hao Tan and Mohit Bansal. 2019. LXMERT: Learning cross-modality encoder representations from transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5100–5111, Hong Kong, China. Association for Computational Linguistics.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoeit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Poloshukin. 2017. Attention is all you need. In *31st Conference on Neural Information Processing Systems (NIPS 2017)*.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. 2021. MiniLMv2: Multi-head self-attention relation distillation for compressing pretrained transformers. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2140–2151, Online. Association for Computational Linguistics.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Nathan Cooper, Griffin Adams, Jeremy Howard, and Iacopo Poli. 2024. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. *Preprint*, arXiv:2412.13663.

Orion Weller, Kathryn Ricci, Marc Marone, Antoine Chaffin, Dawn Lawrie, and Benjamin Van Durme. 2025. Seq vs seq: An open suite of paired encoders and decoders.

Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2021. Fastformer: Additive attention can be all you need. *ArXiv*, abs/2108.09084.

Zhichao Xu, Zhiqi Huang, Shengyao Zhuang, and Vivek Srikumar. 2025. Distillation versus contrastive learning: How to train your rerankers. *Preprint*, arXiv:2507.08336.

Andrew Yates, Rodrigo Nogueira, and Jimmy Lin. 2021. Pretrained transformers for text ranking: Bert and beyond. In *Proceedings of the 14th ACM International Conference on web search and data mining*, pages 1154–1156.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: transformers for longer sequences. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2020. An analysis of bert in document ranking. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 1941–1944, New York, NY, USA. Association for Computing Machinery.

A Additional Details

A.1 Reproducibility Statement

To facilitate rigorous community evaluation and ensure full reproducibility, we release our complete modular codebase alongside all configuration

files required to replicate our training and evaluation pipelines. Furthermore, all MICE model checkpoints and TensorBoard logs will be publicly available on HuggingFace upon publication.

A.2 Hardware Configuration.

All training experiments are conducted using a single NVIDIA H100 (80GB) GPU. Training is highly efficient with MICE's streamlined architecture: a typical run for a MiniLM-based MICE variant requires approximately three hours, depending on the specific architectural parameters. For inference and large-scale evaluation benchmarks, we utilize NVIDIA V100 (32GB) GPUs. All input tokenization, sequence lengths and evaluation methodology remain constant across these environments to ensure parity.

A.3 Experimental Setup

Training hyperparameters (Masking experiment, Section 3). To fine-tune our models and baselines, we rely on distillation with the Margin-MSE loss (Hofstätter et al., 2020). This loss yields superior retrieval performance compared to the standard Binary Cross Entropy (BCE) (Xu et al., 2025), by preserving the magnitude of the relevance difference (margin) between positive and negative pairs, rather than treating them as binary labels. As a teacher, we use the set of rerankers scores on the MS MARCO passage ranking dataset (Bajaj et al., 2016) from Hofstätter et al., already used to train efficient re-ranking architectures such as ColBERT (Khattab and Zaharia, 2020).

We use the same training hyperparameters in all experiments and train all models for 125,000 steps using a batch size of 32, a learning rate of 7×10^{-6}, and 5,000 warmup steps. We validate every 10,000 steps based on RR@10 performance on the MS MARCO development set and keep the best checkpoint.

Training hyperparameters (MICE, Section 4). The set of hyperparameters used to train our MICE models is the same as the one to learn our masks in Section 3. The only difference, is in the validation data. Unlike for learning the masks, we follow the `sentence-transformers` library guidelines [3] and use nano-BEIR (AI, 2025) as a proxy for generalization effectiveness. These subsets of the 13 public datasets of BEIR, allow for frequent validation without the overhead of using the full datasets.

[3] https://sbert.net/index.html

A.4 Models

Configurations of all the base models used in this work are detailed in Table 3.

Table 3: Configurations of the models used in this work.

Model	**MiniLM-L12-v2 (MiniLM)**
Pretrained checkpoint 🤗	`microsoft/MiniLM-L12-H384-uncased`
Architecture	**Layers**=12, **Params**=33M, **Hidden**=384, **Heads**=12
Model	**Ettin-32M**
Pretrained checkpoint 🤗	`jhu-clsp/ettin-encoder-32m`
Architecture	**Layers**=10, **Params**=32M, **Hidden**=384, **Heads**=6
Model	**Ettin-68**
Pretrained checkpoint 🤗	`jhu-clsp/ettin-encoder-68m`
Architecture	**Layers**=19, **Params**=68M, **Hidden**=512, **Heads**=8
Model	**BERT**
Pretrained checkpoint 🤗	None (backbone of ColBERTv2)
Architecture	**Layers**=12, **Params**=110M, **Hidden**=768, **Heads**=12
Model	**ELECTRA**
Pretrained checkpoint 🤗	`google/electra-base-discriminator`
Architecture	**Layers**=12, **Params**=110M, **Hidden**=768, **Heads**=12
Model	**Ettin-150**
Pretrained checkpoint 🤗	`jhu-clsp/ettin-encoder-150m`
Architecture	**Layers**=22, **Params**=150M, **Hidden**=768, **Heads**=12

A.5 Datasets

Table 4: List of the 13 publicly available datasets in the BEIR benchmark (Thakur et al., 2021), along with the corresponding abbreviations used in the article and their domain.

Dataset	Task	Abb.
ArguAna	Arg. Retrieval	Ar
Climate-FEVER	Fact Checking	CF
DBPedia	Entity Retrieval	DB
FEVER	Fact Checking	FE
FiQA-2018	QA	Fi
HotpotQA	QA	HPQ
NFCorpus	Bio-Medical IR	NFC
Natural Questions	QA	NQ
Quora	Dup. Questions	Q
SCIDOCS	Citation Prediction	SD
SciFact	Fact Checking	SF
Touché-2020	Arg. Retrieval	T-v2
TREC-COVID	Bio-Medical IR	T-C

Evaluation. We evaluate our models both *in-domain (ID)*, using MS MARCO (Bajaj et al., 2016) dev set (MSM) and the high-quality annotations of the TREC Deep Learning tracks from 2019 (DL19) and 2020 (DL20) (Craswell et al., 2020, 2021); and *out-of-domain (OOD)*, where we rely on the publicly available subset of datasets of the BEIR benchmark (Thakur et al., 2021) (listed in Table 4). This includes the following list of 13 datasets: ArguAna (Ar), Climate-FEVER (CF), DBPedia (DB), FEVER (FE), FiQA

(Fi), HotpotQA (HPQ), NFCorpus (NFC), Natural Questions (NQ), Quora Question Pairs (Q), SCIDOCS (SD), SciFact (SF), Touché-2020 (T-v2), and TREC-COVID (T-C).

B Masking Experiments

B.1 Additional Details on the Masking Steps

To further motivate the masking steps described in Section 3, we provide additional details behind their design.

Masking Step 0

Across all layers, `[SEP]` tokens are prevented from receiving information from any input part ($\texttt{[SEP]} \not\leftarrow \{\texttt{[CLS]}, Q, D\}$), and `[CLS]` is prevented from sending information to other parts ($\{Q, \texttt{[SEP]}, D\} \not\leftarrow \texttt{[CLS]}$). We further enforce that `[SEP1]` and `[SEP2]` act as dedicated attention sinks respectively for Q and D — allowing $Q \leftarrow$`[SEP1]` and $D \leftarrow$`[SEP2]` while blocking $Q \not\leftarrow$`[SEP2]` and $D \not\leftarrow$`[SEP1]` — as attention sinks are crucial for absorbing undesirable interactions (Zhan et al., 2020).

Masking Step 1

The motivation for blocking $\texttt{[CLS]} \not\leftarrow \{D, \texttt{[SEP2]}\}$ stems from empirical evidence that query tokens act as the primary recipients of query-document interactions, accumulating relevance signals that are subsequently routed to `[CLS]` (Lu et al., 2025). Blocking the direct document-to-`[CLS]` path thus removes a redundant and potentially noisy channel. Subsequent ablations (see Appendix C), show that blocking this interaction is found to be detrimental to MICE.

Masking Step 2

Prior work has shown that query-to-document interactions ($D \not\leftarrow Q$) contribute less to the overall ranking process than document-to-query ones (Zhan et al., 2020), motivating their removal across all layers. Beyond efficiency, this step paves the way towards architectures where document representations can be computed offline (MacAvaney et al., 2020). Note that this step directly contradicts (Schlatt et al., 2024) decision to mask $Q \not\leftarrow D$ in its Sparse cross-encoder design.

Masking Step 3

The rationale for blocking $Q \not\leftarrow D$ in the early layers is that the model primarily contextualizes query and document tokens independently in this stage, with cross-interactions being secondary (Zhan et al., 2020; Lu et al., 2025). We define ℓ^* as the highest layer up to which this mask can be applied without degrading the base model's effectiveness, and determine it empirically.

B.2 Impact of masking superfluous interactions

To assess how masking superfluous interactions between input parts within the self-attention modules affects cross-encoder effectiveness, we report in Table 5 the results obtained by applying our masks to two off-the-shelf cross-encoder models, based on our two backbones: `cross-encoder/ms-marco-MiniLM-L12-v2` for MiniLM-v2, and `tomaarsen/ms-marco-ettin-32m-reranker` for Ettin-32M.

From Table 5, we observe that the MiniLM-based cross-encoder and the Ettin-based cross-encoder respond very differently to the different masking strategies. Although the base effectiveness of the MiniLM model remains stable for masks 0 and 1 (around 1 nDCG@10 point in average on both ID and OOD), these masks impact severely the cross-encoder based on Ettin (drop of more than 30 nDCG@10 on MS MARCO for Mask 0). A possible explanation is that our masking steps are derived from Lu et al. (2025), who study the MiniLM-v2 cross-encoder model only; they may not fit the internal mechanisms of Ettin-based cross-encoders. For instance on Ettin, while Mask 0 and Mask 1 are intended to target interactions that should have only a marginal effect on model performance (by limiting information flow toward attention sinks, as observed with MiniLM), it is plausible that, because they are based on ModernBERT and pre-trained with mechanisms such as sliding-window attention (unlike BERT-base models), their attention sinks are different from those of BERT-based models. As a result, and given the effect of our masking strategy that redistributes the attention probability that was concentrated on the sink, our masks may be less appropriate for Ettin than for MiniLM. We also observe that Mask 1 increases performance over Mask 0 for Ettin, but it remains much lower than with the unmasked model. Finally, while Mask 0 and Mask 1 only slightly impact MiniLM cross-encoder, our results indicate that further masking (Mask 2) leads to a more substantial decrease of its effectiveness (-10 on nDCG@10 for both ID and OOD).

Table 5: Masking off-the-shelf cross-encoders with our approach (Section 3). Reranking is performed over 1000 documents retrieved by BM25.

	Re-Ranker	In-domain MSM	In-domain DL19	In-domain DL20	Average ID	Average BEIR
MiniLM	Cross-Encoder	45.7	75.5	73.6	64.9	49.5
	+ Mask 0	44.4	74.3	72.0	63.5	48.2
	+ Mask 1	44.4	73.4	72.0	63.2	48.0
	+ Mask 2	27.8	61.5	59.8	49.7	36.6
Ettin-32M	Cross-Encoder	43.7	70.8	71.3	61.9	48.1
	+ Mask 0	12.0	29.0	29.3	23.4	7.5
	+ Mask 1	27.0	56.4	52.9	45.4	26.8
	+ Mask 2	21.3	47.8	47.8	39.0	24.0

Together, these results indicate that it is possible to remove some interactions inside the self-attention modules of a cross-encoder, without impacting its effectiveness (see Mask 0 and Mask 1 for MiniLM). However, doing so requires a good understanding of the model internal mechanisms as acknowledged by the results with Ettin. These insights, in addition to the substantial drop induced by Mask 2, suggest that it is possible to maintain a cross-encoder performance while removing unnecessary interactions only up to a certain point. Removing these interactions on an already fine-tuned model seems to harm its effectiveness, showing that the fine-tuned models still have learned to use some of the information transfers we mask to predict relevance. Consequently, empirical evidence from Sparse CE (Schlatt et al., 2024) shows that fine-tuning can preserve the base model's effectiveness even when a key information transfer is removed. This naturally motivates fine-tuning the re-ranker with our masks applied, as studied in the main body of the article (see Section 3).

C Ablations on MICE Architectue

To validate our architectural choices, we report different ablations of the MICE architecture presented in Section 4.1. To limit computational overhead, we limit to evaluations on the ID datasets listed in Section A.5.

From Figure 6 we conclude that:

- Unlike MORES (Gao et al., 2020), MICE does not draw any benefit from having a separate encoder for the query and the document.
- Contrary to our results with Mask 1 in Section 3, it seems that MICE profits from `[CLS]` $\leftarrow$ D transfers. While counter-intuitive, we attribute this difference to the fact that in MICE, we also prevent document contextualization in the interaction layers. This change itself might explain that the model needs an additional degree of liberty to predict relevance.
- Traditionally, a cross-attention layer in decoders is composed of a self-attention, followed by a cross-attention. Yet, in their design, MORES (Gao et al., 2020) exchanged these two modules, which prompted us to compare these two variations. Our conclusion confirms their decision.

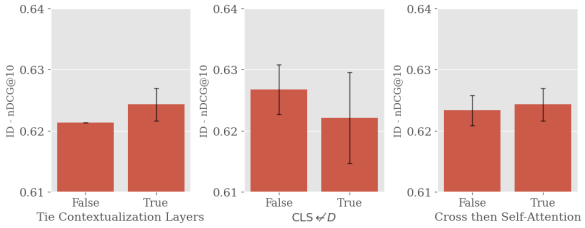

Figure 6: Ablations of three design choices in MICE: 1) whether or not to keep the weights of the contextualization layers tied; whether or not to allow transfer of information from the document to the `[CLS]` (see Mask 1); whether or not to do the cross-attention first or the self-attention first. Impact is measured on the 3 ID datasets (MSM, TREC-DL19 and TREC-DL20).

D MICE Full Results

To complement the results from Table 2, we detail in Table 6 all the results for all datasets on which MICE models were evaluated. We also include evaluations for MICE models trained from other backbones than MiniLM and Ettin-32M, showing the generalization of our proposed architecture, and further reports the results of ColBERTv2 (Santhanam et al., 2022b). Interestingly, if we compare ColBERTv2 to its closest `MICE-ELECTRA-`ℓ`8+3 Lex`, its closest variant as ColBERTv2 is based on $\text{BERT}_{\text{base}}$, we can see that MICE matches its ID results, while outperforming it with a large margin on BEIR. This underlines that our design brings additional capacity to the model and allows MICE to compare favorably with state-of-the-art late-interaction models, both in terms of effectiveness and efficiency. Furthermore, we note that Table 6 results confirm a trend already observed from Table 2: when applied to BERT-based cross-encoders, MICE is able to improve the effectiveness in OOD and the efficiency, while the effectiveness tends to drop slightly with Ettin-based models. Results from ELECTRA (Clark et al., 2020) and the other models in the Ettin-suite (Weller et al., 2025) this

Table 6: Full Re-ranking evaluation results over 1k docs/query from BM25 in nDCG@10 (over 5 seeds) for the experiments on MICE. "MICE-ℓX+[Y/all]" models indicates using only Y interaction layers (or all otherwise) starting from layer X. X* marks a statistically significant difference between models and their cross-encoder baseline. **Bold** values marks the best averaged value per backbone, "_" indicates second best. We also report the theoretical speedup ($\times$) comparing the FLOPs used to compute the relevance score for one Query/document pair.

		In-domain (ID)			BEIR 13													Average	
Re-Ranker	$\times$	**MSM**	**DL19**	**DL20**	**Ar**	**CF**	**DB**	**FE**	**Fi**	**HPQ**	**NFC**	**NQ**	**Q**	**SD**	**SF**	**T-v2**	**T-C**	**ID**	**BEIR**
BM25		23.0	51.2	47.7	30.0	16.5	31.8	65.1	23.6	63.3	32.2	30.6	78.9	14.0	67.9	45.4	59.5	42.5	43.0
ColBERTv2		45.0	74.6	73.4	33.8	17.0	44.3	73.0	34.2	66.2	33.4	53.9	86.3	14.1	63.8	34.2	66.7	64.3	47.8
MiniLM - Baseline	1	**45.0**	**74.0**	**73.5**	16.5	11.9	**46.2**	74.7	32.5	**72.9**	24.8	**56.6**	81.3	12.1	49.1	24.5	68.6	**64.1**	44.0
PreTTR-ℓ8+4	1.0	43.8↓	71.8↓	70.9↓	2.6↓	21.8↑	43.3↓	80.8	36.0↑	68.8↓	33.8↑	53.9↓	78.4	16.0↑	68.8↑	27.4↑	72.8↑	62.2↓	46.5↑
MORES-ℓ10-d12+2	1.0	43.5↓	72.0↓	70.1↓	29.1↑	25.8↑	41.8↓	78.8	36.3↑	68.7↓	34.0↑	52.6↓	78.6↓	15.8↑	69.7↑	29.4↑	71.6↑	61.9↓	48.6↑
ColBERT	1.0	38.2↓	69.1↓	65.2↓	29.5↑	15.0	35.0↓	69.0↓	26.2↓	52.5↓	31.0↑	46.3↓	59.3↓	12.3	58.6↑	25.0	72.0↑	57.5↓	40.6
MICE-ℓ4+3	2.5	43.2↓	73.4	70.9↓	**35.6**↑	24.6↑	44.1↓	80.0	34.5	71.0↓	34.3↑	51.9↓	**81.5**	16.0↑	69.2↑	**31.7**↑	72.0↑	62.5↓	49.7↑
MICE-ℓ8+3	1.4	43.3↓	70.5↓	69.7↓	33.4↑	24.9↑	41.8↓	80.1	36.3↑	68.7↓	33.7↑	52.5↓	78.6↓	15.7↑	68.9↑	29.6↑	72.9↑	61.1↓	49.0↑
MICE-ℓ8+3 Lex	1.3	43.5↓	72.7↓	71.2↓	35.3↑	26.8↑	42.7↓	82.0↑	**36.8**↑	71.3↓	**34.5**↑	53.4↓	80.9	**16.1**↑	70.3↑	31.4↑	**75.1**↑	62.4↓	**50.5**↑
MICE-ℓ8+all Lex	1.3	43.8↓	71.9↓	70.6↓	34.9↑	**27.0**↑	43.2↓	**82.3**↑	36.5↑	71.7↓	34.3↑	53.5↓	80.4	**16.1**↑	**70.9**↑	31.5↑	74.6↑	62.1↓	**50.5**↑
ELECTRA - Baseline	1	**46.1**	**74.7**	**74.5**	21.9	26.5	**47.5**	**84.1**	**39.9**	**74.7**	**35.6**	**58.9**	82.2	**17.3**	71.9	27.6	**75.1**	**65.1**	51.0
MICE-ℓ8+3	1.4	44.8↓	73.5	72.7↓	40.2↑	**27.9**↑	45.1↓	80.5↓	38.6↓	71.6↓	34.7↓	55.3↓	**83.1**	16.5↓	**72.0**	**29.1**	74.9	63.7↓	51.5↑
MICE-ℓ8+3 Lex	1.3	44.8↓	73.2↓	72.7↓	**40.8**↑	27.3↑	45.6↓	81.2↓	38.5	73.0↓	34.8↓	55.7↓	**83.1**↑	16.7↓	71.4↓	29.0	73.6↓	63.6↓	**51.6**↑
Ettin32 - Baseline	1	**43.8**	71.5	**71.7**	10.2	**25.0**	**42.0**	**83.4**	**36.6**	**71.1**	33.1	**53.0**	81.3	**16.0**	69.9	**31.0**	**76.9**	**62.4**	**48.4**
MICE-ℓ3+4	2.1	42.7↓	70.5↓	68.4↓	31.4↑	22.9↓	40.9	73.1↓	33.9↓	69.4↓	32.7	50.1↓	81.1	15.5	70.7	25.7	74.9↓	60.5↓	47.9↓
MICE-ℓ3+4 Lex	2.0	42.4↓	71.1	69.0↓	**31.9**↑	22.5↓	40.4↓	72.1↓	33.8↓	69.0↓	32.5	49.8↓	81.9	15.1↓	70.3	26.6	73.3↓	60.8↓	47.6
MICE-ℓ6+4	1.3	42.8↓	**71.7**	69.8↓	25.0↑	23.1↓	40.2↓	73.5↓	33.4↓	68.5↓	**33.2**	50.9↓	82.5	15.3↓	**70.8**	25.4↓	72.4↓	61.4↓	47.2↓
MICE-ℓ6+4 Lex	1.2	42.6↓	71.5	69.9↓	22.1	22.4↓	39.8↓	73.2↓	33.3↓	69.0↓	**33.2**	50.8↓	**82.9**↑	15.0↓	69.4	24.2↓	73.1↓	61.4↓	46.8↓
Ettin68 - Baseline	1	**45.6**	**74.4**	**73.7**	15.8	**27.7**	**47.0**	**86.1**	**40.3**	**74.0**	**35.0**	**57.4**	81.4	**17.2**	**72.5**	**31.3**	**81.8**	**64.6**	**51.3**
MICE-ℓ10+4+Lex	1.6	43.8↓	71.4↓	72.2↓	**30.5**↑	21.6↓	42.6↓	75.8↓	35.8↓	71.2↓	33.5↓	53.5↓	83.4	15.3↓	70.0	27.2↓	75.6↓	62.5↓	48.9↓
MICE-ℓ8+3	2.1	44.7↓	72.0↓	72.5↓	28.0↑	24.3↓	44.5↓	76.9↓	36.2↓	71.0↓	34.6↓	54.0↓	**83.7**	16.2↓	70.8	26.7↓	77.3↓	63.1↓	49.6↓
MICE-ℓ8+3+Lex	2.0	44.5↓	72.3↓	72.2↓	29.9↑	23.8↓	44.1↓	77.9↓	36.0↓	71.3↓	34.4↓	53.8↓	83.4	16.2↓	71.3	28.0	77.3↓	63.0↓	49.8↓
Ettin150 - Baseline	1	**46.2**	**74.8**	**74.0**	21.9	**28.3**	**47.7**	**85.0**	**40.9**	**75.4**	**35.9**	**59.0**	77.4	**18.3**	**74.9**	**29.2**	**82.2**	**65.0**	**52.0**
MICE-ℓ12+3	1.7	45.5↓	74.6	73.7	**39.8**↑	23.9↓	46.4↓	78.7↓	37.7↓	73.4↓	35.2↓	56.6↓	83.9	16.8↓	74.2	27.1↓	79.9↓	64.6	51.8
MICE-ℓ12+4	1.5	45.6↓	74.3	73.7	35.3↑	23.6↓	46.4↓	78.8↓	37.5↓	74.1↓	35.2↓	56.8↓	**84.7**	17.0↓	73.6	27.4	80.5	64.5↓	51.6

observation. We note that, despite contrasted results in terms of effectiveness, MICE variants for Ettin-{32M-150M} all profit from massive FLOPs speedups, around twice their respective baseline. This effect can also be observed in the Figure 5.

E FLOPs Analysis

E.1 Computing FLOPs in a transformer model

To quantify the computational efficiency of each masking steps and MICE, we estimate the Floating Point Operations (FLOPs) required for a forward pass. By introducing an interaction mask $M \in \{0,1\}^{S\times S}$, we *theoretically* reduce this cost proportional to the masking density $\alpha_\ell = \frac{\|M\|_0}{S^2}$. Where $\alpha_\ell = 1$ for a standard cross-encoder with full attention. Let L denote the number of layers, d the model dimension, and d_{ff} the MLP intermediate dimension. For a sequence of length $S = n + m + 3$ (query length n, document length m), the FLOPs for a transformer layer with attention density α_ℓ is:

The FLOPs estimate for a single masked transformer layer with attention density α_ℓ is expressed as:

$$C_\ell \approx 2 \times S\,(\underbrace{4d^2}_{\text{Proj}} + \underbrace{2dd_{ff}}_{\text{MLP}}) + \underbrace{4\alpha_\ell d_h\, S^2}_{\text{Attention}} \quad (1)$$

where the factor of two comes from the multiply-accumulate operation used in matrix multiplication.

Table 7: FLOPs estimates for a single transformer layer. FLOPs are calculated for a full sequence of length S. The interaction step incorporates the masking density parameter $\alpha_\ell \in [0,1]$, where $\alpha_\ell = 1$ recovers standard full attention. Sub-leading terms such as nonlinearities, biases, and layer normalization are omitted, following Kaplan et al. (2020).

Operation	**FLOPs (S tokens)**
Attention: QKV	$6Sd^2$
Attention: Project	$2Sd^2$
Masked Attention	$4\alpha_\ell S^2 d$
Feedforward	$4Sd \cdot d_{ff}$
Total	$2 \times S(4d^2 + 2dd_{ff}) + 4S^2\alpha_\ell d$

Using this method, our architectural pruning lead to several savings:

(1) Block-Diagonal Attention: For the first ℓ^* layers, query and document are contextualized independently. The attention mask density is reduced

to $\alpha_{\text{early}} = \frac{n^2+m^2}{S^2}$, transforming the quadratic bottleneck from $O(S^2)$ to $O(n^2 + m^2)$ operations.

(2) Frozen Document in Interaction Layers: In the L_{int} interaction layers, only query representations are updated. Document tokens are frozen, bypassing their MLP and self-attention computations. The cost per interaction layer is:

$$C_{\text{int}} \approx \underbrace{8nd^2 + 4nd \cdot d_{ff}}_{\text{Query Projections \& MLP}} + \underbrace{4n(n+m)d}_{\text{Cross-Attention}} \quad (2)$$

This replaces $O(S^2)$ complexity with $O(n \cdot S)$. Since $n \ll m$, this provides a substantial speedup.

(3) Layer Pruning: MICE discards L_{drop} layers. The total FLOPs for MICE is:

$$C_{\text{MICE}} = \ell^* \cdot C_\ell(\alpha_{\text{early}}) + L_{\text{int}} \cdot C_{\text{int}} \quad (3)$$

where $\ell^* + L_{\text{int}} = L - L_{\text{drop}}$. For the MiniLM MICE-$\ell$4+3 model, this reduces the total active layers from 12 to 7.

E.2 Approach

In Table 2, we report a theoretical FLOPs speedup: in practice, we compute it as FLOPs(baseline) / FLOPs(model) to get the approximate speedup for computing one relevance score. We use $S = 512$, $n = 32$, and $m = S - n - 3$ for all models. This method yields a 2.5 $\times$ speedup for the MiniLM-based MICE-ℓ4+3 variant shown in Table 2.

F Efficiency Analysis

To complement the evaluation of the re-ranking effectiveness of MICE and of its efficiency in terms of FLOPs, we compare its latency and memory footprint with the standard cross-encoder architecture (entire forward pass over a query-document pair) and ColBERT (separately encoding the query and document before computing a `MaxSim` over their compressed representations). We present efficiency measures when with and without pre-computing the document representations, as ColBERT and MICE offer this possibility, contrary to a cross-encoder. In practice, we use the same `MiniLM-L12-v2` backbone for each architecture (cross-encoder, ColBERT and MICE) to ensure a fair comparison. We measure the inference time averaged over 100 forward passes on a maximum load setup (512 doc. token length, batch size 128 using an 12Gb Nvidia TITAN-V GPU) and report results in Table 8.

MICE achieves a $2\times$ speedup over standard cross-encoders, rising to $4\times$ with pre-computed document representations—effectively matching ColBERT's latency ($1.15\times$) while delivering superior performance (See Section 4.3). This is achieved by using fewer layers (i.e., the last layers of the backbone are dropped) and reducing interaction to cross-attention. Compared to ColBERT, we, however, have a larger memory footprint ($1.8\times$).

Table 8: Efficiency comparison of retrieval models. All based on a `MiniLM-L12-v2` backbone for comparison.

	Model	Precomp.	#param	Latency (ms)	Docs/s	Peak Mem
MiniLM	MICE ℓ4+3	✅	26.3M	113.28 ± 12.05	1130	598.44 MB
	ColBERT	✅	33.4M	130.36 ± 7.86	982	331.77 MB
	Cross-Encoder	❌	33.4M	470.22 ± 4.87	267	1193.52 MB
	MICE ℓ4+3	❌	26.3M	241.05 ± 6.25	531	1071.61 MB
	ColBERT	❌	33.4M	498.48 ± 8.65	257	1195.27 MB