Adapting Decoder-Based Language Models for Diverse Encoder Downstream Tasks

Anonymous ACL submission

Abstract

Decoder-based transformers, while revolutionizing language modeling and scaling to immense sizes, have not completely overtaken encoder-heavy architectures in natural language processing. Specifically, encoder-only models remain dominant in tasks like classification, regression, and ranking. This is primarily due to the inherent structure of decoder-based models, which limits their direct applicability to these tasks. In this paper, we introduce *Gemma Encoder*, adapting the powerful Gemma decoder model to an encoder architecture, thereby unlocking its potential for a wider range of non-generative applications. To optimize the adaptation from decoder to encoder, we systematically analyze various pooling strategies, attention mechanisms, and hyperparameters (e.g., dropout rate). Furthermore, we benchmark Gemma Encoder against established approaches on the GLUE benchmarks, and MS MARCO ranking benchmark, demonstrating its effectiveness and versatility.

1 Introduction

Decoder-based language models like Gemma (Gemma Team, 2024b,a) and Gemini (Gemini Team, 2023) have demonstrated remarkable language understanding capabilities. Yet, for many downstream tasks such as classification, regression, and ranking, encoder-based models, particularly those derived from BERT (Devlin et al., 2019) or T5's encoder (Raffel et al., 2020), remain the dominant choice. A key question thus arises: can we effectively adapt the powerful knowledge embedded in decoder-only models to excel in these encoder-centric tasks?

This work addresses this gap by introducing Gemma Encoder, a novel adaptation of the Gemma decoder model designed for encoder-only architectures. We leverage Gemma's pre-trained weights as a strong initialization point, and then strategically modify the architecture and training procedure to optimize performance on downstream tasks. Our approach centers on three key innovations:

First, we augment the model with task-specific pooling and Multi-Layer Perceptron (MLP) layers, exploring various pooling strategies to determine the optimal architecture.

Second, we address the critical impact of attention mechanisms. Gemma's causal attention, ideal for generative tasks, inherently limits its applicability to encoder-based tasks. We demonstrate that simply enabling bidirectional attention during fine-tuning dramatically improves performance.

Third, we investigate the role of dropout. While often omitted during pre-training of modern decoder models, our empirical analysis reveals that incorporating dropout during fine-tuning significantly enhances Gemma Encoder's robustness and generalization ability. We also analyze different padding strategies to understand the effect on Encoder models.

In the rest of this paper, we describe the technical details of these modifications. To validate the effectiveness of our Gemma Encoder, we conduct our experiments on GLUE benchmarks (Wang et al., 2019, 2020), for classification and regression tasks, and the MSMARCO benchmark (Bajaj et al., 2018) for ranking tasks. Our results show that our Gemma Encoder models are able to outperform competitive baselines on these benchmark tasks.

2 Model Adaptation Choices

This work explores adapting decoder-only generative AI models, exemplified by GPT-4 (OpenAI, 2023), Gemini (Gemini Team, 2023), and Gemma (Gemma Team, 2024b,a), for encoder-only tasks such as classification, regression, and ranking. Our adaptation strategy, as shown in Figure 1, centers on three key architectural and training modifications: attention mechanism design, pooling strategies, and the application of dropout.

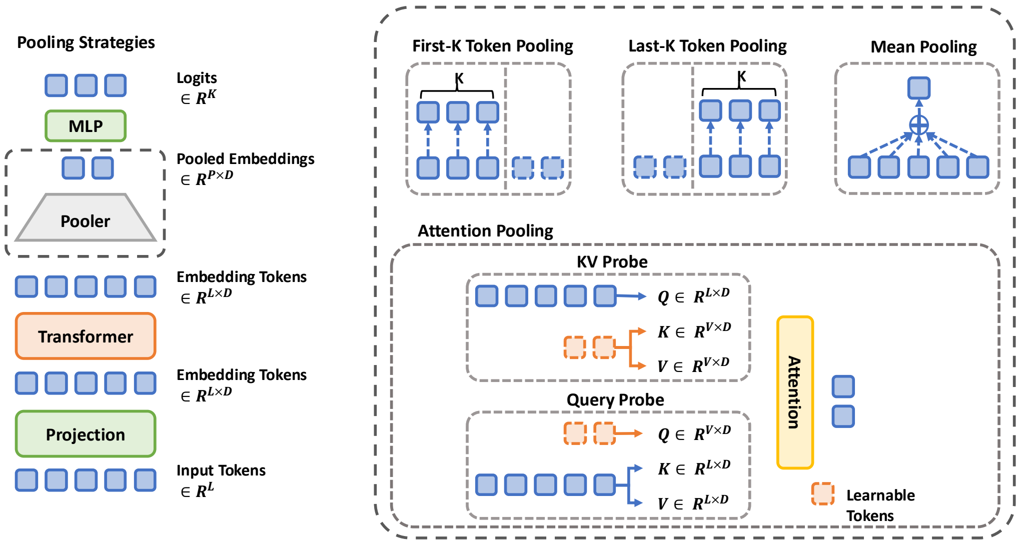

Figure 1: Gemma Encoder Architecture & Pooling: The architecture (left) comprises an encoder transformer initialized from Gemma, followed by task-specific pooler and MLP layers. The right panel illustrates the various pooling strategies considered: **First-K**, **Last-K**, **Mean**, and **Attention Pooling** with *KV*- and *Query-probe* variants).

While both decoder-only and encoder-only Transformers share a similar underlying layer design, their training objectives and downstream applications diverge significantly. Decoder-only models are optimized for next-token prediction, making them well-suited for generative tasks. In contrast, encoder-only models are trained to generate a comprehensive representation of the entire input sequence, empowering a diverse range of predictive tasks, including classification (label prediction), regression (score prediction), and ranking (relative ordering prediction).

By systematically evaluating these crucial modeling choices, we aim to identify the architecture that best captures the essential information embedded within the input sequence. This will ultimately optimize the performance of Gemma Encoder on downstream tasks. Our analysis provides valuable insights into the trade-offs inherent in various design decisions, offering guidance not only for adapting Gemma, but also for transforming other decoder-only models into effective encoders.

2.1 Attention Masking

Pre-trained generative models often employ three types of attention mask patterns: bidirectional, causal, and prefix masking (Figures 3 and 4 in Raffel et al. (2020)). For our focus on non-generative tasks, we limited the explorations to causal attention and bidirectional attention.

Bidirectional masking, also referred to as fully-visible masking (Raffel et al., 2020), is commonly used in encoder models. It allows the encoder to generate a holistic representation of the input by providing complete access to all input tokens, fostering a comprehensive understanding of the entire sequence.

Causal masking, on the other hand, is prevalent in decoder-only and sequence-to-sequence models. Here, tokens are processed sequentially, and predictions for the next token rely solely on preceding tokens. This prevents the model from "looking ahead" during training, preserving the auto-regressive property essential for text generation. The attention mechanism is masked so that each token attends only to itself and prior tokens.

T5 introduced PrefixLM, a hybrid approach that utilizes causal masking with a designated "prefix" section. This prefix is processed bidirectionally, allowing the model to attend to all tokens within it. The remaining sequence is processed causally, enabling generation conditioned on the fully contextualized prefix. This combines the benefits of bidirectional context for understanding the initial input segment with the autoregressive capabilities of causal masking for generating the subsequent sequence.

Given that Gemma models are pre-trained with causal attention, we investigated the impact of both bidirectional and causal attention masks *dur-*

ing fine-tuning to maximize the performance of Gemma Encoder models.

2.2 Pooling Strategies

The Gemma Encoder model aims to create a robust representation by effectively leveraging all input information. It comprises an encoder transformer initialized from the Gemma decoder transformer, and a pooler coupled with Multi-Layer Perceptron (MLP) layers. The pooling and MLP layers are randomly initialized. A crucial component for aggregating the contextualized token representations from the transformer into a fixed-length vector, we explored several pooling options to determine the optimal architecture.

First-K and Last-K token poolings are the two simplest pooling strategies used in transformer models to extract a fixed-size representation from a sequence of tokens for encoder tasks. In *First-K token pooling*, the representation of the first K tokens is used to aggregate information from the entire input sequence through attention mechanisms, with K being the number of classes for classification or 1 for regression and ranking. This approach can be viewed as fine-tuning the first K tokens as special tokens, analogous to the `[CLS]` token in BERT. In contrast, *Last-K token pooling* takes the representation of the last token in the sequence. This approach is particularly relevant in scenarios like language modeling or when the final tokens represents a meaningful conclusion of the sequence, such as an `[EOS]` token. While *both* approaches simplify the transformation of variable-length sequences into fixed-size representations, they may capture slightly different aspects of the input depending on the token's role in the model architecture and the training objective.

Mean pooling is another parameter-free pooling method averages the hidden states across all tokens, creating a single vector representing the average contextualized information. Mean pooling is computationally efficient and provides a basic representation of the entire input. However, it can be susceptible to noise and may dilute the importance of critical tokens by treating all tokens equally.

Attention pooling is a more sophisticated technique that employs an attention mechanism to weight and aggregate token representations. By learning attention weights, the model can focus on the most informative tokens for the downstream task, effectively filtering noise and highlighting relevant information. Attention pooling offers greater flexibility and expressiveness compared to mean pooling.

There are *two main* approaches to implementing attention pooling. The attention mechanism can be expressed as:

$$\text{Output} = \text{Softmax}(\frac{Q \cdot K^{\text{T}}}{\sqrt{D}}) \cdot V,$$

where the input is passed through the query matrix, $Q \in \mathbb{R}^{L \times D}$, and the latent variables, $K, V \in \mathbb{R}^{V \times D}$, are learned. Here, L, D, and V represent the input sequence length, the internal embedding dimension, and the number of latent variables, respectively. This approach is represented as **KV-probe** in Figure 1.

Alternatively, the input can be passed through the key and value matrices, $K, V \in \mathbb{R}^{L \times D}$, while the latent variables are learned through the query matrix, $Q \in \mathbb{R}^{V \times D}$, as introduced in Jaegle et al. (2022). This approach is represented as **Query-probe** in Figure 1.

2.3 Dropout

Dropout is a crucial regularization technique widely used in deep learning to prevent over-fitting and improve generalization by randomly deactivating a fraction of neurons during training. However, overfitting is less of a concern for the latest large language models (LLMs), due to the massive training datasets and high model capacity, which naturally provide robust generalization. Instead of dropout, LLMs (Chowdhery et al., 2022; Anil et al., 2023; Hoffmann et al., 2022) rely on other forms of regularization, such as weight decay or careful scaling strategies, which are better suited to the massive scale of these models and training corpora.

When adapting a decoder-only model to an encoder-only architecture, evaluating the role of dropout through ablation studies is crucial. Encoder-only tasks, such as regression, classification, and ranking, are more prone to overfitting compared to generative tasks handled by decoder-only models. This increased susceptibility arises because encoder-only models are typically trained on limited data using supervised fine-tuning, with less diversity in the output space. Moreover, the supervision signal in encoder-only tasks operates at the sentence or full-input-sequence level, relying on deterministic loss functions. This setup

Benchmark	Dataset	Metric	#Train	#Eval	Task
GLUE	STSB	Spearman Coeff.	5749	1500	Similarity
	CoLA	Matthews Coeff.	8551	1043	Acceptability
	QQP	F1	363846	40430	Paraphrase
	QNLI	Accuracy	104743	5463	QA / NLI
	SST2	Accuracy	67349	872	Sentiment
	RTE	Accuracy	2490	277	NLI
	MRPC	F1	3668	408	Paraphrase
	MNLI-matched	Accuracy	392702	9815	NLI
	MNLI-mismatched	Accuracy	392702	9832	NLI
SuperGLUE	BoolQ	Accuracy	9427	3270	QA
	RTE	Accuracy	2490	277	NLI
	COPA	Accuracy	400	100	QA
	CB	Accuracy	250	56	NLI
	WIC	Accuracy	5428	638	WSD
	WSC	Accuracy	554	104	Resolution
	MULTIRC	F1	27243	4848	QA
Ranking	MS-MARCO	MRR/NDCG	532751	6980	Ranking

Table 1: Dataset Statistics across GLUE, SuperGLUE and Ranking Benchmarks used in the paper. The benchmarks cover a diverse range of encoder tasks, including sentence similarity (`similarity`), `acceptability`, `paraphrase`, question answering (`QA`), natural language inference (`NLI`), sentiment classification (`sentiment`), word sense disambiguation (`WSD`), coreference resolution (`resolution`), and document ranking (`ranking`.)

makes the model more likely to fit spurious features in the input. In contrast, autoregressive tasks in decoder-only models inherently involve uncertainty and softer probability distributions, which naturally help mitigate overfitting.

2.4 Padding Strategies

When processing sequences of varying lengths in batched inputs, padding is essential to create uniform input tensors. However, the choice between **left padding** (prepending padding tokens) and **right padding** (appending padding tokens) may impacts model behavior, especially when adapting encoder model from decoder-only models.

In decoder-only transformer models, such as those used for language generation (e.g., GPT (OpenAI, 2023), Gemma (Gemma Team, 2024a,b) models), left padding is typically used to align the inputs during training or inference, due to the autoregressive training objective and efficient positional embedding. Decoder-only models are trained in an auto-regressive manner, where the task is to predict the next token based on all previous tokens. Decoder-only transformer models use positional embeddings to encode the order of tokens. Left-padding ensures that the relative positions of the actual tokens remain consistent regardless of the sequence length.

However, right padding is acceptable for encoder models because of the way these models process input sequences. Unlike decoder-only models, encoders like those in BERT (Devlin et al., 2019), T5 (Raffel et al., 2020) or other bidirectional transformer models handle the entire input sequence at once, and their attention mechanisms allow tokens to attend to any other token in the input through bidirectional attention. In encoder models, positional embeddings are applied to the entire sequence, including padding tokens. Since the padding tokens are ignored during attention, their positional embeddings don't interfere with the actual computation. Whether the padding is on the right or left does not affect the functionality.

The choice of padding strategy has implications for the pooling layer, especially in conjunction with causal attention and First-K/Last-K token pooling.

2.5 Adaption for Ranking Tasks

A ranking task concerns the relative ordering of a set of text documents by their relevance to a given query. Formally, for each query q_i, we are provided with a list of candidate documents $D_i = (d_{i1}, ..., d_{im})$ along with their relevance labels $y_i = (y_{i1}, ..., y_{im})$. The objective is to train a ranking model f, such that the model takes a query-document pair, (q_i, d_{ij}), as input and outputs a ranking score $\hat{y}_{ij} = f(q_i, d_{ij}) \in R$.

Listwise Inputs and Outputs. Adapting the Gemma Encoder to ranking tasks requires handling **listwise** inputs, which have the shape $[B, M, L]$, where B represents the input batch size, M the number of documents per query, and L the sequence length. To process these inputs, we first flat-

Model	Size	STSB	CoLA	QQP	QNLI	SST2	RTE	MRPC	MNLI		Avg
									m	mm	
BERT-base	108M	84.6	56.9	87.8	90.4	92.2	70.0	90.6	83.1	83.7	82.1
T5-base	110M	85.2	53.4	89.5	93.2	94.7	69.7	91.6	88.8	88.4	83.8
BERT-large	334M	86.4	63.9	88.4	93.0	94.0	75.5	92.4	87.1	86.9	85.3
T5-large	350M	88.0	63.6	89.6	94.8	96.9	85.6	93.5	91.0	90.9	88.2
T5-xl	1.5B	87.4	70.9	90.3	96.2	97.0	92.1	93.5	92.1	91.7	90.1
T5-xxl	6.5B	86.9	**72.9**	**90.4**	**96.4**	**97.2**	92.8	**94.2**	92.1	92.0	90.5
Gemma-2 2B	2B	92.4	67.7	89.2	95.4	97.0	87.0	91.6	91.1	91.1	89.2
Gemma-2 9B	9B	**92.6**	71.3	90.1	**96.4**	96.7	**93.1**	93.9	**92.2**	**92.1**	**90.9**

Table 2: Evaluation on GLUE benchmarks. For MNLI task, **m** and **mm** refers to `matched` and `mismatched` accuracy. All the Gemma experiments are done with bidirectional attention masking, dropout on attention softmax and feedforward network outputs with 10% rate, and right padding. The metrics for Gemma 2B and 9B are based on the best pooling strategies as ablated in Table 5.

Model	Size	BoolQ	RTE	COPA	CB	WIC	WSC	MULTIRC	Avg
T5-xxl	6.5B	90.8	**93.9**	**98.0**	96.4	**77.3**	**96.2**	87.4	**91.4**
Gemma-2 2B	2B	88.8	88.8	90.0	**100.0**	75.1	73.1	85.6	85.9
Gemma-2 9B	9B	**91.3**	93.1	96.0	**100.0**	76.5	90.4	**88.2**	90.8

Table 3: Evaluation on SuperGLUE benchmarks. The same hyper-parameter settings are applied as in Table 2.

ten the listwise structure to a shape of $[B \times M, L]$. This flattened input is then passed through the Gemma Encoder, resulting in logits with a shape of $[B \times M]$. Finally, these logits are reshaped back to $[B, M]$ to align with the expected input format of the ranking loss function.

Ranking Losses. This adaptation enables the model to produce a ranking score for each document. Consequently, any established ranking loss functions can be applied, such as pairwise logistic loss (Burges et al., 2005), PolyLoss (Leng et al., 2022), and Gumbel approximated NDCG loss (Bruch et al., 2020). In this study, we focused on the *listwise softmax cross-entropy* loss (Bruch et al., 2019; Liu, 2011).

$$\ell_{\text{Softmax}}(\mathbf{y}_i, \hat{\mathbf{y}}_i) = -\sum_{j=1}^{m} y_{ij} \log \left(\frac{e^{\hat{y}_{ij}}}{\sum_{j'} e^{\hat{y}_{ij'}}} \right).$$

3 Experiments

We evaluate the effectiveness of various modeling choices for adapting the decoder-only generative model, Gemma, to encoder-only tasks. Performance is assessed across a diverse set of classification, regression, and ranking tasks.

3.1 Data

Specifically, we demonstrate the effectiveness of Gemma Encoder for classification and scoring tasks on the GLUE (Wang et al., 2019) and SuperGLUE (Wang et al., 2020) benchmarks. We further evaluate its ranking capability on the MSMARCO benchmark (Bajaj et al., 2018).

Gemma Encoder's general language understanding abilities were evaluated on the GLUE and SuperGLUE benchmarks, which include diverse text classification and scoring tasks. The tasks and their corresponding evaluation metrics are summarized in Table 1. Our evaluation involved finetuning Gemma Encoder on each task's training set and subsequently assessing its performance on the evaluation set. The RECORD task from SuperGLUE was excluded due to its incompatibility with an encoder-only architecture.

3.2 Model

We evaluated the encoder adaptation using the Gemma-2 (Gemma Team, 2024a), specifically, the 2B and 9B decoder model variants.

3.3 Evaluation

GLUE and SuperGLUE evaluation We evaluate Gemma Encoder performance on GLUE, in Table 2, and SuperGLUE, in Table 3, benchmarks and compare it with baseline T5 and BERT models. Notably, Gemma Encoder achieves competitive performance against similarly sized T5 models, despite not utilizing any uptraining or encoder-decoder masked language modeling (MLM) pretraining. This highlights the efficacy of our architecture adaptation approach in unlocking the potential of decoder-based language models for encoder-based tasks.

The results presented in Tables 2 and 3 reflect the optimal hyperparameter configuration identified through ablation studies (Section 3.4). Specifically, we found that bidirectional attention masking (Section 3.4.2), right padding (Section 3.4.4), and a 10% dropout rate applied to the attention softmax and feedforward network outputs (Section 3.4.3) yielded the best performance.

Using the described training configurations, *especially* with bidirectional attention, no single pooling strategy consistently optimized performance for both Gemma 2B and 9B. The choice of pooling strategy had negligible impact when using our finetuning dataset. However, *with* causal attention masking (Gemma's default decoder setting), last-token pooling significantly outperformed other strategies during finetuning. Please refer to 3.4.1 for more details.

Model	Size	MRR@10	NDCG@10
RankT5-XL	1.5B	0.4358	0.5035
Gemma-2 2B	2B	**0.4456**	0.5133
Gemma-2 9B	9B	0.4450	**0.5148**

Table 4: Evaluation on MS MARCO benchmark for Ranking tasks. MRR and NDCG stand for Mean Reciprocal Rank and Normalized Discounted Cumulative Gain, the standard ranking metrics.

Model	First-K	Mean	Attention		Last-K
			Q	KV	
2B	88.7	**89.4**	89.0	89.0	89.1
9B	90.4	89.7	90.7	90.6	**90.9**

Table 5: Pooling strategy ablation. Q and KV denote Q-probe and KV-probe Attention Pooling, respectively. **Bidirectional** attention is used by default.

Model	BiDi	Causal		
		Mean	Attent	Last-K
2B	**89.4**	84.5	86.0	88.6
9B	**90.9**	87.5	88.4	90.4

Table 6: Attention mechanism Ablation. BiDi and Causal stand for bidirectional and causal attention, respectively. The BiDi results are the best results from Table 5. For causal masking results, we applied mean, attention and last token poolings for comparison. Details in Sec. 3.4.2.

MS MARCO Ranking Evaluation For the ranking task, we utilized the MS MARCO dataset, which contains approximately $530K$ queries in the train partition and $7K$ queries in the dev partition. The candidate passages are drawn from a corpus of over $8.8M$ passages. Each query is associated with relevant passages labeled with a relevance score of 1, and irrelevant passages labeled with a score of 0. Following the setup in RankT5 (Zhuang et al., 2022), our evaluation focuses on the top 1000 retrieved documents using MRR@10 and NDCG@10 as metrics, while our training uses a sample of 36 documents (1 positive plus sampled 35 negatives) per query.

Model	Bidirectional Attn.		Causal Attn.	
	left	right	left	right
2B	88.9	**89.4**	88.4	88.6
9B	90.8	**90.9**	90.6	90.4

Table 7: Padding strategies ablation. No significant difference is found for comparing two choices of padding strategies after finetuning.

We compared the finetuned performance of Gemma Encoder against the RankT5 model (Zhuang et al., 2022). As shown in Table 4, Gemma 2B and 9B variants outperform RankT5, despite not leveraging any up-training or encoder-decoder MLM-style pretraining.

3.4 Ablation

To optimize performance, as demonstrated in Tables 2 and 3, we conducted ablation studies on architectural and hyperparameter choices.

3.4.1 Pooling Strategy

To evaluate the influence of pooling strategy on the performance of the Gemma Encoder, we conducted experiments using the GLUE benchmark. Following the approach in T5 (Raffel et al., 2020), which demonstrated strong performance for encoder-based tasks, we utilized bidirectional attention masking during both finetuning and inference.

Table 5 summarizes the GLUE average scores (refer to Table 8 for per-task results) obtained with different pooling strategies for the Gemma-2 2B and 9B models (Gemma Team, 2024a). Our analysis reveals that simple pooling strategies, specifically last-token and mean pooling, achieve superior performance compared to attention pooling. But with bidirectional attention, no single pooling strategy consistently optimized performance for both Gemma 2B and 9B.

Note on limitation: It is important to acknowledge that the observed superiority of simple pooling strategies is specific to the GLUE benchmark finetuning context, which is characterized by relatively limited training data. In scenarios involving large-dataset finetuning, or pre-finetuning a decoder, especially for retrieval tasks (e.g. Lee et al. (2024) for pretrained retrieval model, Moiseev et al. (2023); Dong et al. (2022) for finetuning), a reevaluation of the relative efficacy of attention pooling and simple pooling methods is warranted. The increased parameter count associated with attention pooling is a significant factor to consider in such contexts. Please refer to Section A.4 for details.

3.4.2 Attention Masking

To investigate the impact of attention masking strategies, we evaluated both causal and bidirectional masking using the GLUE benchmark. Table 6 provides a comparison of the GLUE average scores for Gemma-2 2B and 9B models under both masking conditions. For the bidirectional masking results, we report the optimal performance achieved across different pooling strategies, consistent with the methodology in Table 5.

A key observation from our experiments is the superior performance of bidirectional masking compared to causal masking. This pattern is consistently observed across the majority of tasks within the GLUE benchmark, detailed in Section A.2 for different model sizes. This finding is particularly noteworthy given that the Gemma decoder's pretraining process utilizes solely causal masking. It suggests that employing bidirectional masking, at finetuning time, can still enhance performance on encoder-related tasks, even with a causally pretrained decoder and limited finetuning data.

Furthermore, our results demonstrate that when causal masking is employed, last-token pooling exhibits significantly better performance than alternative pooling methods. This observation aligns with the inherent nature of causal masking, where the last token possesses a comprehensive contextual understanding due to its attention over the entire preceding sequence. This is analogous to the pretraining objective of decoder-based language models, where the embedding of the last token serves as the basis for predicting the subsequent token.

3.4.3 Dropouts

Considering the task-specific nature and deterministic training objectives commonly associated with encoder tasks, we investigated the influence of dropout regularization on the performance of the Gemma Encoder using the GLUE benchmark. Figure 2 presents the GLUE average scores for the Gemma-2 2B and 9B models as a function of increasing dropout rate, starting from a rate of zero.

Our analysis reveals that the application of dropout generally enhances the performance of the Gemma Encoder for both the 2B and 9B models. However, a performance degradation is observed when the dropout rate exceeds 0.1. This pattern is consistently observed across the majority of tasks within the GLUE benchmark, detailed in Section A.3 for both model sizes. Therefore, based on these findings, we recommend a dropout rate of 0.1 applied to both the attention softmax and feedforward network outputs during the finetuning process for encoder-based tasks.

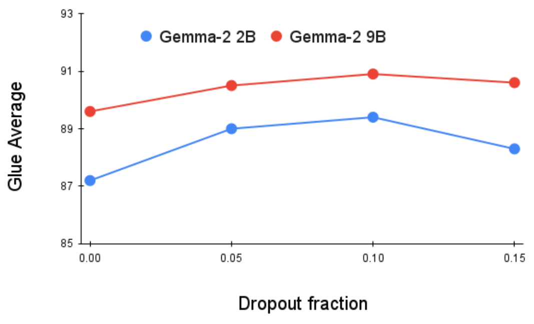

Figure 2: Effect of feed-forward and attention dropout.

3.4.4 Padding Strategy

To evaluate the influence of padding side on the performance of the Gemma Encoder, we compared left and right padding strategies. Table 7 summarizes the GLUE average scores obtained for the Gemma-2 2B and 9B models under both padding conditions. Consistent with our expectations, no statistically significant difference in performance was observed between the two padding approaches. This indicates that the model exhibits robustness to padding side after finetuning.

4 Conclusion

We introduce Gemma Encoder, adapting the decoder-only Gemma model for encoder tasks. We addressed using decoder models' pre-trained knowledge for tasks typically handled by encoders (e.g., BERT, T5's encoder). Our approach involved task-specific pooling/MLP layers, bidirectional attention, and dropout during finetuning. We also analyzed padding strategies.

Gemma Encoder outperformed baselines on GLUE, SuperGLUE, and MSMARCO benchmarks (classification, regression, ranking), proving that well-adapted decoder-only models excel at encoder tasks. Bidirectional attention proved crucial for capturing context, and dropout improved robustness and generalization. Adaptable pooling and output layers further enhanced performance.

5 Limitations

We identify a couple of limitations. First, our experiments are based on the Gemma models. We need to evaluate how the findings would translate to other large language models. Second, we only evaluate the quality of the encoder using classification, scoring and ranking tasks. Another common use case of Encoders, retrieval (or embedding) models are not considered in this work.

References

Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, and 109 others. 2023. Palm 2 technical report. *Preprint*, arXiv:2305.10403.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. Ms marco: A human generated machine reading comprehension dataset. *Preprint*, arXiv:1611.09268.

Sebastian Bruch, Shuguang Han, Mike Bendersky, and Marc Najork. 2020. A stochastic treatment of learning to rank scoring functions. In *Proceedings of the 13th ACM International Conference on Web Search and Data Mining (WSDM 2020)*, pages 61–69.

Sebastian Bruch, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2019. An analysis of the softmax cross entropy loss for learning-to-rank with binary relevance. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '19, page 75–78.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, and 48 others. 2022. Palm: Scaling language modeling with pathways. *Preprint*, arXiv:2204.02311.

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.

Zhe Dong, Jianmo Ni, Daniel M. Bikel, Enrique Alfonseca, Yuan Wang, Chen Qu, and Imed Zitouni. 2022. Exploring dual encoder architectures for question answering. *Preprint*, arXiv:2204.07120.

Gemini Team. 2023. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

Gemma Team. 2024a. Gemma 2: Improving open language models at a practical size. *Preprint*, arXiv:2408.00118.

Gemma Team. 2024b. Gemma: Open models based on gemini research and technology. *Preprint*, arXiv:2403.08295.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, and 3 others. 2022. Training compute-optimal large language models. *Preprint*, arXiv:2203.15556.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier J Henaff, Matthew Botvinick, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. 2022. Perceiver IO: A general architecture for structured inputs & outputs. In *International Conference on Learning Representations*.

Jinhyuk Lee, Zhuyun Dai, Xiaoqi Ren, Blair Chen, Daniel Cer, Jeremy R. Cole, Kai Hui, Michael Boratko, Rajvi Kapadia, Wen Ding, Yi Luan, Sai Meher Karthik Duddu, Gustavo Hernandez Abrego, Weiqiang Shi, Nithi Gupta, Aditya Kusupati, Prateek Jain, Siddhartha Reddy Jonnalagadda, Ming-Wei Chang, and Iftekhar Naim. 2024. Gecko: Versatile text embeddings distilled from large language models. *Preprint*, arXiv:2403.20327.

Zhaoqi Leng, Mingxing Tan, Chenxi Liu, Ekin Dogus Cubuk, Jay Shi, Shuyang Cheng, and Dragomir Anguelov. 2022. Polyloss: A polynomial expansion perspective of classification loss functions. In *International Conference on Learning Representations*.

Tie-Yan Liu. 2011. *Learning to Rank for Information Retrieval*. Springer.

Fedor Moiseev, Gustavo Hernandez Abrego, Peter Dornbach, Imed Zitouni, Enrique Alfonseca, and Zhe Dong. 2023. Samtone: Improving contrastive loss for dual encoder retrieval models with same tower negatives. *Preprint*, arXiv:2306.02516.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2020. Superglue: A stickier benchmark for general-purpose language understanding systems. *Preprint*, arXiv:1905.00537.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding. *Preprint*, arXiv:1804.07461.

Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. 2022. Rankt5: Fine-tuning t5 for text ranking with ranking losses. *Preprint*, arXiv:2210.10634.

A Additional Results

A.1 Pooling Ablation Details

Table 8 provides a breakdown of per-task performance on the GLUE benchmark for different pooling strategies. The query-probe approach is utilized for attention pooling. The results indicate that simple pooling strategies achieve performance competitively to that of attention pooling, even though attention pooling introduces a greater number of parameters.

A.2 Attention Ablation Details

Table 9 provides a breakdown of per-task performance on the GLUE benchmark for different attention mechanisms.

A.3 Dropout Ablation Details

Table 10 provides a breakdown of per-task performance on the GLUE benchmark across a range of dropout rates.

A.4 Attention Pooling Ablation

Table 11 provides a comparison of per-task performance on the GLUE benchmark for various attention pooling variants. An increase in attention complexity, reflected in a larger number of parameters, resulted in a decrease in performance on the GLUE benchmarks. As detailed in Table 1, the GLUE training sets are limited in size (all containing fewer than 1 million examples), which is inadequate for the effective adaptation of a randomly initialized attention pooler with millions of parameters.

Model	Pooling	STSB	CoLA	QQP	QNLI	SST2	RTE	MRPC	MNLI		Avg
									m	mm	
Gemma 2B	First-K	92.3	65.5	88.3	95.2	96.7	87.0	92.2	90.6	90.9	88.7
	Mean	92.4	67.7	89.2	95.4	97.0	87.3	93.3	91.1	91.1	89.4
	Attention	92.3	67.1	88.7	95.3	96.7	87.3	92.0	90.7	90.9	89.0
	Last token	92.4	65.7	88.1	95.2	97.4	88.8	92.7	90.8	90.8	89.1
Gemma 9B	First-K	92.1	72.3	89.2	96.1	96.3	90.6	93.0	92.4	92.0	90.4
	Mean	91.5	69.5	90.3	96.4	96.5	88.8	89.4	92.4	92.2	89.7
	Attention	92.5	72.1	90.1	96.5	96.5	91.3	93.1	92.5	92.1	90.7
	Last token	92.4	70.9	89.8	96.3	96.7	94.2	93.2	92.3	92.1	90.9

Table 8: Impact of pooling on GLUE task performance. Experiments used bidirectional attention, 10% dropout, and right padding. Attention pooling utilizes a Query probe.

Model	Attention	STSB	CoLA	QQP	QNLI	SST2	RTE	MRPC	MNLI		Avg
									m	mm	
Gemma 2B	Causal (Mean)	90.7	65.2	87.7	94.4	96.2	61.0	85.4	89.8	90.1	84.5
	Causal (Attent.)	91.0	68.3	87.9	94.7	95.8	69.3	86.1	90.5	90.7	86.0
	Causal (Last-K)	92.1	67.3	88.1	94.9	96.3	86.6	90.6	90.6	90.7	88.6
	BiDi	92.4	67.7	89.2	95.4	97.0	87.3	93.3	91.1	91.1	89.4
Gemma 9B	Causal (Mean)	91.7	72.5	89.4	95.9	96.3	73.6	85.0	91.8	91.7	87.5
	Causal (Attent.)	92.1	71.9	89.6	96.1	96.5	76.8	88.3	92.0	91.9	88.4
	Causal (Last-K)	92.4	72.2	89.4	95.9	96.5	91.7	91.8	91.9	91.8	90.4
	BiDi	92.4	70.9	89.8	96.3	96.7	94.2	93.2	92.3	92.1	90.9

Table 9: Impact of attention mechanism on GLUE tasks: `BiDi` (bidirectional) uses optimal pooling; `Causal` uses mean, attention, and last-token pooling. All experiments use 10% dropout and right padding. Attention pooling utilizes a Query probe.

Model	Dropout	STSB	CoLA	QQP	QNLI	SST2	RTE	MRPC	MNLI		Avg
									m	mm	
Gemma 2B	0	91.0	62.8	89.2	95.2	96.2	78.3	89.0	91.3	91.5	87.2
	0.05	92.0	66.4	88.9	95.3	96.4	87.3	92.4	91.0	91.2	89.0
	0.10	92.4	67.7	89.2	95.4	97.0	87.3	93.3	91.1	91.1	89.4
	0.15	92.0	66.5	88.3	95.0	96.7	84.4	91.0	90.5	90.7	88.3
Gemma 9B	0	91.3	68.5	90.2	96.4	96.6	88.8	90.0	92.4	92.2	89.6
	0.05	92.2	71.0	90.0	96.3	96.4	92.4	92.0	92.3	92.0	90.5
	0.10	92.4	70.9	89.8	96.3	96.7	94.2	93.2	92.3	92.1	90.9
	0.15	92.3	71.4	89.5	96.2	96.6	92.4	93.3	92.1	92.0	90.6

Table 10: Impact of dropout on GLUE task performance. Experiments used bidirectional attention and right padding. Gemma 2B employed mean pooling, while Gemma 9B used last-token pooling.

Probe	H	T	#params	STSB	CoLA	QQP	QNLI	SST2	RTE	MRPC	MNLI		Avg
											m	mm	
Q	1	1	2.3M	92.3	67.1	88.7	95.3	96.7	87.3	92.0	90.7	90.9	89.0
Q	2	1	4.7M	92.2	66.4	88.6	95.1	96.6	86.3	92.1	90.6	90.6	88.7
Q	4	1	9.4M	92.2	67.4	88.8	95.1	96.7	86.6	92.2	90.6	90.6	88.9
Q	8	1	18.8M	92.2	66.4	88.6	95.4	96.8	85.6	91.9	90.7	90.7	88.7
Q	1	512	2.3M	92.0	66.1	88.7	95.3	96.8	85.9	91.7	90.6	90.6	88.6
Q	8	512	18.8M	92.2	66.1	88.7	95.3	96.8	84.1	91.9	90.5	90.5	88.5
KV	1	512	2.3M	92.3	66.6	88.5	95.1	96.9	88.4	92.2	90.6	90.6	89.0
KV	8	512	18.8M	92.3	67.9	88.6	95.1	96.7	87.0	92.2	90.7	90.7	89.0

Table 11: Gemma 2B performance on GLUE tasks was evaluated with ablations on **attention pooling probes**, varying the number of attention heads (**H**) and pooled tokens (**T**). Experiments used bidirectional attention, 10% dropout, and right padding.