Simple Projection Variants Improve ColBERT Performance

Benjamin Clavié[1,2], Sean Lee[1], Rikiya Takehi[1,3], Aamir Shakir[1] and Makoto P. Kato[2,4]

[1]*Mixedbread AI*
[2]*National Institute of Informatics (NII)*
[3]*Waseda University*
[4]*University of Tsukuba*

Abstract

Multi-vector dense retrieval methods like ColBERT systematically use a single-layer linear projection to reduce the dimensionality of individual vectors. In this study, we explore the implications of the MaxSim operator on the gradient flows of the training of multi-vector models and show that such a simple linear projection has inherent, if non-critical, limitations in this setting. We then discuss the theoretical improvements that could result from replacing this single-layer projection with well-studied alternative feedforward linear networks (FFN), such as deeper, non-linear FFN blocks, GLU blocks, and skip-connections, could alleviate these limitations. Through the design and systematic evaluation of alternate projection blocks, we show that better-designed final projections positively impact the downstream performance of ColBERT models. We highlight that many projection variants outperform the original linear projections, with the best-performing variants increasing average performance on a range of retrieval benchmarks across domains by over 2 NDCG@10 points. We then conduct further exploration on the individual parameters of these projections block in order to understand what drives this empirical performance, highlighting the particular importance of upscaled intermediate projections and residual connections. As part of these ablation studies, we show that numerous suboptimal projection variants still outperform the traditional single-layer projection across multiple benchmarks, confirming our hypothesis. Finally, we observe that this effect is consistent across random seeds, further confirming that replacing the linear layer of ColBERT models is a robust, drop-in upgrade.

Keywords

Multi-vector Retrieval, ColBERT, Model Architecture, Neural Information Retrieval

1. Introduction

During the past several years, a rapidly growing subfield of Information Retrieval (IR) has been Neural IR, largely consisting of deep learning methods built on the Transformer architecture [1] and leveraging the pretrained weights of language models such as BERT [2]. Within Neural IR, many individual paradigms have appeared. Among others, single-vector dense retrieval [3], learned sparse retrieval [4, 5] and late-interaction multi-vector retrieval, frequently referred to as ColBERT [6] after the model introducing this paradigm, have been particularly notable.

Multi-vector models, e.g. ColBERT and its variants, work by encoding both queries and documents into many small token-level vectors, where the outputs of the fine-tuned backbone models are passed through a linear projection to lower their dimensions, in contrast to single-vector methods where the original model dimensions are frequently used [3, 7]. These representations are then subsequently used at retrieval-time to compute fine-grained interactions between documents, using the MaxSim operator, further explained in Section 3.1.

Currently, all existing multi-vector models largely follow variations of the original ColBERT architecture, and tweaks have largely focused on the use of different backbone models to unlock novel modalities or context length capabilities [8, 9], better training methods [10, 11] or the introduction of modality-specific components [12], with all models adopting a form of the original architecture, passing the final backbone model's hidden states to a single-layer linear projection to obtain the final output representations.

LIR Submission
ben@mixedbread.com (B. Clavié)

In parallel, much work in deep learning has focused on further our understanding and improving the architecture of neural models [13, 14]. A large stream of this work has been the exploration of the impact of the feedforward block [15, 16], of which ColBERT's single-layer linear projection is the simplest form, as well as how to design better ones [17, 18, 19, 20]. While not immediately applicable, recent work in information retrieval exploring the use of hypernetwork as document encoders further informs this point, showing that better feedforward layer construction results in better relevance scoring [21]. However, there currently has been no in-depth study evaluating the impact, or lack thereof, of ColBERT's final projection on downstream performance.

1.1. Contributions

In this paper, we seek to explore whether or not different projection heads could result in greater downstream ColBERT performance.

We first highlight the impact that the MaxSim operator has on gradient flow, before discussing the potential limits of single-layer projections which can be further compounded by this gradient flow.

Building on the existing deep learning literature, we then propose a series of modifications to the final feedforward block of ColBERT models and demonstrate how their properties could result in improved retrieval performance.

We empirically demonstrate the correctness of our hypothesis, showing that improved projection heads consistently outperform the widely-used single-layer projection on all evaluated benchmarks, with the best-performing variant improving overall performance by over 2NDCG@10 points when averaged over multiple common benchmarks.

Finally, we explore individual factors contributing to this improved performance and further demonstrate that projection head design matters, with certain "improvements" resulting in worsened performance, likely due to the conflicting theoretical properties manifest during training.

2. Related Work: Improving Multi-vector Retrieval

Since the original release of ColBERT, substantial work has focused on improving the retrieval performance of multi-vector models, and extending them for additional uses. Notably among those, ColBERTv2 [11] introduced significant performance gains by leveraging knowledge distillation over a large number of teacher-scored documents for each query. JaColBERTv2.5 [10] and Jina-Colbert-v2 [22] subsequently introduced further refinement to the training process, showing strong empirical downstream gains.

In the meantime, the multi-vector retrieval paradigm has been demonstrated to be easily transposable to various domains, reaching strong multilingual [23, 22], cross-lingual [24] results, but also modalities, with multi-vector models reaching state-of-the-art performance in text→image [8] and text→video [12] retrieval without any major changes to the underlying late-interaction mechanism. Further explorations into multi-vector multimodal retrievers have recently highlighted remarkable parameter-efficiency, with 300 million parameters multi-vector retrievers reaching error reduction rates of over 30% compared to similarly trained single-vector retrievers using the same backbone, almost matching the performance of models with ten times more parameters [25].

Finally, while initially limited due to its considerable storage requirements, subsequent research has considerably improved the usability of late-interaction methods by targeting efficiency improvements, with aggressive quantization in ColBERTv2 [11], better indexing methods such as PLAID [26] and WARP [27], among others, vector count reduction via near-lossless pruning [28, 29] or using a fixed number of representative tokens[30, 31]. The combination of these methods have led to multi-vector models being a viable option for many uses, and it currently stands as one of the main paradigms studied in neural IR research [11, 23, 24, 8, 32, 33, 34].

3. Theoretical Limitations of Current Methods

ColBERT, and all existing related models building on it, such as multi-lingual [35, 10] or multi-modal variants [8] use a simple mechanism to produce token-level representations: the hidden states of the final layer of a pre-trained backbone model, such as BERT [2] or PaliGemma [36] are passed through a single linear projection $h(x) = xW$ where $W \in \mathbb{R}^{d \times k}$ to reduce their dimension from d to k, with k most commonly set to 128 [11, 8, 37], before L2-normalizing the output:

$$\hat{q}_i = \frac{h(q_i)}{\|h(q_i)\|_2}, \quad \hat{d}_j = \frac{h(d_j)}{\|h(d_j)\|_2} \tag{1}$$

3.1. MaxSim

3.1.1. Definition

Using these embeddings, multi-vector retrieval techniques compute relevance scores using the MaxSim operation[1] [6]. MaxSim computes the cosine similarity between each query token and every document token, retains only the highest similarity for each query token, and sums these maximum similarities as the final relevance score.

$$\begin{aligned} \text{MaxSim}(q, d) = \sum_{i=1}^{m} \max_{1 \le j \le n} \hat{q}_i^\top \hat{d}_j \\ \text{where } m \text{ denotes the number of query tokens,} \\ \text{and } n \text{ denotes the number of document tokens.} \end{aligned} \tag{2}$$

3.1.2. Maxsim's Gradient Flow

Despite strong empirical performance, MaxSim limits the information that flows back through the model during training. Let the winning document token for query token i be:

$$j^*(i) = \arg\max_{1 \le j \le n} \hat{q}_i^\top \hat{d}_j \tag{3}$$

Through the chain rule for max operations, we can observe that during the training phase of the model, gradients during backpropagation [40] will only flow through winning tokens:

$$\frac{\partial \text{score}}{\partial \hat{q}_i} = \hat{d}_{j^*(i)}, \quad \frac{\partial \text{score}}{\partial \hat{d}_j} = \begin{cases} \hat{q}_i & \text{if } j = j^*(i) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

This creates an information bottleneck: the gradient with respect to $\hat{d}_j$ is nonzero only when $j = j^*(i)$, meaning only tokens achieving the maximum similarity for at least one query token contribute to learning. Similarly, each $\hat{q}_i$ receives gradient only from its winning $\hat{d}_{j^*(i)}$. As a result, only a small subset of token pairs contribute to each optimization step, restricting the signal path for parameter updates. We refer to this as the "winner-takes-all" mechanism.

3.2. Potential Limits of Token-level Linear Projections

While computationally efficient, the single-layer projection used in existing multi-vector models applies the same transformation to every token, regardless of content or role in matching. Indeed, a linear head $h(x) = xW$ induces a single transformation matrix W used uniformly for all tokens. After L2-normalization, cosine similarity is measured under a *fixed* metric:

$$\text{sim}(\hat{q}_i, \hat{d}_j) = \frac{(q_i W)(d_j W)^\top}{\|q_i W\|_2 \, \|d_j W\|_2} = \frac{q_i^\top M d_j}{\sqrt{q_i^\top M q_i} \sqrt{d_j^\top M d_j}}, \qquad M := WW^\top \succeq 0. \tag{5}$$

[1] Or an approximation thereof [38, 39].

However, in practice, MaxSim rewards high peak similarities through its winner-takes-all mechanism, which can conflict with this mapping: Consider the trace constraint $\text{tr}(M) = k$, which is enforced by dimensionality reduction and weight decay during training. With orthonormal directions $e_1, e_2, \ldots, e_d$ representing different semantic dimensions, we have:

$$\sum_{i=1}^{d} e_i^\top M e_i = \text{tr}(M) = k \tag{6}$$

For tokens aligned along direction e_i, their maximum achievable similarity after normalization is proportional to $e_i^\top M e_i$. To serve all token types adequately, M must allocate some weight to every relevant direction, preventing it from concentrating strongly in any subset. This forced spreading theoretically yields lower peaks, as M's eigenvalues are distributed rather than concentrated.

Under MaxSim's winner-takes-all supervision (Eq. (4)), frequent winners pull M toward their preferred directions, but the single M must still maintain some support for all directions to avoid completely failing on certain token types. This creates a tension between the optimization objective which favours peaked distributions and the architectural constraint encouraging spreading through a single global metric.

4. Alternate ColBERT Projections and Their Expected Effects

The limitations highlighted above have not stopped ColBERT models and the associated MaxSim operators to empirically yield strong results across modalities, both in and out of domain. These results are in line with our assumption: while single-matrix projections face limitations, they are mitigated by the representation capabilities of the underlying Transformer-based [1] pre-trained backbone models, and the lack of a sharpening effect would not be sufficient to render performance non-competitive.

However, we theorise that straightforward modifications, borrowed from the greater deep learning community practices, could help ColBERT models further alleviate the limitations of their naive projection mechanism. Specifically, we believe that factorization benefits that arise from modest model depth alone would yield considerable cross-domain improvements.

We further propose the use of residual skip-connections as part of these multi-layered projection, to allow the projection to focus on producing a sharpening effect while being able to rely on the backbone models' original representations to stabilise the final embeddings.

We also investigate the use of various forms of non-linearities, through the use of common non-linear activation functions, as well Gated Linear Units [17], a widely-used alternative to the traditional feedforward block that introduces an additional non-linear gating [18].

All of these mechanisms are commonly used as part of modern deep model architectures, with model depth thought to contribute to downstream performance more than model width [41, 42] and various forms of non-linearity being considered key to model feedforward layers [17, 43, 44, 19].

4.1. Depth Introduces Sharpening-Improving Factorization

Multi-layer feedforward networks (FFNs) are constructed by simply stacking linear layers, with an activation function applied to the output of a layer before being passed to the next one. In their simplest form, the activation function can be a simple Identity function, in which case the output of an intermediate layer in dimension m is passed as-is to the next:

$$h_{\text{FFN}}(x) = \phi(xW_1)W_2, \quad W_1 \in \mathbb{R}^{d\times m}, W_2 \in \mathbb{R}^{m\times k} \quad \text{where } \phi \text{ is an activation function} \tag{7}$$

It is also common for multilayered feedforward blocks to adopt a so-called bottleneck design, following the original Transformer [1], where the first projection expands to a higher dimension before the final layer projects back to the desired output dimension. We define the projection scale ρ (rho), which controls the intermediate dimension. Given an input $\mathbf{x} \in \mathbb{R}^d$, the bottleneck operations are:

$$h = \phi(W_{\text{up}}x + b_{\text{up}}), \quad W_{\text{up}} \in \mathbb{R}^{d\times m} \tag{8}$$

$$y = W_{\text{down}} h + b_{\text{down}}, \quad W_{\text{down}} \in \mathbb{R}^{d \times m} \tag{9}$$

where m is an intermediate dimension controlled by ρ and defined as $\rho \times d$, $\mathbf{h} \in \mathbb{R}^{dm}$ is the intermediate representation with expanded dimensionality. The first projection (upcasting) expands the dimension from d to m, while the second projection (downcasting) reduces it back to d in the case of an intermediate layer, or k if it is the final layer of a down-projection FFN, as it is in the ColBERT context.

Even with the activation function ϕ defined as the identity function rather than a non-linear activation function, we suggest that the factorization introduced by the addition of an additional layer to the ColBERT projection head would lead to two improvements which benefit MaxSim: increased spectral concentration and better gradient aggregation.

4.1.1. Spectral concentration

All standard training methods for ColBERT employ weight decay [45], which applies L_2 regularization to model weights. In practice, weight decay on factors $\|W_1\|_F^2 + \|W_2\|_F^2$ implicitly regularizes the nuclear norm $\|W_1 W_2\|_*$, encouraging low effective rank. This occurs because for any factorization:

$$\|W_1 W_2\|_* \leq \|W_1\|_F \|W_2\|_F \leq \frac{1}{2}(\|W_1\|_F^2 + \|W_2\|_F^2) \tag{10}$$

For a rank-r approximation with singular values $\sigma_1 \geq \ldots \geq \sigma_r$, the trace constraint from dimensionality reduction gives $\sum_i \sigma_i^2 = k$. Lower rank solutions concentrate this "budget" into fewer, larger singular values, yielding:

$$\max_{\|v\|=\|u\|=1} v^T W_1 W_2 u = \sigma_1 \gg \frac{k}{\sqrt{d}} \tag{11}$$

This concentration effect encourages projections to be concentrated towards fewer singular directions, leading to sharper, or "peakier", token embeddings with higher potential maximum similarities, thus directly benefiting from MaxSim's winner-takes-all effect.

4.1.2. Better handling of gradient aggregation

The factorized structure improves conditioning for aggregating the many sparse rank-1 updates from MaxSim. Under MaxSim's gradient flow (Eq. (4)), each winning pair $(q_i, d_{j^*(i)})$ contributes a rank-1 update to the projection. With a single matrix W, these updates directly compete:

$$\Delta W \propto \sum_{\text{winners}} d_{j^*(i)} q_i^T \tag{12}$$

In contrast, factorization $W_1 W_2$ creates an intermediate representation space of dimension h where updates are first aggregated in W_1 before being projected by W_2. This two-stage process allows the model to learn shared intermediate features that benefit multiple token types, rather than forcing each token type to claim dimensions in the final space. The intermediate bottleneck could act as a regularizer, encouraging the discovery of composable features that can be combined differently for different semantic types, without damaging the sharpening that is beneficial for MaxSim.

4.2. Residual connections

Residual connections [46] are frequently used in deep learning, as they have been demonstrated to improve training stability and downstream performance. A residual connection effectively adds the input to the projection's output, with a learned multiplier α:

$$h_{\text{residual}}(x) = x + \alpha \cdot g(x), \quad \text{where } g(x) = xW_1 W_2 \tag{13}$$

In the context of multi-vector retrieval, we believe that residual connections could potentially offer the benefit of enabling greater role decomposition in the learned projections. The effective metric induced by this formulation becomes Eq. (14), where $W = W_1 W_2$ for notational simplicity:

$$h_{\text{residual}} = (I + \alpha W)(I + \alpha W)^T = I + \alpha(W + W^T) + \alpha^2 W W^T \tag{14}$$

This decomposition highlights two complementary components: the identity I preserves the semantic geometry of the fine-tuned backbone model's until the final projection, while the learned term αW theoretically gains greater freedom to focus on amplifying distinctive tokens during the training process by creating an interaction between the original and learned representations. In the context of MaxSim, this allows the model to selectively boost winners through the learned components. We theorise that this can potentially lead to higher peak similarities without sacrificing performance on non-dominant token types.

4.2.1. Residual Connection In 2-Layer FFNs

When implementing residual connections with a 2-layer projection ($d \to m \to k$), we adopt a ResNet-style [46] approach: the input is upcast using an additional layer initialized as an identity matrix, ensuring dimension compatibility while minimally modifying the input. This is necessary because a direct residual connection would require an intermediate downcasting back to d.

4.3. Non-Linearity and Gating

Non-linearity enables *input-dependent* transformations that can selectively emphasize token dimensions. It can be injected via activation functions (ReLU [43], SiLU [19], GELU [44]) or gated blocks such as Gated Linear Units (GLU) [17].

4.3.1. Multi-layer block with activations.

As introduced in Section 4.1, a non-linear activation can be applied to the intermediate layer:

$$h_{\text{FFN}}(x) \;=\; \phi(xW_1)\,W_2, \qquad W_1 \in \mathbb{R}^{d\times h},\;\; W_2 \in \mathbb{R}^{h\times k}, \tag{15}$$

followed by L2-normalization as in Eq. (1). The non-linear ϕ induces an input-dependent Jacobian:

$$J_{h_{\text{FFN}}}(x) \;=\; \frac{\partial h_{\text{FFN}}(x)}{\partial x} \;=\; W_1\,\mathrm{Diag}\big(\phi'(xW_1)\big)\,W_2. \tag{16}$$

Here $\phi'(xW_1)$ gates columns of W_1 before mixing by W_2. After normalization, the local cosine geometry depends on $J^\top J$, enabling token-specific emphasis and potentially greater similarity peaks rewarded by MaxSim.

4.3.2. Gated Linear Units (GLU)

GLUs introduce a multiplicative gate modulating a value stream:

$$h_{\text{GLU}}(x) \;=\; \big(xW_v\big) \;\odot\; \psi\big(xW_g\big), \qquad W_v \in \mathbb{R}^{d\times k},\;\; W_g \in \mathbb{R}^{d\times k}, \tag{17}$$

where ψ is the gating nonlinearity and $\odot$ denotes elementwise multiplication. The original formulation uses $\psi = \sigma$ (sigmoid) [17], with subsequent variants replacing it with common activations: ReGLU (ψ = ReLU), GEGLU (ψ = GELU), and SwiGLU (ψ = SiLU) [18]. Performance between variants has been shown to vary, and the reasons remain poorly understood.

Non-linearity even with identity gate. Finally, it is worth noting that even with an identity gate ($\psi(u) = u$), GLU introduces non-linearity nonetheless, where GLU with identity gating reduces to a *bilinear* layer that introduces pairwise feature interactions $x_i x_j$:

$$\big(h_{\text{GLU}}(x)\big)_k \;=\; \big(x^\top W_v^{(:,k)}\big)\,\big(x^\top W_g^{(:,k)}\big) \;=\; x^\top\Big(W_v^{(:,k)} W_g^{(:,k)^\top}\Big)\,x, \tag{18}$$

Therefore, in our context, even an identity-gated GLU would result in introducing non-linearity through quadratic feature interactions. This mechanism could potentially capture more complex

semantic relationships between token dimensions than linear projections alone, creating a situation in which the projection could learn that certain feature combinations are particularly indicative of relevance, should that be the case in the backbone model's hidden states. Under the lense of MaxSim, this means the projection could learn to amplify similarities when specific pairs of features co-occur, potentially creating sharper peaks.

4.3.3. Potential Effects of Non-Linearity

Potentially Increased Sharpening Eqs. (16) and (17) show that both non-linearities and gating enable input-dependent reweighting, which can concentrate each token's mass along a few decisive directions, increasing the dynamic range of $\hat{q}_i^\top \hat{d}_j$ and producing clearer winners $j^*(i)$. Since embeddings are L2-normalized, absolute scale changes are suppressed, but *directional* changes induced by non-linear activations still alter cosine similarity. The effective metric is governed by $J^\top J$ for FFNs or the product-rule Jacobian of Eq. (17) for GLU, facilitating sharpening via directional emphasis.

Potential Negative Effects Harder sparsity from non-linearity and gating can cause over-sharpening, increasing winner instability and amplifying the winner-takes-all bottleneck of Eq. (4), potentially hindering convergence [40]. Additionally, certain non-linearities can dampen the learning signal: sigmoid gates risk saturation in extreme regions, while ReLU zeros gradients for negative inputs, potentially blocking signal from reaching earlier layers even for MaxSim-selected pairs.

5. Experimental Setting

Our aim is to be thorough in our experiments, evaluating all the combinations of settings described in Section 4 in a way that is both significant and applicable to state-of-the-art training methods, to demonstrate that potential improvements are not dependent on a weak baseline. In this section, we present the training decisions made to ensure both of these while keeping compute requirements reasonable.

5.1. Implementation

Building on the justification presented in Section 4, we present an experimental framework which will allow us to measure the effects of our proposed model modifications. Specifically, we seek to measure the impact of introducing various projection blocks to replace the currently used linear projection.

We extend the PyLate library [47], a widely-used framework for the training and evaluation of multi-vector retrieval models, to support modular projection blocks. Specifically, we implement the ability to control the following parameters:

- **Projection depth**: How many feedforward blocks should be used for the projection.
- **Gated Linear Units**: Whether to use GLU layers instead of traditional feedforward layers without gating.
- **Residual connection**: Whether there should be a skip-connection between layers or not, as presented in Section 4.2.
- **Activation function**: The activation function to be applied to the output of non-final layers.
- **Projection Scale**: As presented in Section 4.1, it is common for feedforward layer to adopt a larger scale for intermediate projections. For the sake of thoroughness, we ablate the effect of a using non-scaled up projections, as well as projections where the intermediate layer's dimension is twice that of the input dimension, on retrieval performance.

Despite many studies exploring activation functions, the empirical performance of different activation functions remains fluctuating and largely task-dependent, without, as of yet, clear general patterns or theoretical reasons as to why performance fluctuates [48]. Identifying these exact factors remains out

of the scope of this study, and we follow existing practice [18] in comparing empirically performance among multiple activations, among the most commonly used ones for natural language processing tasks: Identity (no activation), ReLU, GELU, SiLU, and, for the sake of thoroughness, GLU layers with their original sigmoid gating, all briefly presented in Section 4.3.

5.2. Training Setting

Base Model Smaller models, such as MiniLM [49], have repeatedly been demonstrated to be well-suited for retrieval tasks, reaching strong performance. This is especially true for ColBERT models, where recent empirical results have shown that even a 4-million parameter ColBERT model could be competitive with models over 30 times larger [50]. Moreover, it is common to conduct experiments on smaller models, with scaling laws showing that their results are extremely strongly correlated with the results of larger variants [51]. As such, we choose to use the 32M parameter variant of Ettin as our backbone model. Ettin [52] is an improved reproduction across model sizes of ModernBERT [53], itself a variant of the original BERT [2] incorporating recent advances in model training.

Training Setting We conducted limited sweeps over hyperparameters on a handful of settings. Our findings largely match previous research, with a batch size of 64, a learning rate of $1e-4$ with a linear decay schedule following a warmup phase for 10% of total training steps reaching consistently strong performance. As such, we adopt these settings for all experiments. Following ColBERT training efforts since ColBERTV2 [11], we adopt a knowledge distillation loss where the training objective is to minimize the difference between the score distribution of the student and teacher models. We follow standard existing practice use Kullback–Leibler divergence (KL-Div) between the student and teacher scores as our loss function, which has empirically been shown to be well suited for ColBERT training [10, 11] and retrieval models in general [54, 55].

Data To increase applicability of our method to the real-world, we train our model using a 640,000 sample of the data commonly used to train current state-of-the-art ColBERT models [56, 50, 37]. This training set is effectively a downsample of ColBERTV2's large original corpora of 64-way training tuples from MS Marco [57], each composed of a query, a positive example, and 63 negative examples mined via MiniLMv2 [11, 49] with teacher scores generated by a reranker. Our sampled set instead uses 640,000 randomly selected 16-way tuples, reducing the number of negatives to 15, with scores generated by bge-reranker-m3 [58]. This downsampling has previously been shown to be sufficient to yield results that are significantly correlated with performances obtained when training on 10x more data [10, 56], while significantly lowering training compute requirements.

5.3. Consistency

The reproducibility of experiments in machine learning is an often-discussed topic, with studies showing that reported results are often, even if involuntarily, cherry-picked, with more neutral evaluation methods showing different results [59].

We conduct all training and evaluation runs five times, using five individual random seeds for PyTorch seeding, parameter initialization and dataset shuffling: 1, 42, 1337, 1789 and 1861. All results are reported as the mean of the checkpoints resulting from the five seeds, across three separate indexing runs each to eliminate indexing variance.

5.4. Evaluation Settings

Data We report results across a set of commonly used, standardised benchmarks: TREC-DL19 and TREC-DL20, as well as the high-quality search subsets of the BEIR evaluation suite [60]: SciFact [61], TREC-Covid [62], FiQA2018 [63] and NFCorpus [64]. We select these benchmarks as they cover multiple domains and are widely used and generally considered to be high quality collections, without incurring the computational cost of running full BEIR evaluations across multiple seeds for all evaluated settings.

Table 1
Main results showing a comparison of the linear baseline with various depth for the most common settings for each FFN family across model depths. All results reported are NDCG@10 averaged across 5 training runs. Results in **bold** are the best overall results and results underlined are results which outperform the baseline projection. †denotes statistical significance with $p < 0.05$.

Model	DL19	DL20	COVID	SciFact	NFC	FiQA	Avg.
Baseline							
Linear Projection	0.6857	0.7081	0.6831	0.6728	0.3355	0.3311	0.5694
FFN (identity)							
Depth 2	**0.7095**†	0.7040	0.7402†	**0.7076**†	0.3382	**0.3455**†	**0.5908**†
Depth 3	0.6931†	0.7012	0.7452†	0.7044†	0.3369	0.3369†	0.5861†
Depth 4	0.6911†	0.7082	0.7338†	0.6891†	0.3356	0.3414†	0.5864†
GLU (sigmoid)							
Depth 2	0.6944†	0.7022	0.7448†	0.7049†	0.3383	0.3435†	0.5880†
Depth 3	0.6995†	**0.7108**	0.7274	0.7002†	0.3388	0.3406†	0.5862†
Depth 4	0.6966	0.7061	**0.7465**†	0.7010†	**0.3399**	0.3461†	0.5893†

Indexing and Searching All evaluations are ran using the standardised ColBERTv2 [11]+PLAID [26] indexing method. PLAID is an optimized index type built upon an inverted file index coupled with aggressive product quantization, allowing for fast multi-vector retrieval while reducing index sizes. We employ 4-bit quantization for individual token vectors follow the optimal parameters identified by a recent thorough PLAID reproduction study [34] at inference time. Query length is set to 32 and document length to 300, following commonly used settings [6]. All indexes are created and searched through using the PyLate library [47], with DL-19 and DL-20 loaded separately via ir-datasets [65].

6. Experimental Results

In this section, we will empirically explore the performance of our proposed projection modification.

We will first highlight a high-level overview of so-called "canonical", that is, using widely used default parameters, FFN and GLU blocks, comparing their performance to that of the commonly used single-layer linear projection baseline.

Subsequently, we will present the results of targeted evaluations seeking to further explore individual factors that impact well-performing model variants, such as the choice of activation function (Sec. 6.2), the use of residual connections(Sec. 6.4) and of a higher intermediate projection dimension(Sec. 6.3).

6.1. Overall Results

Table 1 presents a comparison of the performance of the commonly used linear projection against a set of varying depths FFN and GLU projections. For the ease or readability, we provide only the most standardised version of these projection blocks: residual connections are used, a projection scale of 2.0, i.e. twice the input dimension, is used in intermediate layers and we do not use a non-linear activation function for the FFN blocks, while we use the canonical sigmoid gate for GLU blocks.

The results starkly demonstrate that these projection variants significantly outperform the baseline projection on all datasets evaluated, with the exception of DL20 where the performance of some projections is very slightly inferior to that of the baseline. In this context, it is worth noting that both DL19 and DL20 are in-domain datasets, using the same MS Marco [57] document collection that was used to train the model, while the other four datasets are fully out-of-domain. Under this light, we can note that alternate projections observe no degradation, and even gains on DL19, while in-domain, while noticeably improving performance on 3 out of 4 OOD evaluations, with moderate gains on the fourth.

The significant gains achieved on Trec-COVID, SciFAct and FiQA appear to support the theory expressed in Section 4.1, in which we propose that alternate projections would be particularly useful in

Table 2
Comparison of activation functions for FFN and GLU projection variants with all other parameters kept equal, averaged across five model checkpoints. All results are NDCG@10. Results in **bold** are the best overall per column, and results underlined indicate they outperform the baseline.

Model	DL19	DL20	COVID	SciFact	NFC	FiQA	Avg.
Baseline							
Linear Projection	0.6857	0.7081	0.6831	0.6728	0.3355	0.3311	0.5694
FFN							
$FFN_{Identity}$	**0.7095**	0.7040	**0.7402**	**0.7076**	**0.3382**	**0.3455**	**0.5908**
FFN_{ReLU}	0.6929	0.7005	0.7237	0.7030	0.3381	0.3436	0.5836
FFN_{GELU}	0.6957	0.6967	0.6905	0.7048	0.3357	0.3437	0.5778
FFN_{SiLU}	0.6791	0.6968	0.7124	0.7009	0.3354	0.3326	0.5762
GLU							
$GLU_{Sigmoid}$	0.6944	0.7022	**0.7448**	0.7049	0.3383	0.3435	0.5880
$GLU_{Identity}$	0.6898	**0.7105**	0.7394	0.6993	0.3381	0.3431	0.5869
GLU_{ReLU}	0.6891	0.7034	0.7196	0.6999	0.3377	0.3428	0.5829
GLU_{GELU}	**0.7082**	0.7038	0.7371	**0.7061**	0.3380	**0.3449**	0.5892
GLU_{SiLU}	0.6969	0.7007	0.7306	0.7008	0.3372	0.3409	0.5845

facilitating the representation of domain-specific vocabulary, thus increasing performance.

Overall, we note that these results support the idea that the use of alternate projection is an underexplored, "almost-free lunch" to improve the retrieval performance of ColBERT models.

6.2. Activation Functions and Non-Linearity

Table 2 presents the results of varying activation functions, with all other parameters being fixed to the best performing depth 2 variants presented in Table 1.

For FFN blocks, the use of activation function appears to be a net negative in terms of performance, across all datasets. While all activation functions continue to outperform the baseline, the gains are less pronounced. As such, it seems to indicate that the potentially sharpening effects of adding non-linearity do not outweigh their potential negative effects and rather ends up dampening the positive effects of other modifications.

For GLU blocks, which as indicated in Section 4.3.2 are non-linear no matter the activation function, it seems that the choice of activation function has only moderate impact, with all variants reaching broadly similar results, even if GELU pulls slightly ahead. Interestingly, GLU variants, while ultimately all outperformed by the $FFN_{identity}$ variant, reach more consistent results than non-linear FFN blocks and outperform the baseline in all evaluated settings. This seems to suggest that GLU layers do improve the quality of representations, although in a way directly tied to the gating mechanism.

Overall, these results appear to indicate that non-linear activation functions do not, overall, contribute to improving the projection quality of multi-vector retrieval models.

6.3. Upscaling

Next, we focus on the importance of upscaled representations within the intermediate layers. This projection is a common component of the modern Transformer feedforward block design [1], with virtually all modern models adopting it, and has also been shown to improve the performance of even older architectures such as Recurrent Neural Networks [66]. However, as demonstrated by the GLU results above, not all architectural modifications which improve Transformer networks appear to directly translate to improving our considerably-smaller network focused on dimensionality reduction.

Table 3 presents the results of using an upscaled projection with a ρ of 2, meaning that the intermediate representations' dimension is twice that of the inptu dimension, compared to a ρ of 1 where intermediate representations are not upscaled. The overall results vary setting by setting, but ultimately appear to

Table 3
Comparison of projection scale ρ across depths for FFN and GLU projection variants with all other parameters kept equal. All results are NDCG@10. Results in **bold** are the best overall per column, and results underlined outperform the baseline.

Model	DL19	DL20	COVID	SciFact	NFC	FiQA	Avg.
Baseline							
Linear Projection	0.6857	0.7081	0.6831	0.6728	0.3355	0.3311	0.5694
FFN (identity)							
ρ=1, Depth 2	0.6905	0.7003	0.6881	0.7034	0.3391	0.3391	0.5767
ρ=2, Depth 2	**0.7095**	0.7040	0.7402	0.7076	**0.3382**	**0.3455**	**0.5908**
ρ=1, Depth 3	0.6859	0.6921	0.6488	0.6994	0.3351	0.3322	0.5656
ρ=2, Depth 3	0.6931	0.6997	**0.7452**	0.7044	0.3369	0.3369	0.5861
ρ=1, Depth 4	0.6121	0.6037	0.6325	0.6891	0.3281	0.3186	0.5307
ρ=2, Depth 4	0.6911	**0.7082**	**0.7338**	**0.7081**	0.3356	**0.3414**	0.5864
GLU (GELU)							
ρ=1, Depth 2	0.6940	**0.7111**	**0.7405**	**0.7056**	**0.3366**	0.3430	0.5885
ρ=2, Depth 2	0.6979	0.6987	0.7151	0.6990	**0.3373**	0.3401	0.5813
ρ=1, Depth 3	0.6642	0.6400	0.6601	0.6867	0.3313	0.3206	0.5505
ρ=2, Depth 3	**0.7042**	0.7025	0.7371	**0.7056**	**0.3392**	**0.3448**	0.5889
ρ=1, Depth 4	0.6923	0.7000	0.6903	0.6983	0.3335	0.3315	0.5743
ρ=2, Depth 4	0.7005	**0.7099**	0.7197	**0.7099**	0.3357	0.3308	0.5844

Table 4
Comparison of models with and without residual connections across projection scales ρ for select FFN and GLU configurations. ΔAvg denotes the performance difference between settings, with all else equal (Residual − No Residual). Values outperforming the baseline are underlined, and the overall best result is in **bold**.

Model	DL19	DL20	COVID	SciFact	NFC	FiQA	Avg.	Δ
Baseline								
Linear Projection	0.6857	0.7081	0.6831	0.6728	0.3355	0.3311	0.5694	–
FFN								
Depth 2								
ρ=1.0 No Residual	0.6907	0.7001	0.6883	0.7032	0.3390	0.3390	0.5769	–
ρ=1.0 Residual	0.6905	0.7003	0.6881	0.7034	0.3391	0.3391	0.5767	−0.0002
ρ=2.0 No Residual	0.6875	0.7030	0.7336	0.6954	0.3368	0.3342	0.5707	–
ρ=2.0 Residual	0.7095	0.7040	0.7402	0.7076	0.3382	0.3455	**0.5908**	+0.0201
Depth 3								
ρ=1.0 No Residual	0.6871	0.6929	0.6493	0.6978	0.3349	0.3315	0.5889	–
ρ=1.0 Residual	0.6859	0.6921	0.6488	0.6994	0.3351	0.3322	0.5656	−0.0233
ρ=2.0 No Residual	0.6849	0.6938	0.7413	0.7035	0.3359	0.3329	0.5730	–
ρ=2.0 Residual	0.6931	0.6997	0.7452	0.7044	0.3369	0.3369	0.5861	+0.0131
GLU								
Depth 2								
ρ=1.0 No Residual	0.6932	0.6991	0.6875	0.7013	0.3341	0.3345	0.5714	–
ρ=1.0 Residual	0.6881	0.6924	0.6816	0.6987	0.3333	0.3328	0.5447	−0.0267
ρ=2.0 No Residual	0.6934	0.6965	0.7108	0.6951	0.3365	0.3388	0.5747	–
ρ=2.0 Residual	0.6979	0.6987	0.7151	0.6990	0.3373	0.3401	0.5813	+0.0176
Depth 3								
ρ=1.0 No Residual	0.6660	0.6504	0.6659	0.6851	0.3317	0.3218	0.5600	–
ρ=1.0 Residual	0.6642	0.6400	0.6601	0.6867	0.3313	0.3206	0.5505	−0.0095
ρ=2.0 No Residual	0.7007	0.6980	0.7342	0.7039	0.3386	0.3428	0.5824	–
ρ=2.0 Residual	0.7042	0.7025	0.7371	0.7056	0.3392	0.3448	0.5889	+0.0065

strongly favor upscaling. Interestingly, while it does not appear to considerably benefit GLU networks at a depth of 2, even resulting in a slight performance decrease, but mitigates large decreases in performance at the deeper depths of 3 and 4. For FFN blocks, results do show a similar preserving effect as depth increases, but a ρ of 2 is superior to the no-upscaling setting across all model depths.

Overall, these results highlight that a higher-dimension intermediate dimension appear to contribute positively to stronger multi-vector retrieval performance, but also appear to have a stabilising effect, with the performance of similar model families using ρ=2 remaining more consistent across model depths while it greatly fluctuates without these upscaled representations.

6.4. Residual Connections

Finally, we attempt to identify the effect of residual connections, and confirm their theoretical benefits of residual connections presented in 4.2. Table 4 presents a comparison of the effect of the use of residual connections on two different checkpoint families, across both intermediate projection scales.

The results, presented in Table 4, show an interesting phenomenon, which shines additional light on the seemingly stabilizing effect of larger intermediate projections highlighted above. Indeed, it appears that the use of residual connections consistently reduces the retrieval performance of models without upcasting. This effect appears milder in the simpler setting of the Depth 2 FFN block, our most simple model design, where the use of a residual projection has a negligible impact on performance, but is very noticeable in every other evaluated setting, resulting in large decreases.

On the other hand, when combined with a ρ value of 2, where intermediate projections are upscaled, the use of residual connections significantly improve performance in all cases. Additionally, it seems to once again produce a stabilising effect, reducing the gap between various projection variants.

These results seem to the support the intuition expressed in Section 4.2 in the sense that combining better projections with residual connections as part of these projections appear to result in greater performance, potentially as a result of better leveraging and ”improving” the backbone model’s projections rather than aggressively modifying them.

7. Conclusion

In this paper, we demonstrated the learning limitations imposed by the MaxSim operator of multi-vector retrieval models. We subsequently the hypothesis that these limitations are potentially harmful to downstream performance when combined with the simple, single-layer linear projection that is commonly used as the final layer of all existing multi-vector retrieval models. We then proposed a series of improvements to the projection blocks of multi-vector models, discussing their potential benefits and limitations. Building on this proposal, we then trained numerous ColBERT models with all combinations of our proposed modifications. Our results, evaluated across 5 independent training runs for each setting, demonstrate that the use of alternate projection heads appear to improve multi-vector performance across a variety of settings, with the best variant increasing performance by an average of over 2NDCG@10 points.

Finally, our exploration studies focus on independent modifications in order to better understand their role in this improved performance. We show that non-linearity, introduced either via GLU blocks or common activation functions, is not a significant performance driver, but that the use of modern FFN blocks with intermediate dimension upcasting and residual connections is crucial to our results.

While we propose theoretical explanations for these results, the learning process of Neural IR models, and particularly multi-vector models, is still poorly understand. We believe our empirical results are only an early step in the design of better multi-vector retrieval model architecture, and hope that they will support future work in better understanding their underlying mechanisms.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Advances in neural information processing systems 30 (2017).

[2] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, 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), 2019, pp. 4171–4186.

[3] A. Yates, R. Nogueira, J. Lin, Pretrained transformers for text ranking: Bert and beyond, in: Proceedings of the 14th ACM International Conference on web search and data mining, 2021, pp. 1154–1156.

[4] T. Formal, B. Piwowarski, S. Clinchant, 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, 2021, pp. 2288–2292.

[5] T. Wen, Y. Wang, Z. Zeng, Z. Peng, Y. Su, X. Liu, B. Chen, H. Liu, S. Jegelka, C. You, Beyond matryoshka: Revisiting sparse coding for adaptive representation, in: Proceedings of the 42nd International Conference on Machine Learning, Proceedings of Machine Learning Research, 2025. URL: https://arxiv.org/abs/2503.01776. arXiv:2503.01776, oral presentation at ICML 2025.

[6] O. Khattab, M. Zaharia, 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, 2020, pp. 39–48.

[7] L. Wang, N. Yang, X. Huang, B. Jiao, L. Yang, D. Jiang, R. Majumder, F. Wei, Text embeddings by weakly-supervised contrastive pre-training, arXiv preprint arXiv:2212.03533 (2022).

[8] M. Faysse, H. Sibille, T. Wu, B. Omrani, G. Viaud, C. HUDELOT, P. Colombo, Colpali: Efficient document retrieval with vision language models, in: The Thirteenth International Conference on Learning Representations, 2025. URL: https://openreview.net/forum?id=ogjBpZ8uSi.

[9] A. Chaffin, Reason-moderncolbert, 2025. URL: https://huggingface.co/lightonai/Reason-ModernColBERT.

[10] B. Clavié, Jacolbertv2. 5: Optimising multi-vector retrievers to create state-of-the-art japanese retrievers with constrained resources, Journal of Natural Language Processing 32 (2025) 176–218.

[11] K. Santhanam, O. Khattab, J. Saad-Falcon, C. Potts, M. Zaharia, 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, 2022, pp. 3715–3734.

[12] A. Reddy, A. Martin, E. Yang, A. Yates, K. Sanders, K. Murray, R. Kriz, C. M. de Melo, B. Van Durme, R. Chellappa, Video-colbert: Contextualized late interaction for text-to-video retrieval, in: Proceedings of the Computer Vision and Pattern Recognition Conference, 2025, pp. 19691–19701.

[13] A. R. Sajun, I. Zualkernan, D. Sankalpa, A historical survey of advances in transformer architectures, Applied Sciences 14 (2024) 4316.

[14] L. Balderas, M. Lastra, J. M. Benítez, Optimizing dense feed-forward neural networks, Neural Networks 171 (2024) 229–241.

[15] I. Gerber, Attention is not all you need: The importance of feedforward networks in transformer models, arXiv preprint arXiv:2505.06633 (2025).

[16] M. Geva, R. Schuster, J. Berant, O. Levy, Transformer feed-forward layers are key-value memories, in: M.-F. Moens, X. Huang, L. Specia, S. W.-t. Yih (Eds.), Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 2021, pp. 5484–5495. URL: https://aclanthology.org/2021.emnlp-main.446/. doi:10.18653/v1/2021.emnlp-main.446.

[17] Y. N. Dauphin, A. Fan, M. Auli, D. Grangier, Language modeling with gated convolutional networks, in: Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML'17, JMLR.org, 2017, p. 933–941.

[18] N. Shazeer, Glu variants improve transformer, arXiv preprint arXiv:2002.05202 (2020).

[19] S. Elfwing, E. Uchibe, K. Doya, Sigmoid-weighted linear units for neural network function approximation in reinforcement learning, Neural networks 107 (2018) 3–11.
[20] P. Ramachandran, B. Zoph, Q. V. Le, Searching for activation functions, arXiv preprint arXiv:1710.05941 (2017).
[21] J. Killingback, H. Zeng, H. Zamani, Hypencoder: Hypernetworks for information retrieval, in: Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '25, Association for Computing Machinery, New York, NY, USA, 2025, p. 2372–2383. URL: https://doi.org/10.1145/3726302.3729983. doi:10.1145/3726302.3729983.
[22] H. Xiao, B. Wang, R. Jha, Jina-colbert-v2: A general-purpose multilingual late interaction retriever, in: Proceedings of the Fourth Workshop on Multilingual Representation Learning (MRL 2024), 2024, pp. 159–166.
[23] A. Louis, V. Saxena, G. van Dijck, G. Spanakis, Colbert-xm: A modular multi-vector representation model for zero-shot multilingual information retrieval, arXiv preprint arXiv:2402.15059 (2024).
[24] S. Nair, E. Yang, D. Lawrie, K. Duh, P. McNamee, K. Murray, J. Mayfield, D. W. Oard, Transfer learning approaches for building cross-language dense retrieval models, in: European Conference on Information Retrieval, Springer, 2022, pp. 382–396.
[25] P. Teiletche, Q. Macé, M. Conti, A. Loison, G. Viaud, P. Colombo, M. Faysse, Modernvbert: Towards smaller visual document retrievers, 2025. URL: https://arxiv.org/abs/2510.01149. arXiv:2510.01149.
[26] K. Santhanam, O. Khattab, C. Potts, M. Zaharia, Plaid: an efficient engine for late interaction retrieval, in: Proceedings of the 31st ACM International Conference on Information & Knowledge Management, 2022, pp. 1747–1756.
[27] J. L. Scheerer, M. Zaharia, C. Potts, G. Alonso, O. Khattab, Warp: An efficient engine for multi-vector retrieval, in: Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '25, Association for Computing Machinery, New York, NY, USA, 2025, p. 2504–2512. URL: https://doi.org/10.1145/3726302.3729904. doi:10.1145/3726302.3729904.
[28] B. Clavié, A. Chaffin, G. Adams, Reducing the footprint of multi-vector retrieval with minimal performance impact via token pooling, 2024. URL: https://arxiv.org/abs/2409.14683. arXiv:2409.14683.
[29] R.-C. Chen, C.-J. Lee, An information-theoretic account of static index pruning, in: Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval, 2013, pp. 163–172.
[30] S. MacAvaney, A. Mallia, N. Tonellotto, Efficient constant-space multi-vector retrieval, in: European Conference on Information Retrieval, Springer, 2025, pp. 237–245.
[31] Z. Xiao, Q. Ma, M. Gu, C.-c. J. Chen, X. Chen, V. Ordonez, V. Mohan, Metaembed: Scaling multimodal retrieval at test-time with flexible late interaction, arXiv preprint arXiv:2509.18095 (2025).
[32] T. Formal, B. Piwowarski, S. Clinchant, A white box analysis of colbert, in: Advances in Information Retrieval: 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28–April 1, 2021, Proceedings, Part II 43, Springer, 2021, pp. 257–263.
[33] S. Hofstätter, O. Khattab, S. Althammer, M. Sertkan, A. Hanbury, Introducing neural bag of whole-words with colberter: Contextualized late interactions using enhanced reduction, in: Proceedings of the 31st ACM International Conference on Information & Knowledge Management, 2022, pp. 737–747.
[34] S. MacAvaney, N. Tonellotto, A reproducibility study of plaid (2024) 1411–1419.
[35] A. Louis, V. K. Saxena, G. van Dijck, G. Spanakis, Colbert-xm: A modular multi-vector representation model for zero-shot multilingual information retrieval, in: Proceedings of the 31st International Conference on Computational Linguistics, 2025, pp. 4370–4383.
[36] L. Beyer, A. Steiner, A. S. Pinto, A. Kolesnikov, X. Wang, D. Salz, M. Neumann, I. Alabdulmohsin, M. Tschannen, E. Bugliarello, et al., Paligemma: A versatile 3b vlm for transfer, arXiv preprint arXiv:2407.07726 (2024).

[37] A. Chaffin, Gte-moderncolbert, 2025. URL: https://huggingface.co/lightonai/GTE-ModernColBERT-v1.
[38] R. Jayaram, L. Dhulipala, M. Hadian, J. D. Lee, V. Mirrokni, Muvera: Multi-vector retrieval via fixed dimensional encoding, Advances in Neural Information Processing Systems 37 (2024) 101042–101073.
[39] J. Lee, Z. Dai, S. M. K. Duddu, T. Lei, I. Naim, M.-W. Chang, V. Zhao, Rethinking the role of token retrieval in multi-vector retrieval, Advances in Neural Information Processing Systems 36 (2024).
[40] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, nature 323 (1986) 533–536.
[41] Y. Tay, M. Dehghani, J. Rao, W. Fedus, S. Abnar, H. W. Chung, S. Narang, D. Yogatama, A. Vaswani, D. Metzler, Scale efficiently: Insights from pretraining and finetuning transformers, in: International Conference on Learning Representations, 2022. URL: https://openreview.net/forum?id=f2OYVDyfIB.
[42] T. Nguyen, M. Raghu, S. Kornblith, Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth, in: International Conference on Learning Representations, 2021. URL: https://openreview.net/forum?id=KJNcAkY8tY4.
[43] B. Xu, N. Wang, T. Chen, M. Li, Empirical evaluation of rectified activations in convolutional network, arXiv preprint arXiv:1505.00853 (2015). URL: https://arxiv.org/abs/1505.00853.
[44] D. Hendrycks, K. Gimpel, Gaussian error linear units (gelus), 2016. URL: https://arxiv.org/abs/1606.08415. arXiv:1606.08415.
[45] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, in: International Conference on Learning Representations, 2019. URL: https://openreview.net/forum?id=Bkg6RiCqY7.
[46] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
[47] A. Chaffin, R. Sourty, Pylate: Flexible training and retrieval for late interaction models, arXiv preprint arXiv:2508.03555, to be published at CIKM 2025 (2025).
[48] S. R. Dubey, S. K. Singh, B. B. Chaudhuri, Activation functions in deep learning: A comprehensive survey and benchmark, arXiv preprint arXiv:2109.14545 (2021). URL: https://arxiv.org/abs/2109.14545.
[49] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, M. Zhou, Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, Advances in Neural Information Processing Systems 33 (2020) 5776–5788.
[50] D. Mezzetti, Colbert-muvera-micro, Hugging Face model repository, 2025. URL: https://huggingface.co/NeuML/colbert-muvera-micro.
[51] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, D. Amodei, Scaling laws for neural language models, arXiv preprint arXiv:2001.08361 (2020).
[52] O. Weller, K. Ricci, M. Marone, A. Chaffin, D. Lawrie, B. V. Durme, Seq vs seq: An open suite of paired encoders and decoders, 2025. URL: https://arxiv.org/abs/2507.11412. arXiv:2507.11412.
[53] B. Warner, A. Chaffin, B. Clavié, O. Weller, O. Hallström, S. Taghadouini, A. Gallagher, R. Biswas, F. Ladhak, T. Aarsen, G. T. Adams, J. Howard, I. Poli, Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference, in: W. Che, J. Nabende, E. Shutova, M. T. Pilehvar (Eds.), Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Vienna, Austria, 2025, pp. 2526–2547. URL: https://aclanthology.org/2025.acl-long.127/. doi:10.18653/v1/2025.acl-long.127.
[54] R. Ren, Y. Qu, J. Liu, W. X. Zhao, Q. She, H. Wu, H. Wang, J.-R. Wen, Rocketqav2: A joint training method for dense passage retrieval and passage re-ranking, in: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, 2021, pp. 2825–2835.
[55] C. Lassance, H. Déjean, T. Formal, S. Clinchant, Splade-v3: New baselines for splade, arXiv preprint arXiv:2403.06789 (2024).
[56] B. Clavié, Small but mighty: Introducing answerai-colbert-small, 2024. URL: https://www.answer.ai/posts/2024-08-13-small-but-mighty-colbert.html.

[57] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, L. Deng, Ms marco: A human-generated machine reading comprehension dataset (2016).

[58] J. Chen, S. Xiao, P. Zhang, K. Luo, D. Lian, Z. Liu, Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation, 2024. arXiv:2402.03216.

[59] J. Dodge, S. Gururangan, D. Card, R. Schwartz, N. A. Smith, Show your work: Improved reporting of experimental results, 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), 2019, pp. 2185–2194.

[60] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, I. Gurevych, Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models, arXiv preprint arXiv:2104.08663 (2021).

[61] D. Wadden, S. Lin, K. Lo, L. L. Wang, M. van Zuylen, A. Cohan, H. Hajishirzi, Fact or fiction: Verifying scientific claims, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020, pp. 7534–7550.

[62] E. Voorhees, T. Alam, S. Bedrick, D. Demner-Fushman, W. R. Hersh, K. Lo, K. Roberts, I. Soboroff, L. L. Wang, Trec-covid: constructing a pandemic information retrieval test collection, in: ACM SIGIR Forum, volume 54, ACM New York, NY, USA, 2021, pp. 1–12.

[63] M. Maia, S. Handschuh, A. Freitas, B. Davis, R. McDermott, M. Zarrouk, A. Balahur, Www'18 open challenge: financial opinion mining and question answering, in: Companion proceedings of the the web conference 2018, 2018, pp. 1941–1942.

[64] V. Boteva, D. Gholipour, A. Sokolov, S. Riezler, A full-text learning to rank dataset for medical information retrieval, in: Advances in Information Retrieval: 38th European Conference on IR Research, ECIR 2016, Padua, Italy, March 20–23, 2016. Proceedings 38, Springer, 2016, pp. 716–722.

[65] S. MacAvaney, A. Yates, S. Feldman, D. Downey, A. Cohan, N. Goharian, Simplified data wrangling with ir_datasets, in: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021, pp. 2429–2436.

[66] S. Merity, Single headed attention rnn: Stop thinking with your head, 2019. URL: https://arxiv.org/abs/1911.11423. arXiv:1911.11423.