# Improving Decoder-only Language Models for Sequence Labeling through Sequence Repetition

Anonymous ACL submission


## Abstract

Modern language models (LMs) are trained in an autoregressive manner, conditioned only on the prefix. In contrast, sequence labeling (SL) tasks assign labels to each individual input token, naturally benefiting from bidirectional context. This discrepancy has historically led SL to rely on inherently bidirectional encoder-only models. However, the rapid development of decoder-only models has raised the question of whether they can be adapted to SL. While causal mask removal has emerged as a viable technique for adapting decoder-only models to leverage the full context for SL, it requires considerable changes to the base model functionality. In this work, we explore sequence repetition (SR) as a less invasive alternative for enabling bidirectionality in decoder-only models. Through fine-tuning experiments, we show that SR inherently makes decoders bidirectional, improving the quality of token-level embeddings and surpassing encoders and unmasked decoders. Contrary to earlier claims, we find that increasing the number of repetitions does not degrade SL performance. Finally, we demonstrate that embeddings from intermediate layers are highly effective for SR, comparable to those from final layers, while being at least 1.33 times more efficient to compute. Our findings underscore that SR alleviates the structural limitations of decoders, enabling more efficient and adaptable LMs and broadening their applicability to other token-level tasks.

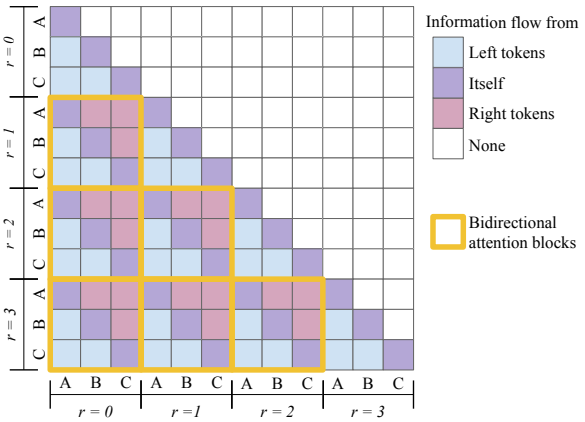




Figure 1: Attention weight matrix in a decoder block for an input sequence "ABC" repeated $r$ times. Sequence repetition (SR) forms bidirectional attention block regions with no modifications to the naturally unidirectional architecture of decoder-only language models. The share of bidirectional blocks contributing to the attention outputs increases with subsequent repetitions, enabling the model with a higher capacity to process tokens to the right of the currently processed token.

## 1 Introduction

The NLP community has invested immense effort in training models for natural language understanding tasks (NLU). Arguably, one of the more complex NLU tasks is sequence labeling (SL), which involves assigning labels to individual tokens within a sequence. From text chunking and named entity recognition (Tjong Kim Sang and Buchholz, 2000; Tjong Kim Sang and De Meulder, 2003) to more challenging tasks such as event extraction (Doddington et al., 2004) and aspect-based sentiment analysis (Pontiki et al., 2014), SL models rely on high-quality token embeddings.

While encoder-only LMs like BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) have long been the standard for bidirectional SL tasks, recent scaling has enabled decoder-only LMs to produce superior embeddings, surpassing their encoder counterparts on various token- and sequence-level tasks (BehnamGhader et al., 2024; Lin et al., 2025a). This trend has motivated research into repurposing decoders as bidirectional encoders, typically by removing the causal mask (CM) in combination with parameter-efficient fine-tuning (Houlsby et al., 2019; Dukić and Šnajder, 2024). However, while this modification provides the model with a fully bidirectional context, it is a substantial architectural change that risks impairing ac-

cess to its pre-trained knowledge (Lin et al., 2025a). A parallel line of work has shown that simply repeating the input sequence can improve sequence-level representations in decoders (Springer et al., 2025; Duan et al., 2025; Xu et al., 2024). Sequence repetition (SR) offers a less invasive method for incorporating bidirectionality without modifying the decoder's core architecture, as illustrated in Figure 1. By repeating the input sequence several times, the decoder blocks form bidirectional regions inside attention matrices. However, despite its promise for sequence-level tasks, the effect of SR on token-level embeddings—and, by extension, on SL performance in a decoder-as-encoder setup—remains largely unexplored.

Aiming to fill this gap, we explore the capacity of SR as a tool for token-level representation learning. Through controlled fine-tuning experiments, we demonstrate that SR implicitly enables bidirectionality within decoder-only models, yielding token-level embeddings that outperform both strong encoders and unmasked decoders across multiple SL benchmarks. In a follow-up analysis, we show that, contrary to previous findings, repeating the input more than once often results in performance gains rather than degradation (Springer et al., 2025).[1] To mitigate the computational cost of sequence repetition, we draw inspiration from early exit techniques (Xin et al., 2020; Zhou et al., 2020; Rep et al., 2024) and extract embeddings from intermediate decoder layers. We find that fine-tuning the model only up to a selected layer yields representations that perform on par with those from the final layer, while being at least 1.33× more efficient to compute. These results reinforce the findings that intermediate LM layers are often critical for downstream tasks (Skean et al., 2025; Fan et al., 2024).

The contributions of our work are threefold:

- We find SR is a simple, practical, and effective strategy for SL with decoder-only models, outperforming more complex alternatives;
- We show that, contrary to prior findings, performance improves as the number of sequence repetitions increases;
- We demonstrate that early exiting mitigates the computational overhead of SR while maintaining performance competitive with last-layer embeddings.

[1]This performance degradation was reported in experiments not included in the main paper https://openreview.net/forum?id=Ahlrf2HGJR&noteId=zYNHEaQiWO.

## 2 Sequence Repetition

Previous works have shown that repeating the input sequence once is a simple method for bypassing the CM used in decoder-only models (Springer et al., 2025; Xu et al., 2024). In this way, tokens in the repeated sequence have access to the entire sequence on the left-hand side, allowing the decoder-only model to function bidirectionally. However, while prior work has found that additional repetitions yield no further gains on sequence-level tasks (Springer et al., 2025; Xu et al., 2024; Duan et al., 2025), our findings show that the opposite is true for token-level tasks, where performance improves with each repetition (§4.2). We hypothesize that this occurs because additional repetitions increase the model's effective processing capacity for each token. We further posit that while this enriches individual token embeddings, the improvement is lost in sequence-level representations due to pooling, which averages out these marginal gains.

We now outline our intuition of how repetitions enable bidirectional information flow during the computation of attention weights. Let $s = [t_1, \ldots, t_n]$ be an input sequence of length $n$. We denote the repeated sequence as $s^k$, where $k = r + 1$ is the total number of instances and $r$ is the number of repetitions. The final sequence has a length of $k \cdot n$, as illustrated in Figure 1. Let $A$ be the attention weight matrix computed via standard scaled dot-product attention (Vaswani et al., 2017) using causal masking (Radford et al., 2018):

$$h_a(s) = A(s)V = \text{softmax}\left(\frac{QK^T + CM}{\sqrt{d}}\right)V,$$

where $Q = sW_Q$, $K = sW_k$, and $V = sW_v$ are query, key, and value matrices, respectively, $d$ is the dimensionality of query and key vectors, and $CM$ is the causal mask defined as:

$$CM = \begin{bmatrix} 0 & -\infty & \cdots & -\infty \\ 0 & 0 & \cdots & -\infty \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}.$$

Due to causal masking, $A$ is a lower-triangular $n \times n$ matrix. For the repeated sequence $s^k$, which is composed of $k$ instances of $s$, the dimensions of $A$ grow to $kn \times kn$, and it can be viewed as a block

matrix made of $k^2$ individual $n \times n$ submatrices:

$$A(s_k) = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1k} \\ A_{21} & A_{22} & \cdots & A_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ A_{k1} & A_{k2} & \cdots & A_{kk} \end{bmatrix}.$$

Each submatrix $A_{i,j}$ corresponds to the attention weights between tokens in the $i$-th and $j$-th instances of sequence $s$ within the repeated sequence $s^k$ (see Figure 1). Due to the causal mask, the structure of these submatrices is determined by their position: for $i > j$, the submatrix $A_{ij}$ is dense (full-rank); for $i = j$, $A_{ij}$ is lower-triangular; and for $i < j$, $A_{ij}$ is a zero matrix.

In practice, this means that $A_{ij}$ contributes to the attention output with full bidirectional information for $i > j$ and with left and diagonal-only information for $i = j$, while zero matrices contribute nothing to the output. Thus, in the full attention matrix $A(s^{(k)})$, there are $k(k+1)/2$ non-zero blocks that contribute any information, of which $k(k+1)/2-k$ are fully bidirectional. The proportion of bidirectional blocks in $A(s^k)$ approaches 1 as $k$ increases:

$$\lim_{k\to\infty} \frac{k(k+1)/2-k}{k(k+1)/2} = \lim_{k\to\infty} \frac{k-1}{k+1} = 1$$

A token's representation in the $k$-th instance of the sequence is computed using the attention weights in the blocks $A_{k1} \ldots A_{kk}$. Since all of these blocks except the final one ($A_{kk}$) are bidirectional, the overall attention mechanism more closely emulates full bidirectionality as $k$ increases.

Naturally, this comes at a higher computational cost due to the $k$-fold increase in sequence length. However, we hypothesize that even small values of $k$ can yield substantial performance gains by closely approximating full bidirectionality. We verify this claim experimentally in subsequent sections.

## 3 Experimental Setup

In this section, we first describe the standard SL tasks we consider (§3.1), the language models used in our experiments (§3.2), and their optimization setup (§3.3). We then outline the competitive baselines to which we compare our results (§3.4). Finally, we detail the early-exit setup used to mitigate the computational cost of sequence repetition (§3.5).

| Dataset | Train | Valid | Test | Total | #Cl. |
|---|---|---|---|---|---|
| CoNLL03 | 14,041 | 3,250 | 3,453 | 20,744 | 4 |
| ACE05 | 14,672 | 873 | 711 | 16,256 | 33 |
| Rest14 | 2,737 | 304 | 800 | 3,841 | 4 |
| NLU++ | 2,152 | 309 | 619 | 3,080 | 17 |

Table 1: Dataset statistics: the number of sentences per split, the total number of sentences, and the total number of class labels.

### 3.1 Datasets

**NER.** For NER, we use the CoNLL03 dataset (Tjong Kim Sang and De Meulder, 2003), specifically the version available in *Hugging Face Datasets* (Lhoest et al., 2021), which follows the IOB2 sequence tagging scheme and provides predefined train, validation, and test splits.

**Aspect Term Extraction and Polarity.** We use data from the restaurants domain of SemEval-2014 Task 4 (Pontiki et al., 2014) (Rest14). For our experiments, we merge the two main aspect-based sentiment analysis subtasks, aspect term extraction (ATE) and aspect term polarity (ATP), into a single sequence labeling task (ATE+ATP). We then tokenize the dataset with spaCy (Honnibal et al., 2020) and match the given character spans of aspect terms with token spans to obtain IOB2 tags. While the training and test splits are predefined, we create our validation set by following prior work (Wang et al., 2021) and randomly sampling 10% of the training data.

**Slot Labeling.** For this task, we use the NLU++ dataset (Casanueva et al., 2022) designed for NLU in task-oriented dialogue systems. The dataset is divided into two distinct domains (*banking* and *hotels*), each containing a mix of generic slots and domain-specific slots. To create a larger and more robust dataset for our experiments, we merge the two domains. Merging domains results in a unified set of 17 unique slot labels, as some generic slots overlap. We create IOB2 tags for this combined dataset by mapping the character-level offsets of slot values to tokens generated by spaCy (Honnibal et al., 2020). Although the original authors recommend k-fold cross-validation, this approach is computationally prohibitive when fine-tuning large language models. Therefore, we opt for fixed data splits, creating our training, validation, and test sets by shuffling the data and partitioning it into a 70/10/20 ratio.

**Event Trigger Classification.** The ACE05 dataset (Doddington et al., 2004) is a widely used event trigger classification (ETC) dataset. The ETC task combines two event extraction tasks into a single SL task: trigger identification, i.e., finding spans of tokens that evoke a particular event, and classifying them. We use the English train, validation, and test split obtained with the standard ACE preprocessing tool,[2] which we also use to generate tokenized sentences and create IOB2 tags.

### 3.2 Models

We conduct our experiments with seven open-weight decoder-only LMs from three different model families: Gemma-7B (Team et al., 2024a), Gemma2-2B, Gemma2-9B (Team et al., 2024b), Qwen3-1.7B, Qwen3-4B, Qwen3-8B (Yang et al., 2025), and Mistral-7B (Jiang et al., 2023). By default, we use these decoder Transformers with a softmax token classification head on top of the model's Transformer body and train the models to output IOB2 tags. We compare performance of decoder-only models to state-of-the-art encoder-only models: RoBERTa (Liu et al., 2019), and ModernBERT (Warner et al., 2025) with the same training and evaluation protocol. See Appendix A for implementation details.

### 3.3 Optimization

We apply QLoRA (Dettmers et al., 2023) to query, key, value, and output attention matrices in all decoder layers in our fine-tuning experiments (in early exit experiments with decoders we apply fine-tuning only up to a chosen intermediate layer). We use the following hyperparameters: fixed rank $r = 16$, scaling parameter $\alpha = 16$, dropout probability of $p = 0.1$ and learning rate 2e$-$4. We train the models with the AdamW (Loshchilov and Hutter, 2019) optimizer. The parameters of AdamW optimizer are left at default values when loaded from the PyTorch library $\beta_1 = 0.9, \beta_2 = 0.99, \varepsilon = 1\text{e}{-8}$, with an exception of weight decay which is set to 0.05. We employ gradient checkpointing with the reentrant implementation, using a batch size of 8 and gradient accumulation over 4 steps to fit our memory constraints, resulting in an effective batch size of 32. All models were fine-tuned over 10 epochs, and the best-performing weights were selected according to the highest micro F1 value on the validation set. For each fine-tuning experiment, we report the average micro F1 performance from five different runs (i.e., with different random seeds). We refer the reader to Appendix A.3 for optimization details.

### 3.4 Baselines

We compare the performance of SR approaches to decoder-only baselines for SL listed below.

**Masked Decoder.** We fine-tune the base decoder model to output IOB2 tags with no modifications to the input sequence or the model's architecture. Under the SR formulation, this case corresponds to no repetitions ($r = 0$).

**Unmasking.** Following (Li et al., 2023) we remove CM from the attention computation at each layer, thereby allowing the models to utilize full bidirectional information in the input sequence at each token, effectively transforming them to encoders.

**Layer-Group unmasking.** (Dukić and Šnajder, 2024) showed that, for optimal SL performance, CM should be removed only from selected layer groups, most commonly from middle layers. This is sensible, since intermediate layers have been shown to be important for improvements on downstream tasks (Skean et al., 2025). For a model with $N$ layers, we remove the CM from the middle $N/3$ layers. See Appendix A.2.1 for details on how we select the exact indices to perform unmasking.

### 3.5 Early Exit

As SR introduces additional input processing, we propose combining it with early exit for a trade-off between quality and efficiency. Following prior findings that earlier layers can yield embeddings comparable in quality to those from later layers for downstream tasks (Xin et al., 2020; Zhou et al., 2020; Rep et al., 2024), we investigate fine-tuning the model up to layer $L$ in conjunction with SR. Specifically, we consider a model with $N = 32$ layers, and fine-tune up to layer $L \in \{9, 14, 19, 24\}$. This improves computational efficiency by factors ranging from $3.56\times$ to $1.33\times$ when exiting at layers 9 and 24, respectively.

## 4 Results

In this section, we first compare SR to encoder-only models and decoder-only approaches to SL (§4.1). We then analyze SR performance as the number of repetitions is varied (§4.2). We follow up by exploring early exiting as a strategy of mitigating additional compute incurred by SR (§4.3). Finally,

[2] https://bit.ly/ace2005-preprocessing

| Model | Best #repetitions | NLU++ | ACE05 | Rest14 | CoNLL03 |
|---|---|---|---|---|---|
| Gemma-7B | r=8 | 79.34 (±1.29) | 71.62 (±3.79) | 81.79 (±1.11) | 93.25 (±0.10) |
| Gemma2-2B | r=2 | 79.98 (±0.77) | 72.86 (±2.30) | 83.03 (±0.65) | 93.05 (±0.19) |
| Gemma2-9B | r=2 | 78.83 (±1.56) | 74.49 (±1.08) | 83.51 (±0.56) | 93.58 (±0.15) |
| Mistral-7B | r=4 | **80.28** (±1.12) | **77.79 (±3.74)** | 83.31 (±0.99) | **93.79 (±0.12)** |
| Qwen3-1.7B | r=4 | 77.75 (±0.90) | 70.18 (±2.16) | 79.26 (±0.65) | 91.56 (±0.26) |
| Qwen3-4B | r=8 | 77.97 (±0.82) | 72.58 (±3.28) | 81.22 (±0.30) | 92.33 (±0.18) |
| Qwen3-8B | r=4 | 77.61 (±1.59) | 72.11 (±3.69) | 82.27 (±0.70) | 92.74 (±0.16) |
| **Middle unmasking** | | | | | |
| Gemma-7B | | 79.90 (±2.12) | 72.15 (±3.50) | **83.64** (±0.43) | 91.46 (±3.11) |
| Gemma2-2B | | 78.21 (±1.28) | 72.30 (±2.52) | 82.21 (±0.26) | 92.60 (±0.20) |
| Gemma2-9B | | 77.63 (±0.77) | 75.91 (±1.85) | 83.34 (±0.61) | 93.17 (±0.23) |
| Mistral-7B | | 79.35 (±0.33) | 77.56 (±2.09) | 83.00 (±0.56) | 93.48 (±0.08) |
| Qwen3-1.7B | | 73.67 (±0.92) | 66.71 (±2.44) | 75.71 (±0.65) | 90.31 (±0.26) |
| Qwen3-4B | | 74.56 (±0.51) | 69.73 (±1.84) | 79.60 (±0.26) | 91.87 (±0.21) |
| Qwen3-8B | | 75.06 (±0.75) | 71.45 (±1.88) | 80.17 (±0.47) | 92.23 (±0.24) |
| **Full unmasking** | | | | | |
| Gemma-7B | | 74.34 (±1.75) | 73.08 (±4.17) | 76.63 (±3.14) | 92.75 (±0.10) |
| Gemma2-2B | | 76.42 (±1.36) | 74.33 (±2.65) | 81.26 (±0.41) | 92.82 (±0.22) |
| Gemma2-9B | | 76.83 (±1.32) | 75.79 (±2.34) | 82.55 (±0.81) | 93.44 (±0.08) |
| Mistral-7B | | 80.23 (±0.78) | 76.26 (±2.38) | 83.18 (±0.53) | 93.69 (±0.15) |
| Qwen3-1.7B | | 67.28 (±2.68) | 68.68 (±2.14) | 75.32 (±1.26) | 90.27 (±0.29) |
| Qwen3-4B | | 73.79 (±0.74) | 70.59 (±2.83) | 77.76 (±0.39) | 91.17 (±0.11) |
| Qwen3-8B | | 74.45 (±1.72) | 69.01 (±1.97) | 80.01 (±0.26) | 91.99 (±0.18) |
| **Encoders** | | | | | |
| ModernBERT | | 70.50 (±2.05) | 68.30 (±2.64) | 75.51 (±0.64) | 90.02 (±0.15) |
| RoBERTa | | 72.22 (±1.06) | 68.96 (±5.08) | 80.30 (±0.50) | 92.64 (±0.08) |

Table 2: Micro F1 scores (mean ±95% CI) across Rest14, ACE05, CoNLL03, and NLU++ test sets. Bold values indicate the highest mean per dataset within each group. All results are averages of five runs. We report the results for r which overall had the best average F1 score for a given model.

we conduct a qualitative analysis of performance between adaptation strategies (§4.4).

### 4.1 Performance on Sequence Labeling

We report results of the best-performing encoder-based methods, unmasking baselines and sequence repetition on the test splits of datasets in Table 2. Generally, decoder-based models outperform encoders. Specifically, encoders lag behind unmasked decoders and even more so behind SR decoders, which outperform them by a large margin. When considering unmasking baselines, unmasking only the middle layers generally outperforms removing the CM entirely. On Rest14, Gemma-7B achieves the best results among all models with middle-layer unmasking. Other models also benefit more from partial bidirectionality than from full bidirectionality. The exception is Mistral-7B, where full unmasking surpasses middle-layer unmasking on all SL datasets except ACE05. This suggests that Mistral-7B benefits more from full bidirectionality than solely from the bidirectionality of the middle layers. Most importantly, the best overall results are achieved with SR models, where Mistral-7B stands out as the best-performing model, achieving the highest score across NLU++, ACE05, and CoNLL03 for $r = 4$. This trend is even more apparent in Figure 2, where Mistral-7B leverages additional repetitions most effectively across models and datasets on both the validation and test sets.

### 4.2 Effect of Multiple Sequence Repetitions

We now investigate how does model performance behave when varying the number of repetitions. We report the results of SR fine-tuning for $r \in \{0, 1, 2, 4, 8\}$ in Figure 2. Repeating the sequence once ($r = 1$) yields clear gains over the masked decoder ($r = 1$). This consistency is observed across all SL datasets, models, and both validation and test sets. The highest jump in performance is from $r = 0$ to $r = 1$. Increasing the number of sequence repetitions further improves the performance for

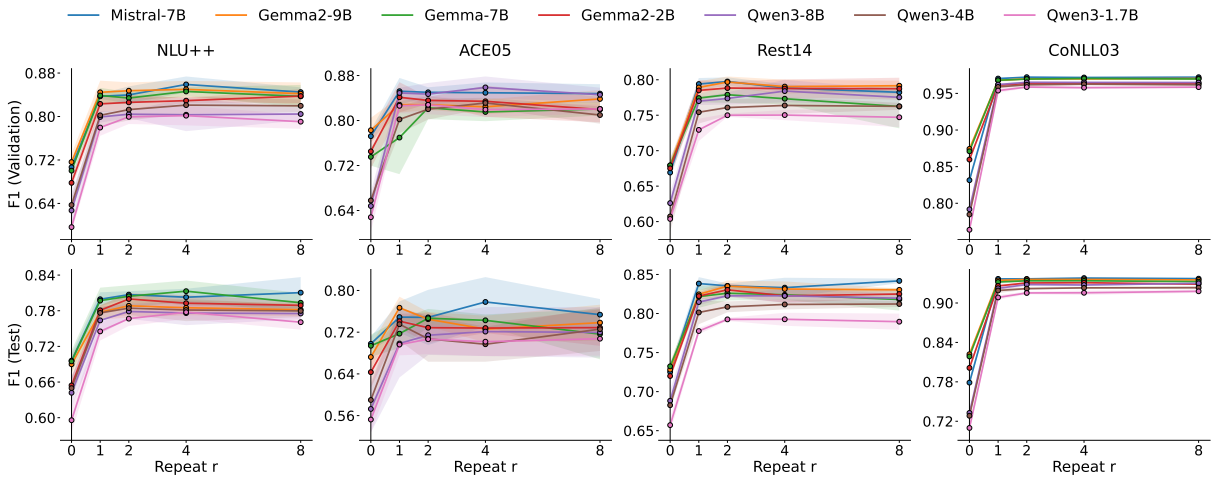



Figure 2: Micro F1 performance for sequence repetition across datasets and splits. Each subplot displays validation (top row) or test (bottom row) results for one dataset (NLU++, ACE05, Rest14, CoNLL03), with the x-axis indicating the number of repetitions and the y-axis showing the mean F1 score. Lines represent different models trained for sequence repetition, shaded bands denote ±95% CI. All results are averages of five runs.

most models and datasets. Performance gains generally saturate at $r = 2$ or $r = 4$ and remain stable with additional repetitions, contrary to prior reports of degradation on sequence-level tasks for $r > 1$ (Springer et al., 2025). Mistral-7B stands out as the best-performing model, especially as the number of repetitions approaches $r = 8$. Smaller models (Qwen3-1.7B and Gemma2-2B) exhibit a noticeable performance drop in most cases after $r = 2$ or $r = 4$, suggesting that they lack the capacity to effectively utilize additional repetitions. However, already at 4B parameters (Qwen3-4B), performance stabilizes and saturates as $r$ grows. We hypothesize that larger models, with larger effective context windows, could leverage additional repetitions to a greater extent.

### 4.3 Improving Efficiency by Early Exiting

SR yields higher-quality token-level embeddings than unmasking, leading to greater gains on SL tasks. However, this method introduces a significant computational cost, especially as the number of repetitions increases. To mitigate this overhead, we explore the use of an early-exit strategy. We hypothesize that intermediate layer embeddings can yield competitive performance compared to last layer embeddings. To verify this, we fine-tuned our best-performing decoder (Mistral-7B) with sequence repetitions $r \in \{1, 2, 4, 8\}$ up to layer $L \in \{9, 14, 19, 24\}$. Mistral-7B has 32 layers, and exiting at layer $L = 24$ results in a 25% reduction in the number of parameters. We report results on validation and test sets in Figure 3. *Earliest* layer embeddings from $L = 9$ do not perform on par with the embeddings from the last layer, especially in the case when $r = 1$. However, embeddings from layers as early as $L = 14$ can already match the performance of the last layer (see results on Rest14). Generally, the best performing embeddings originate from layers $L = 19$ and $L = 24$, and their quality improves with more repetitions. In some cases, these intermediate embeddings can even outperform those from the last layer (see Rest14 validation and NLU++ test for $L = 19$), indicating that intermediate layer embeddings might be more suitable for some downstream tasks, a finding that aligns with previous work (Skean et al., 2025).

### 4.4 Qualitative Analysis of Adaptation Strategy Performance

Figure 4 summarizes our performance comparisons, based on the full results presented in Appendix B. The figure shows both pairwise results for each strategy (Figure 4a) and the total number of wins per strategy across all (model, dataset) pairs (Figure 4b).

Sequence repetition with more than one repetition ($r > 1$) proves to be the superior strategy, winning in 24 out of 28 (model, dataset) pairs and thus demonstrating the efficacy of multiple repetitions. Although overall performance tends to saturate around $r = 4$ (cf. Figure 2), this high win rate is maintained up to $r = 8$.

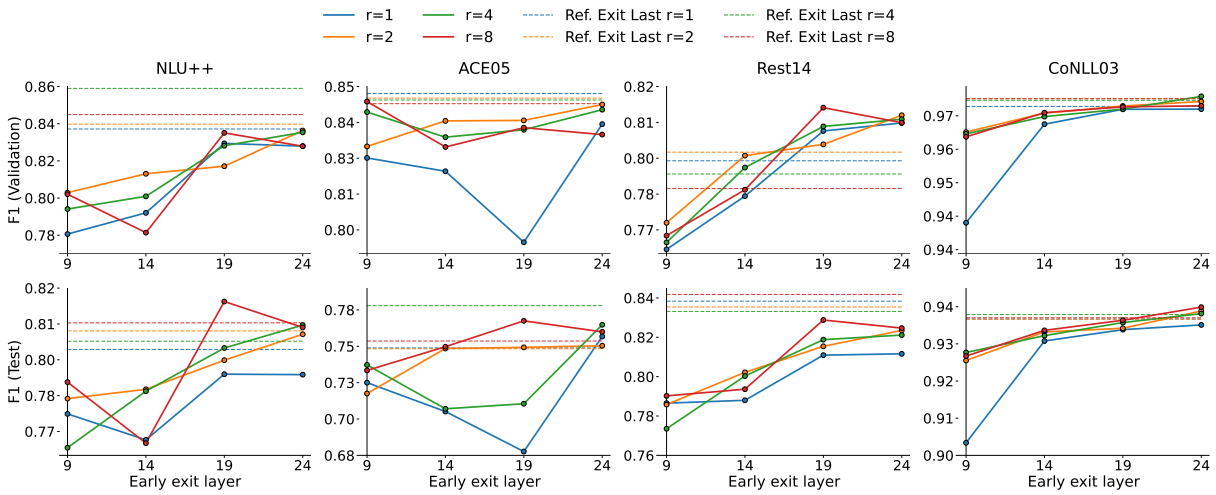



Figure 3: Micro F1 scores for early exit experiments across datasets and splits for Mistral-7B model. Each column displays a dataset (NLU++, ACE05, Rest14, CoNLL03), with the top row reserved for validation and the bottom row for testing. Solid lines trace mean F1 over exit layers (9–24 out of 32) for sequence repetition $r$, while dashed lines mark the corresponding "Ref. Exit Last $r$" baselines (exit at 32nd layer). All results are averages of five runs.

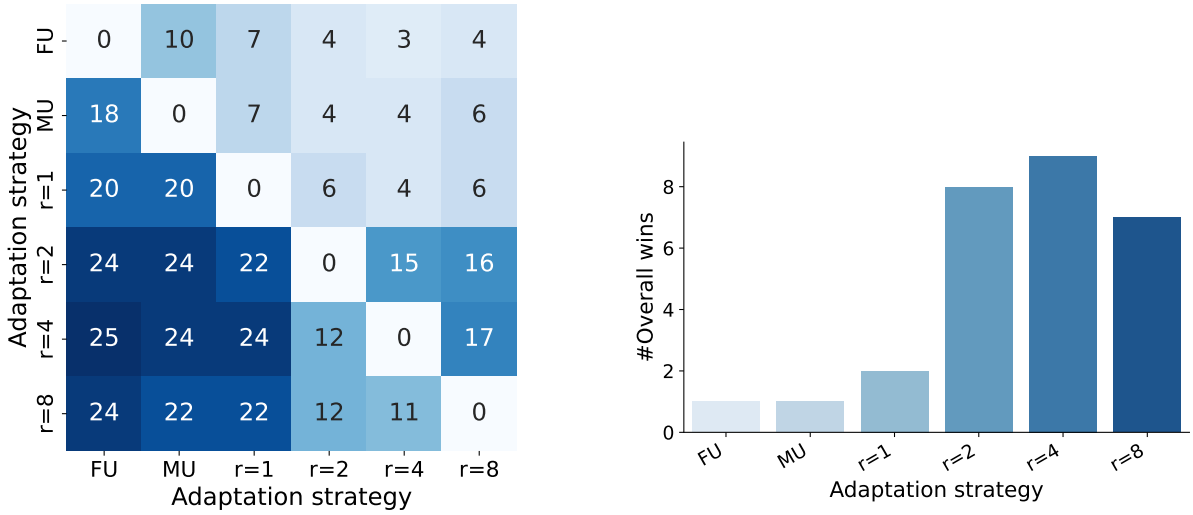
## 5 Related Work

**Encoders and Decoders.** Encoder-only LMs popularized masked language modeling (MLM) with BERT (Devlin et al., 2019), while decoder-only LMs established causal language modeling (CLM) with GPT (Radford et al., 2018), shaping two dominant pre-training paradigms. While scaling studies subsequently reported performance gains for both model families on NLU, most community efforts have since focused on further scaling decoders.

**Causal Mask Removal.** Causal LM pre-training enforces a causal mask that blocks attention to future tokens, a property that benefits left-to-right generation but prohibits token-level representations from being bidirectionally contextualized, which is needed for SL. While decoder-only models exhibit strong performance on sequence classification tasks, their unidirectional context harms the inherently bidirectional token-level tasks. Recent works have relaxed this limitation by removing the causal mask in selected layers (Li et al., 2023; Dukić and Šnajder, 2024), enabling decoder-as-encoder usage and improving sequence labeling. Complementary approaches recast decoders into encoders with specialized training objectives (Huang et al., 2025) or training-free reconfiguration (Lin et al., 2025b).

**Sequence Repetition.** A growing body of work demonstrates that input-level repetition (*echoing*) can strengthen representations without requiring architectural changes or additional pre-training. Repeating the input and extracting embeddings from the echoed span improves the quality of text embeddings, often narrowing the gap to bidirectional encoders (Springer et al., 2025). Initial evidence further suggests benefits of echoing in zero-shot and multilingual settings (Duan et al., 2025). Related ideas include re-reading prompts that enhance reasoning by allowing the model to process the duplicate content twice (Xu et al., 2024), as well as analyses that disentangle mechanisms of repetition in LMs (Mahaut and Franzon, 2025; Hiraoka and Inui, 2025). Our work complements this line of research by investigating whether repetition has a similar effect on token-level embeddings and, consequently, on downstream sequence labeling performance.

**Intermediate-Layer Embeddings Quality.** Early exiting techniques utilize intermediate hidden representations of transformer-based models, rather than final-layer ones, motivated by the fact that the residual stream ties semantic spaces between layers (Geva et al., 2022). Early exiting offers a simple and effective approach for balancing inference costs by leveraging the computational nature of transformers (Teerapittayanon et al., 2016; Huang et al., 2018; Jazbec et al., 2024). Indeed, extracting embeddings from intermediate layers has been shown to yield superior sequence-level representations compared to final-layer pooling (Rep et al., 2024). This benefit has been observed across numerous downstream tasks, not only in NLP but also in computer vision (Alain and Bengio, 2017; Skean et al., 2025).




(a) (b)

Figure 4: Figure 4a shows the adaptation strategy pairwise superiority matrix. Cell $i, j$ contains the number of (model, dataset) pairs for which an adaptation strategy at row $i$ achieved a superior performance than the strategy at column $j$. Figure 4b reports the number of (model, dataset) pairs (28 in total) for which an adaptation strategy achieved a superior result to all other strategies. FU and MU stand for full and middle unmasking, respectively.

## 6 Conclusion

While powerful decoder-only models are reporting competitive results on sequence labeling (SL) tasks, adapting their unidirectional architecture to inherently bidirectional tasks often requires invasive changes, such as removing the causal mask. In contrast, sequence repetition (SR) is a more straightforward, input-level method that enables bidirectionality without altering the model's core structure. We demonstrate that sequence repetition improves token-level representations in decoder-as-encoder setups, yielding consistent SL gains over strong encoders and unmasked decoders. Moreover, we demonstrate that performance continues to improve with multiple repetitions and that an early-exit strategy utilizing intermediate-layer embeddings provides a favorable quality–efficiency trade-off. Furthermore, a qualitative analysis contrasting model performance across adaptation strategies further demonstrates performance benefits obtained by SR, highlighting the effectiveness of this simple method for SL. Taken together, our results establish sequence repetition as an effective, architecture-preserving method for achieving bidirectionality in decoder-only LMs. Future work could extend this approach to multilingual and structured prediction settings and explore adaptive repetition schedules conditioned on instance complexity.

## Limitations

Our approach has a number of key limitations. Firstly, we report means over five runs with different random seeds to improve reliability. Increasing the number of runs would further narrow the confidence intervals. Then, we maintainin fixed learning rates and hyperparameters across experiments. This may have led to suboptimal adaptation for SL. Furthermore, additional unmasking configurations other than unmasking only the middle layers could yield further performance improvements. However, trying out even a subset of all possible layer group unmasking configurations would be computationally demanding. While the open-weight landscape is rich, our experiments target models with 1.7 to 9 billion parameters. Experimenting with larger open-weight LLMs could yield further insights into the behavior of decoders when presented with additional sequence repetitions. Crucially, our experiments are limited to English language datasets. Extending to other languages and more diverse corpora should be explored as a next step. Finally, although sequence repetitions evidently help to improve the performance of decoders on SL tasks, they become increasingly inefficient to compute beyond $r = 4$, even when early exiting at layer $L = 24$, offering diminishing returns relative to their additional cost.

## References


Guillaume Alain and Yoshua Bengio. 2017. Understanding intermediate layers using linear classifier probes.

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. 2024. Llm2vec: Large language models are secretly powerful text encoders. In *First Conference on Language Modeling*.

Inigo Casanueva, Ivan Vulić, Georgios Spithourakis, and Paweł Budzianowski. 2022. NLU++: A multi-label, slot-rich, generalisable dataset for natural language understanding in task-oriented dialogue. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1998–2013, Seattle, United States. Association for Computational Linguistics.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient finetuning of quantized LLMs. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ACE) program – tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

Yifei Duan, Raphael Shang, Deng Liang, and Yongqiang Cai. 2025. Retrieval backward attention without additional training: Enhance embeddings of large language models via repetition. *arXiv preprint arXiv:2502.20726*.

David Dukić and Jan Šnajder. 2024. Looking right is sometimes right: Investigating the capabilities of decoder-only LLMs for sequence labeling. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14168–14181, Bangkok, Thailand. Association for Computational Linguistics.

Siqi Fan, Xin Jiang, Xiang Li, Xuying Meng, Peng Han, Shuo Shang, Aixin Sun, Yequan Wang, and Zhongyuan Wang. 2024. Not all layers of llms are necessary during inference. *CoRR*, abs/2403.02181.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 30–45, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Tatsuya Hiraoka and Kentaro Inui. 2025. Repetition neurons: How do language models produce repetitions? In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 483–495, Albuquerque, New Mexico. Association for Computational Linguistics.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength natural language processing in Python.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q. Weinberger. 2018. Multi-scale dense networks for resource efficient image classification. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Zixian Huang, Xinwei Huang, Ao Wu, Xiaxia Wang, and Gong Cheng. 2025. Transforming decoder-only models into encoder-only models with improved understanding capabilities. *Knowledge-Based Systems*, 309:112907.

Metod Jazbec, Alexander Timans, Tin H Veljković, Kaspar Sakmann, Dan Zhang, Christian A Naesseth, and Eric Nalisnick. 2024. Fast yet safe: Early-exiting with risk control. *Advances in Neural Information Processing Systems*, 37:129825–129854.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. *arXiv preprint arXiv:2310.06825*.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, and 13 others. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zongxi Li, Xianming Li, Yuzhang Liu, Haoran Xie, Jing Li, Fu-lee Wang, Qing Li, and Xiaoqin Zhong. 2023. Label supervised Llama finetuning. *arXiv preprint arXiv:2310.01208*.

Ailiang Lin, Zhuoyun Li, and Kotaro Funakoshi. 2025a. Causal2vec: Improving decoder-only llms as versatile embedding models. *arXiv preprint arXiv:2507.23386*.

Ziyong Lin, Haoyi Wu, Shu Wang, Kewei Tu, Zilong Zheng, and Zixia Jia. 2025b. Look both ways and no sink: Converting LLMs into text encoders without training. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 22839–22853, Vienna, Austria. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Matéo Mahaut and Francesca Franzon. 2025. Repetitions are not all alike: distinct mechanisms sustain repetition in language models. *arXiv preprint arXiv:2504.01100*.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. SemEval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35, Dublin, Ireland. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, and 1 others. 2018. Improving language understanding by generative pre-training.

Ivan Rep, David Dukić, and Jan Šnajder. 2024. Are ELECTRA's sentence embeddings beyond repair? the case of semantic textual similarity. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 9159–9169, Miami, Florida, USA. Association for Computational Linguistics.

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Nikul Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. 2025. Layer by layer: Uncovering hidden representations in language models. In *Forty-second International Conference on Machine Learning*.

Jacob Mitchell Springer, Suhas Kotha, Daniel Fried, Graham Neubig, and Aditi Raghunathan. 2025. Repetition improves language model embeddings. In *The Thirteenth International Conference on Learning Representations*.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, and 89 others. 2024a. Gemma: Open models based on Gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, and 179 others. 2024b. Gemma 2: Improving open language models at a practical size. *Preprint*, arXiv:2408.00118.

Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. 2016. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd international conference on pattern recognition (ICPR)*, pages 2464–2469. IEEE.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task chunking. In *Fourth Conference on Computational Natural Language Learning and the Second Learning Language in Logic Workshop*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021. Automated concatenation of embeddings for structured prediction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2643–2660, Online. Association for Computational Linguistics.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Griffin Thomas Adams, Jeremy Howard, and Iacopo Poli. 2025. Smarter, better, faster, longer:

A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2526–2547, Vienna, Austria. Association for Computational Linguistics.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2246–2251, Online. Association for Computational Linguistics.

Xiaohan Xu, Chongyang Tao, Tao Shen, Can Xu, Hongbo Xu, Guodong Long, Jian-Guang Lou, and Shuai Ma. 2024. Re-reading improves reasoning in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15549–15575, Miami, Florida, USA. Association for Computational Linguistics.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. 2020. Bert loses patience: fast and robust inference with early exit. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

# A Details of Experimental Setup

## A.1 Classification with Decoders

To perform classification tasks, decoder models must be equipped with a classification layer. We implement this by using a softmax linear layer whose input dimension is the size of the hidden vectors produced by the model, and output dimension of the number of target classes. This assigns a set of logits to every token in the input sequence for each possible label. We input the final hidden state after the decoder model's forward pass into this layer.

## A.2 LLM Adaptation Details

In this section, we outline the methodology behind modifications to decoder-only models: middle layer unmasking (§A.2.1) and sequence repetition (§A.2.2).

### A.2.1 Unmasking of Middle Layers

Firstly, we determine which layers are the middle ones in a given model. The selection of these layers is determined dynamically based on the total number of hidden layers, denoted as $N = \texttt{config.num_hidden_layers}$. The interval bounds are computed as follows:

$$N_u = \left\lfloor \frac{N}{3} \right\rfloor_{\text{even}}$$

$$lb = \frac{N}{2} - 1 - \frac{N_u}{2}, \quad ub = \frac{N}{2} + \frac{N_u}{2}.$$

This formulation ensures that a symmetric range of layers is selected around the midpoint of the model. The parameter *mask_cutoff* defines the width of the interval (approximately one-third of the total number of decoder layers, rounded to an even number), while $lb$ and $ub$ mark the lower and upper bounds of the chosen layer indices, meaning layers with the indices in $[lb, ub]$ are unmasked.

### A.2.2 Sequence Repetition

Repetition is implemented by concatenating a given input tensor $k$ times. This is done before the model's forward pass, meaning that padding tokens are also repeated. We also experimented with padding that leaves no padding tokens between each repetition but found no significant difference in performance, likely because rotary positional embeddings encode relative positional information (Su et al., 2024).

## A.3 Tokenization, Optimization and Compute

RoBERTa requires tokenization with added prefix space, and its tokenizer was the only one enabled with it. We adjust the cross-entropy loss to consider only the first token of each tokenized word from the input sequence. Maximum length for a tokenized sequence is set to 256 tokens since no sequence in the tested datasets exceed it. All of the reports are computed from five runs with $seeds = 5, 29, 42, 81, 123$. Tokenized sequences are padded to the length of the longest sequence in a given batch.

In our QLoRA usage, we quantize all models with four-bit precision. Model parameters are stored using a normalized four-bit floating-point format, $nf4$. Computation is performed in a half-precision $fp16$ format. No double quantization is performed. This configuration reduces the memory footprint when training models, allowing us to efficiently perform training on the resources available to us.

The total GPU usage for all experiments amounts to 2800 hours on *Ampere A100* GPU.

| Model family | Model identifier |
|---|---|
| RoBERTa | FacebookAI/roberta-large |
| ModernBERT | answerdotai/ModernBERT-large |

Table 3: Encoder-only models and their Hugging Face identifiers

| Model family | Model identifier |
|---|---|
| Gemma | google/gemma-7b |
| Gemma2 | google/gemma-2-2b |
| Gemma2 | google/gemma-2-9b |
| Qwen3 | Qwen/Qwen3-1.7B |
| Qwen3 | Qwen/Qwen3-4B |
| Qwen3 | Qwen/Qwen3-8B |
| Mistral | mistralai/Mistral-7B-v0.3 |

Table 4: LLMs and their Hugging Face identifiers

## B Full Experimental Results

In Table 4 we report the full results for all the considered models, datasets, and adaptation strategies.

## C AI Use Statement

During the preparation of this manuscript, we used AI assistance for language editing, rephrasing, and spell-checking.

| Model | ABSA-Restaurants | ACE | CoNLL2003 | NLU++ |
|---|---|---|---|---|
| **Encoders** | | | | |
| ModernBERT | 0.7551 (±0.0064) | 0.6830 (±0.0264) | 0.9002 (±0.0015) | 0.7050 (±0.0205) |
| RoBERTa | **0.8030** (±0.0050) | 0.6896 (±0.0508) | **0.9264** (±0.0008) | **0.7222** (±0.0106) |
| **Unmasked models** | | | | |
| gemma-7B-unmasked | 0.7663 (±0.0314) | 0.7308 (±0.0417) | 0.9275 (±0.0010) | 0.7434 (±0.0175) |
| gemma-7B-unmasked-middle | **0.8364** (±0.0043) | 0.7215 (±0.0350) | 0.9146 (±0.0311) | 0.7990 (±0.0212) |
| gemma-2-2B-unmasked | 0.8126 (±0.0041) | 0.7433 (±0.0265) | 0.9282 (±0.0022) | 0.7642 (±0.0136) |
| gemma-2-2B-unmasked-middle | 0.8221 (±0.0026) | 0.7230 (±0.0252) | 0.9260 (±0.0020) | 0.7821 (±0.0128) |
| gemma-2-9B-unmasked | 0.8255 (±0.0081) | 0.7579 (±0.0234) | 0.9344 (±0.0008) | 0.7683 (±0.0132) |
| gemma-2-9B-unmasked-middle | 0.8334 (±0.0061) | 0.7591 (±0.0185) | 0.9317 (±0.0023) | 0.7763 (±0.0077) |
| mistral-7B-unmasked | 0.8318 (±0.0053) | 0.7626 (±0.0238) | **0.9369** (±0.0015) | **0.8023** (±0.0078) |
| mistral-7B-unmasked-middle | 0.8300 (±0.0056) | **0.7756** (±0.0209) | 0.9348 (±0.0008) | 0.7935 (±0.0033) |
| qwen3-1.7B-unmasked | 0.7532 (±0.0126) | 0.6868 (±0.0214) | 0.9027 (±0.0029) | 0.6728 (±0.0268) |
| qwen3-1.7B-unmasked-middle | 0.7571 (±0.0065) | 0.6671 (±0.0244) | 0.9031 (±0.0026) | 0.7367 (±0.0092) |
| qwen3-4B-unmasked | 0.7776 (±0.0039) | 0.7059 (±0.0283) | 0.9117 (±0.0011) | 0.7379 (±0.0074) |
| qwen3-4B-unmasked-middle | 0.7960 (±0.0026) | 0.6973 (±0.0184) | 0.9187 (±0.0021) | 0.7456 (±0.0051) |
| qwen3-8B-unmasked | 0.8001 (±0.0026) | 0.6901 (±0.0197) | 0.9199 (±0.0018) | 0.7445 (±0.0172) |
| qwen3-8B-unmasked-middle | 0.8017 (±0.0047) | 0.7145 (±0.0188) | 0.9223 (±0.0024) | 0.7506 (±0.0075) |
| **Repeat models** | | | | |
| gemma-7B-repeat-k-1 | 0.8216 (±0.0113) | 0.7172 (±0.0218) | 0.9320 (±0.0020) | 0.7970 (±0.0171) |
| gemma-7B-repeat-k-2 | 0.8260 (±0.0073) | 0.7467 (±0.0140) | 0.9338 (±0.0015) | 0.8044 (±0.0146) |
| gemma-7B-repeat-k-4 | 0.8240 (±0.0050) | 0.7427 (±0.0089) | 0.9338 (±0.0011) | **0.8129** (±0.0141) |
| gemma-7B-repeat-k-8 | 0.8179 (±0.0111) | 0.7162 (±0.0379) | 0.9325 (±0.0010) | 0.7934 (±0.0129) |
| gemma-2-2B-repeat-k-1 | 0.8224 (±0.0039) | 0.7411 (±0.0109) | 0.9260 (±0.0010) | 0.7807 (±0.0063) |
| gemma-2-2B-repeat-k-2 | 0.8303 (±0.0065) | 0.7286 (±0.0230) | 0.9305 (±0.0019) | 0.7998 (±0.0077) |
| gemma-2-2B-repeat-k-4 | 0.8226 (±0.0068) | 0.7279 (±0.0075) | 0.9309 (±0.0009) | 0.7927 (±0.0106) |
| gemma-2-2B-repeat-k-8 | 0.8250 (±0.0046) | 0.7287 (±0.0274) | 0.9287 (±0.0009) | 0.7891 (±0.0057) |
| gemma-2-9B-repeat-k-1 | 0.8246 (±0.0040) | 0.7669 (±0.0166) | 0.9344 (±0.0015) | 0.7776 (±0.0096) |
| gemma-2-9B-repeat-k-2 | 0.8351 (±0.0056) | 0.7449 (±0.0108) | 0.9358 (±0.0015) | 0.7883 (±0.0156) |
| gemma-2-9B-repeat-k-4 | 0.8314 (±0.0076) | 0.7259 (±0.0201) | 0.9359 (±0.0010) | 0.7850 (±0.0098) |
| gemma-2-9B-repeat-k-8 | 0.8304 (±0.0014) | 0.7383 (±0.0270) | 0.9351 (±0.0011) | 0.7817 (±0.0085) |
| mistral-7B-repeat-k-1 | 0.8384 (±0.0065) | 0.7489 (±0.0236) | 0.9367 (±0.0005) | 0.7993 (±0.0092) |
| mistral-7B-repeat-k-2 | 0.8354 (±0.0024) | 0.7483 (±0.0415) | 0.9366 (±0.0015) | 0.8071 (±0.0045) |
| mistral-7B-repeat-k-4 | 0.8331 (±0.0099) | **0.7779 (±0.0374)** | **0.9379 (±0.0012)** | 0.8028 (±0.0112) |
| mistral-7B-repeat-k-8 | **0.8418** (±0.0024) | 0.7536 (±0.0235) | 0.9370 (±0.0014) | 0.8105 (±0.0205) |
| qwen3-1.7B-repeat-k-1 | 0.7776 (±0.0031) | 0.6962 (±0.0162) | 0.9085 (±0.0028) | 0.7453 (±0.0121) |
| qwen3-1.7B-repeat-k-2 | 0.7925 (±0.0016) | 0.7066 (±0.0226) | 0.9153 (±0.0017) | 0.7667 (±0.0097) |
| qwen3-1.7B-repeat-k-4 | 0.7926 (±0.0065) | 0.7018 (±0.0216) | 0.9156 (±0.0026) | 0.7775 (±0.0090) |
| qwen3-1.7B-repeat-k-8 | 0.7895 (±0.0081) | 0.7075 (±0.0254) | 0.9178 (±0.0018) | 0.7608 (±0.0099) |
| qwen3-4B-repeat-k-1 | 0.8014 (±0.0119) | 0.7353 (±0.0247) | 0.9190 (±0.0032) | 0.7764 (±0.0071) |
| qwen3-4B-repeat-k-2 | 0.8085 (±0.0055) | 0.7072 (±0.0344) | 0.9219 (±0.0011) | 0.7843 (±0.0090) |
| qwen3-4B-repeat-k-4 | 0.8116 (±0.0050) | 0.6969 (±0.0263) | 0.9230 (±0.0011) | 0.7812 (±0.0147) |
| qwen3-4B-repeat-k-8 | 0.8122 (±0.0030) | 0.7258 (±0.0328) | 0.9233 (±0.0018) | 0.7797 (±0.0082) |
| qwen3-8B-repeat-k-1 | 0.8147 (±0.0033) | 0.6983 (±0.0511) | 0.9224 (±0.0022) | 0.7638 (±0.0178) |
| qwen3-8B-repeat-k-2 | 0.8226 (±0.0013) | 0.7141 (±0.0296) | 0.9281 (±0.0020) | 0.7788 (±0.0099) |
| qwen3-8B-repeat-k-4 | 0.8227 (±0.0070) | 0.7211 (±0.0369) | 0.9274 (±0.0016) | 0.7761 (±0.0159) |
| qwen3-8B-repeat-k-8 | 0.8196 (±0.0051) | 0.7196 (±0.0379) | 0.9300 (±0.0006) | 0.7753 (±0.0076) |

Table 5: Model performance (mean ± 95% CI) across ABSA-Restaurants, ACE, CoNLL2003, and NLU++. Bold values indicate the highest mean per dataset within each group.