LAIT: Efficient Multi-Segment Encoding in Transformers with Layer-Adjustable Interaction

Jeremiah Milbauer[1,2]*, **Annie Louis**[1], **Mohammad Javad Hosseini**[1], **Alex Fabrikant**[1], **Donald Metzler**[1], **Tal Schuster**[1]

[1]Google Research [2]Carnegie Mellon University

Abstract

Transformer encoders contextualize token representations by attending to all other tokens at each layer, leading to quadratic increase in compute effort with the input length. In practice, however, the input text of many NLP tasks can be seen as a sequence of related segments (e.g., the sequence of sentences within a passage, or the hypothesis and premise in NLI). While attending across these segments is highly beneficial for many tasks, we hypothesize that this interaction can be delayed until later encoding stages. To this end, we introduce **L**ayer-**A**djustable **I**nteractions in **T**ransformers (LAIT). Within LAIT, segmented inputs are first encoded independently, and then jointly. This partial two-tower architecture bridges the gap between a Dual Encoder's ability to precompute representations for segments and a fully self-attentive Transformer's capacity to model cross-segment attention. The LAIT framework effectively leverages existing pretrained Transformers and converts them into the hybrid of the two aforementioned architectures, allowing for easy and intuitive control over the performance-efficiency tradeoff. Experimenting on a wide range of NLP tasks, we find LAIT able to reduce 30-50% of the attention FLOPs on many tasks, while preserving high accuracy; in some practical settings, LAIT could reduce actual latency by orders of magnitude.

1 Introduction

Although the meaning of a sentence may depend on the context in which it appears, sentences still have meaning *per se*. However, in tasks involving reasoning across multiple sentences or text segments — like natural language inference (NLI), fact verification, question answering (QA), semantic similarity (STS), etc. — the common setting is to concatenate and *jointly* process all tokenized segments as input to a neural model, most often some form of bidirectional Transformer-based architecture (Vaswani et al., 2017). In this setting, the self-attention blocks of the Transformer layers contextualize the per-token representations against all other input tokens, including those of different input segments. The potential for independent sentence-level semantics is largely ignored.

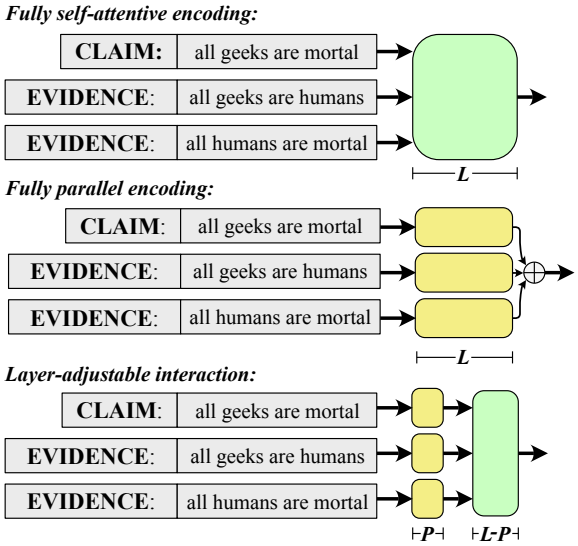

Figure 1: A comparison of three approaches to multi-segment modeling for an arbitrary claim verification task. a) Fully-self attentive architecture, with each token attending to each other token over L layers. b) Generalized dual encoder, with each segment encoded separately by an L-layer Transformer and representations concatenated. c) Layer-adjustable interactions (**ours**), with N layers of independent segment encoding and $L - P$ layers of fully self-attentive segment encoding.

While this practice has shown to achieve high accuracy, it is computationally expensive due to the quadratic increase in cost with the input length. And in practical settings, such as large-scale citation retrieval (Petroni et al., 2022a) or document-level NLI (Koreeda and Manning, 2021), where a given segment may occur multiple times, the full Cartesian product of the sets of text segments

* Work done as an intern at Google Research.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 10251–10269
July 9-14, 2023

must be processed, e.g., Schuster et al. (2022a) processes all sentence pairs from two Wikipedia articles around one subject but in two different languages to identify potential discrepancies. This leads to yet another quadratic increase in cost. Our goal is to reduce both of these computational burdens, rendering transformer architectures more efficient for large-scale multi-segment reasoning.

In this paper, we present LAIT (/leɪt/), a late interaction Transformer model with easy to implement **L**ayer-**A**djustable **I**nteractions. LAIT includes encoder layers that process each segment locally and independent of the other segments, followed by traditional Transformer layers, in a simple but effective way. Unlike the late interaction components of other models, such as ColBERT (Khattab and Zaharia, 2020), which are specifically geared toward measuring a similarity score between two text segments, LAIT generally supports any sequence-to-sequence task and any number of input segments.

LAIT enables several desirable properties for an efficient encoder: it (1) is easy to train on top of existing pretrained language models; (2) readily supports any seq-2-seq task, and any segmentation of the input; (3) improves the encoding efficiency by skipping a large number of attention computations; (4) disentangles independent segment representations from joint processing to allow caching of intermediate segment representations for repeated computations; and (5) provides an easy-to-tune hyperparameter for controlling the efficiency-performance tradeoff.

2 Background: Full Self-attention vs. Dual Encoders

A key strength of a fully self-attentive (FSA) architecture, such as BERT or T5 (Devlin et al., 2019; Raffel et al., 2020) is the ability of each token in the input to interact with each other token in the input throughout all layers of the model. Although expensive, this type of architecture has shown impressive performance across a wide variety of NLP tasks such as those in the GLUE and SuperGLUE benchmarks (Wang et al., 2019b,a).

A common alternative to FSA is the dual encoder (DE) framework (Gillick et al., 2018). With DE, two text segments are embedded independently, either by separate networks or by two networks that share parameters. A DE typically involves two encoders, $\text{Enc}_q(\cdot)$ and $\text{Enc}_d(\cdot)$, and a comparison function $\text{Comp}(\cdot)$, and for a given pair of input segments q, d: $\text{score} = \text{Comp}(\text{Enc}_q(q), \text{Enc}_d(d))$. In practice, the two encoders can share parameters.

DE is typically trained with a contrastive loss over a set of positive q, d pairs, with the goal of having the score of positive pairs greater than that of negatives. Therefore, DE is most suited for similarity tasks such as information retrieval.

A specific advantage of the DE architecture for retrieval tasks is its ability to independently encode the two input segments. In practice, this allows encoding and storing many documents' representations in parallel in advance. Then, only new queries need to be encoded into a vector that can be used for retrieving the top similar documents from the pre-encoded corpus using efficient methods such as maximum inner product search (MIPS).

The method above, however, only supports similarity tasks or binary classification tasks over input pairs. To expand this setting to multi-class tasks, prior approaches like Casanueva et al. (2020); Ni et al. (2022) add a classification head with optional non-linear layers on top of the two encoded representations. Since the classifier requires a fixed-size input, the segment representations are aggregated (e.g., by taking the average over tokens, or by selecting a predefined special token). While conceptually enabling any classification task, the performance of such models is usually far behind the state-of-the-art (see Section 5).

3 Layer-Adjustable Interactions

We argue that both FSA and DE Transformer models can be seen as special cases of a general architecture with adjustable layer depths for both segment-independence and segment-interaction, which we will call a "**L**ayer-**A**djustable **I**nteraction **T**ransformer" (LAIT).

For a Transformer with L layers and an input with N segments, LAIT is a set of N independent stacks of P layers each, followed by $L - P$ fully self-attentive encoder layers. Any function can be used after the encoder. Thus a typical fully self-attentive Encoder-Decoder Transformer is a LAIT where $P = 0$, and a shared-parameter dual encoder is a LAIT where $P = L$ and $N = 2$. In the fully self-attentive Transformer, each token in each segment is interacting with each token in each other segment throughout the entire depth of the encoder; in a Dual Encoder, each segment is treated independently throughout the encoder.

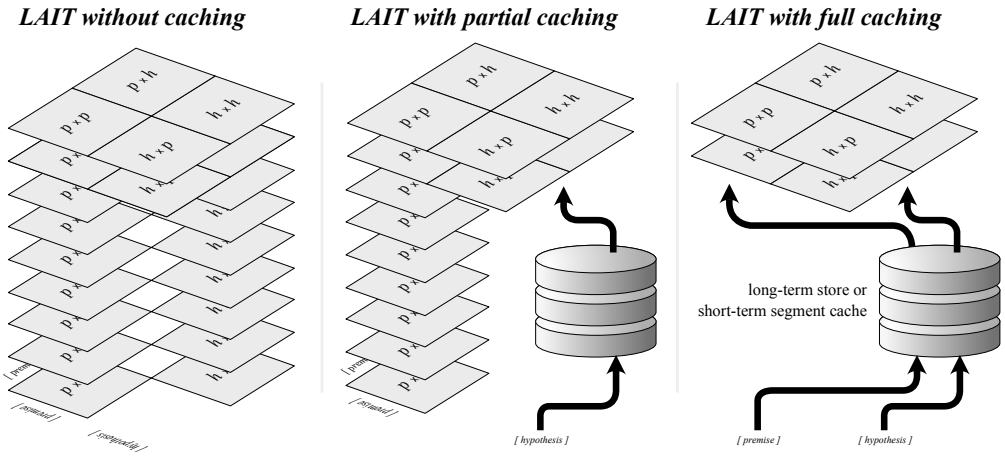

Figure 2: Depiction of a 9-layer LAIT architecture for a 2 segment task (such as NLI) with 7 parallel layers and 2 fully self-attentive layers. Without caching, LAIT reduces computation by eliminating cross-attention for 7 layers. With partial or full caching of segments, LAIT achieves further reductions by re-using independently encoded segment representations.

The LAIT framework allows us to make the core questions of this work precise: (1) to what extent are interactions across multiple input text segments necessary? And (2) If they are not always necessary, how can we take advantage of this fact to perform multi-segment modeling efficiently at scale?

Specifically, given an input X with m tokens that is split into n segments $s_i \dots s_n$ of possibly different lengths, the LAIT encoder is defined as:

$$\text{LAIT}(s_1, s_2, \dots, s_n) = \\ \text{Enc}_{L-P}([\text{Enc}_P(s_1); \text{Enc}_P(s_2); \dots; \text{Enc}_P(s_n)]),$$

where $[x; y]$ denotes concatenating vectors x and y, and $\text{Enc}_K(\cdot)$ denotes a Transformer encoder with K layers.

The rule for splitting the input into segments $R(x_1, \dots, x_m) \rightarrow s_1, \dots, s_n$ is predefined for each task, based either on prior knowledge of the input structure, or on a simple segmentation function. For example, in NLI we can simply use the hypothesis and premise as two segments. In passage-level QA, we can use the question as one segment and the passage as another. However, splitting the passage into multiple shorter segments could help further reduce compute. For instance, we can split the passage by sentences to k segments, leading to a total of $k + 1$ segments.

For $P \in [0, L]$, LAIT interpolates between an N-Encoder model and a fully-self attentive Transformer. Because interaction between segments is delayed, representations computed at layer P of the model can be stored or cached for later reuse as they are independently generated. Figure 2 demonstrates the basic LAIT architecture, as well as possibilities for partial caching (for instance, multiple unique questions about the same passage), or full caching (for instance, NLI-based cross-document reasoning (Schuster et al., 2022a)).

Similar to general text-to-text models, the outputs of the LAIT encoder, consisting of m contextualized representations for m tokens, are passed to the Transformer-decoder for generating the output sequence. Similarly, the decoder may be replaced with a classification head, or any other module.

3.1 Attention Complexity

By first processing text independently, and then processing the intermediate representations jointly, LAIT reduces the attention complexity within a Transformer in accordance with both the degree of independence (i.e., P) and the balance of length across segment inputs. We can calculate the number of attention operations, $\mathcal{O}$, for a given input to LAIT with the formula:

$$\mathcal{O} = \mathcal{O}_{\text{PAR}} + \mathcal{O}_{\text{FSA}} \quad (1)$$

$$\mathcal{O}_{\text{PAR}} = P \cdot \sum_{i=1}^{n} |s_i|^2 \quad (2)$$

$$\mathcal{O}_{\text{FSA}} = (L - P) \cdot \Big[\sum_{i=1}^{n} |s_i|\Big]^2 \quad (3)$$

where $|s_i|$ denotes the length of segment i out of n total segments for a given input.

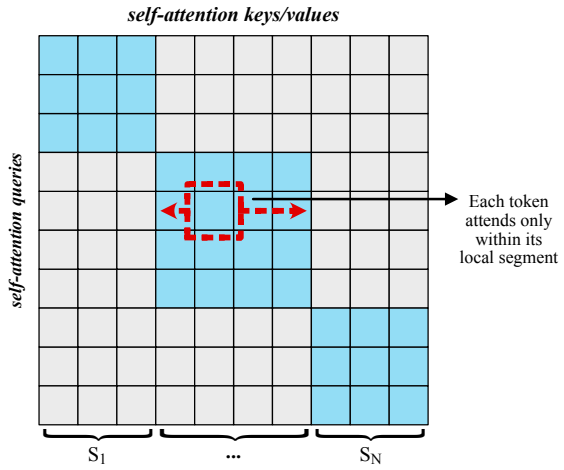

Figure 3: In the parallel layers of LAIT, segments are concatenated but a block-diagonal attention mask maintains independent encoding of each segment. Figure design adopted from Guo et al. (2021).

Ultimately, the number of FLOPs to process a single example will depend on the lengths of the input segments, the Transformer architecture used, and the degree of independence P. We discuss these practical details in Section 4.2, and Table 4.

3.2 Training LAIT

Thanks to LAIT not adding any new parameters to the Transfomer architecture, we can easily convert an existing Transformer to the LAIT framework and train it end-to-end with any objective. In this work, we focus on the T5 (Raffel et al., 2020) model since it is a general text-to-text Transfomer, and apply LAIT to the encoder stack. In our experiments here, since we focus on classification tasks, we only keep a single decoding layer.

Given an input with n text segments, LAIT first encodes and concatenates the segments. During encoding, a block-diagonal attention mask restricts attention between different text segments for the early layers of the model (denoted "parallel layers"), and allows cross-segment attention for the later layers of the model ("joint layers"). Figure 3 illustrates the block-diagonal attention mask used for parallel layers.

This approach allows for parameter sharing while independently encoding the segments, as well as flexibility for tasks with different numbers of input segments without needing to initialize additional models.

4 Experimental Setting

Below, we describe our evaluation setting, tasks, used metrics, and baselines.

4.1 Implementation details

We implement LAIT on top of the T5 model (Raffel et al., 2020) using Google's T5x library (Roberts et al., 2022). In all experiments, we use `T5-base` which has a total of 12 encoder layers and 220M parameters. To reduce compute effort, we use only a single decoder layer for LAIT (See Appendix B.1 for larger models). We load the parameters from the public pretrained checkpoint, and finetune on the target task for up to 100K steps with different LAIT configurations (value of P). We train LAIT on 16 TPUv3 chips, taking about 4 hours per run. We run a small grid search over learning rate and batch size configurations, and pick the top performing checkpoint based on validation performance.

4.2 Tasks and metrics

We experiment using LAIT on a diverse set of common tasks and datasets. For each task, we must determine which fields of the dataset to use as input segments for LAIT. We evaluate each task using its typical quality metric. In addition, to measure the efficiency gains of different LAIT configurations, we compute the average self-attention FLOPs. We use Equation (1) and the precise configuration of the `T5-base` model we implement LAIT within, which has 768-dimensional embeddings and 12 64-dimensional attention heads.

The evaluated tasks are described below. Many of these tasks are from the popular GLUE (Wang et al., 2019b) and SuperGLUE (Wang et al., 2019a) benchmarks, and all are in English. Number of used segments and average lengths per task are summarized in Table 1. Pre-processing and concatenation strategy are described in Appendix A.

MNLI (Williams et al., 2018): A dataset for natural language inference across diverse categories. We use the hypothesis and premise as separate segments, and predict one of three labels: "entailment", "contradiction", and "neutral". We report accuracy on the "matched" eval set.

RTE: The Recognizing Textual Entailment dataset combines the data from a series of annual textual entailment challenges (Dagan et al., 2005; Bar-Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009). We use the hypothesis and the

Task	n	Avg. segment lengths
MNLI	2	hyp.: 16.14, prem.: 30.79
RTE	2	hyp.: 9.40, prem.: 43.39
QQP	2	q.1: 11.94, q.2: 12.17
STSB	2	sent.1: 19.71, sent.2: 19.75
AE	3	cand.: 6.80, ref.: 6.12, q.: 12.10
BoolQ	2	pass.: 135.82, q.: 14.54
BoolQ-Split	6	pass.1-5: 29.57, q.: 14.54
WiC	2	w.+sent.1: 14.69, w.+sent.2: 14.88
FEVER	2	claim: 15.90, evid.: 46.20
VitaminC	2	claim: 21.43, evid.: 43.78
MultiRC	3	pass.: 253.49, q.: 11.70, ans.: 5.84

Table 1: Summary of the evaluated tasks: number of segments (n) and average token length of each segment. Measured on training sets.

premise as separate segments and predict "entailment" vs. "non-entailment", and measure accuracy.

QQP (Iyer et al., 2017): Quora Question Pairs dataset is a collection of question pairs from Quora, where the task is to determine whether a pair of questions have the same meaning. For LAIT, we treat each question as a segment, and predict "duplicate" or "not_duplicate", and measure accuracy.

STSB (Cer et al., 2017): Semantic textual similarity benchmark, a task for estimating the similarity of a pair of sentences. We use each sentence as a separate segment, and predict a score in $[0, 5]$, represented as a string rounded to 2 decimal places. We measure Spearman correlation.

AE (Bulian et al., 2022): Answer Equivalence requires determining whether a "candidate" answer is semantically equivalent to a "reference" answer, given a question. We use the question and each of the answers as independent text segments, make a binary prediction "true" or "false", and measure accuracy.

BoolQ (Clark et al., 2019): Boolean Questions is a binary question answering task with passages and questions. We use the provided text passage and the question as text segments, and make a binary prediction "true" or "false", and measure accuracy.

BoolQ-Split A modification of BoolQ, where each passage is split into 5 sub-passages, treated as independent input segments. The sub-passages are formed by greedily merging the passage's sentences, smallest merge first.

WiC (Pilehvar and Camacho-Collados, 2019): Words in Context is a task for evaluating contextual word meanings. Given a word and two sentences in which it occurs, determine whether the word has the same meaning in each sentence. For LAIT, we prefix each sentence by the specified word and treat the newly-prefixed sentences as our text segments. We then predict "true" or "false", corresponding to whether the word has the same in-context meaning in both sentences. Evaluation is by accuracy.

FEVER (Thorne et al., 2018): A dataset for fact verification with claims and corresponding evidence. Each claim-evidence pair is labeled as "supported," "refuted," or "NotEnoughInfo." For LAIT, we treat the claim and the evidence as our separate text segments, and aim to predict the correct label. Evaluation is done by accuracy.

VitaminC (Schuster et al., 2021): A challenging dataset for fact verification which includes "contrastive evidence", i.e., claim-evidence pairs that differ only slightly (in either the text of the claim or that of the evidence) from another claim-evidence pair, but have a different label. We treat the claim and evidence as independent text segments, and evaluate by accuracy.

MultiRC (Khashabi et al., 2018): The Multi-Sentence Reading Comprehension dataset is a question answering dataset, where each example contains a passage, a question, and an answer[1]. For LAIT, we use the passage, the question, and the answer as the segments. The label is either "True" or "False" meaning whether the answer is correct or not. Evaluation is done by computing the F1 score over all answers.

4.3 Baselines

We compare LAIT against two groups of baselines: Dual Encoder models and Fully self-attentive models. For the Dual Encoder, we use the SentenceT5-base (Ni et al., 2022) shared-parameter Dual Encoder which outputs the concatenation of the average of the per-token output representations from the two encoders, together with their difference and dot product, followed by a classifier. We experiment with two depths of classifier: One with a single non-linear layer, and one with 2 additional hidden layers ($d = 768$ for all layers). As fully self-attentive baselines, we consider `T5-base` and `T5-small` (Raffel et al., 2020).

5 Results

To study the performance-efficiency tradeoff, we consider multiple configurations of LAIT to fully interpolate between a Dual Encoder and a fully self-

[1]The original examples have a list of possible answers to each question, but they are split into one example per answer.

Model	MNLI Accuracy	RTE Accuracy	QQP Accuracy	STSB Spearman	WiC Accuracy	BoolQ Accuracy	FEVER Accuracy	VitaminC Accuracy
DE + 1× MLP	75.40	51.26	90.06	24.88	61.75	69.39	86.16	56.03
DE + 3× MLP	77.22	56.32	89.69	62.16	60.66	69.36	87.09	65.03
T5-small (60M)	83.42	72.92	91.14	88.67	65.83	77.92	96.57	85.35
T5-base (220M)	86.98	84.84	91.94	90.43	72.41	83.12	97.54	88.38
LAIT-0	87.14	80.87	91.80	90.31	70.53	82.45	97.33	88.07
LAIT-1	87.14	79.78	91.94	90.36	68.65	82.54	97.25	87.88
LAIT-2	86.81	81.59	91.87	90.19	70.22	82.39	97.25	87.89
LAIT-3	86.81	79.78	91.96	89.94	69.44	82.35	97.31	87.96
LAIT-4	86.84	81.59	91.84	90.38	69.59	**82.32**	97.26	87.95
LAIT-5	86.80	79.78	91.85	89.91	70.38	80.86	97.17	87.77
LAIT-6	86.23	**80.14**	91.79	89.63	71.16	80.86	97.10	**87.46**
LAIT-7	**86.29**	78.70	91.79	89.72	69.44	80.43	97.07	86.31
LAIT-8	86.08	77.98	91.55	**89.47**	71.79	80.37	97.05	86.49
LAIT-9	85.70	78.34	91.55	89.39	**70.85**	80.40	**96.82**	86.26
LAIT-10	84.42	61.01	**91.07**	82.26	67.40	71.62	95.35	84.27
LAIT-11	83.00	59.57	90.87	53.39	65.05	72.11	92.13	82.75
LAIT-12	73.21	60.29	86.85	22.68	59.56	71.50	88.35	57.00

Table 2: Results comparing LAIT configurations with Dual Encoder and Transformer baselines across a variety of sentence-level reasoning tasks. To make comparison easier with other works, we report the best score on the validation set. See Table 9 for a synthetic test-set comparison of LAIT configurations. Most efficient LAIT model within a 99% performance of LAIT-0 in **bold**, most efficient LAIT model within 95% performance of LAIT-0 is underlined, most efficient LAIT model where the validation score is within the bootstrapped 95% confidence interval of LAIT-0 is boxed.

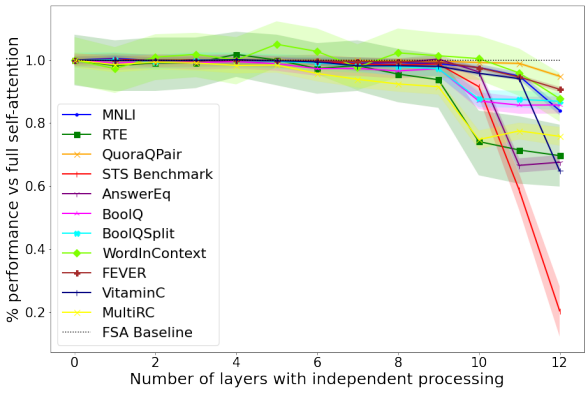

Figure 4: Relative performance of LAIT vs. T5 Fully self-attentive baseline on a variety of multi-segment natural language processing tasks. For all tasks, we report performance on the validation set. Performance only degrades after 8-10 layers of independent segment processing. 95% confidence interval via bootstrapping on the evaluation data.

attentive Transformer. As `T5-base` has a 12-layer encoder, we consider all LAIT-p, for $p \in [0, 12]$, where p is the number of layers of independent segment processing before the fully self-attentive component. Note that LAIT-0 is roughly equivalent to `T5-base`, though it uses a 1-layer decoder vs. the 12-layer decoder of `T5-base`.

As can be seen in Tables 2 and 3, which compare best validation-set performance across models, LAIT either matches, nearly-matches, or outperforms the `T5-base` baseline for every task. This holds even in configurations where cross-segment interaction is delayed to the last few layers of the encoder. As long as there are a few cross-segment interactions later in the model, performance remains relatively stable even as the architecture becomes increasingly efficient; crucially, **LAIT can delay cross-segment interaction by 8-10 layers without a notable decrease in performance.** We specifically focus on the most efficient LAIT models that: (1) achieve within 99% of LAIT-0 performance, which we call LAIT-99%; (2) achieve within 95% of LAIT-0 perfomrance, called LAIT-95%; and (3) achieve within the 95% confidence interval within LAIT-0 performance, called LAIT⋆. To select these models with higher validity, we perform a synthetic dev/test split of the validation sets and report the held-out validation performance of the LAIT models with the highest performance on the synthetic dev set, reported in Appendix B.

These results also suggest differences in the proportion of cross-segment processing necessary for

Model	AnswerEq Accuracy	BoolQ-split Accuracy	MultiRC F1
T5-small	89.65	77.92	73.25
T5-base	91.09	83.12	80.07
LAIT-0	91.25	81.71	78.12
LAIT-1	91.36	82.35	77.86
LAIT-2	90.46	81.93	77.82
LAIT-3	90.89	81.53	77.69
LAIT-4	90.85	**82.11**	77.18
LAIT-5	90.78	80.43	**77.41**
LAIT-6	90.62	80.76	75.60
LAIT-7	90.60	79.94	73.56
LAIT-8	90.06	79.82	71.88
LAIT-9	**90.98**	79.85	71.43
LAIT-10	87.00	72.20	59.50
LAIT-11	61.16	71.13	61.07
LAIT-12	61.02	71.41	59.60

Table 3: Results for tasks with more than two segments. **Bold**, underline, and box indicate model performance as in Table 2.

different tasks. Sentence and word representation tasks (i.e., Answer Equivalence, STSB, and WiC) have much better LAIT⋆ models than reasoning-intensive tasks, such as MNLI, BoolQ, and VitaminC. We note that FEVER appears to be easier for LAIT than other "reasoning" tasks, which we explore further in Section 5.3. We also note that some degree of cross-segment processing is necessary for all tasks, evidenced by the steep drop in performance as p approaches 12 (see Figure 4).

5.1 Scalability

By deferring the expensive cross-segment attention to later stages of the model, LAIT both reduces the attention complexity of the model, and enables the caching and reuse of partial representations computed before the cross-segment attention layers.

Table 4 shows improvements in attention FLOPs for LAIT, both with and without caching of the intermediate representations, when using the LAIT-95% model. Table 10 contains results for LAIT⋆. As we would expect from Equation 1, datasets with text segments of similar size benefit the most in the typical setting. Howevever, to fully realize this benefit for single forward passes would require a custom kernel, such as those implemented in work on sparse transformers.

Task	Full Encoding ↓	with Caching ↓
MNLI	66.66%	39.71%
STSB	63.07%	62.78%
AnswerEq	49.94%	29.73%
BoolQ	89.72%	83.53%
BoolQ-S	42.28%	40.51%
WiC	63.45%	63.45%
FEVER	72.74%	34.68%
VitaminC	67.37%	41.34%
MultiRC	93.91%	50.83%
RTE	92.22%	92.02%
QQP	56.06%	53.37%
Potential practical settings:		
ContractNLI	98.92%	21.50%
WikiClusters	63.02%	16.94%

Table 4: Percent of encoder attention FLOPs (compared to `T5-base`) when using LAIT-95% model for each task to process the entire validation set (lower is better). LAIT-95% selection is based on results in tables 9 and 11 in the Appendix.

5.2 Caching and Reusing Representations

A key advantage of the delayed cross-segment interaction in LAIT is the ability to cache and reuse intermediate representations of text segments. Unlike in benchmarks, real-world settings almost never process a set of segments in isolation; it is much more likely that the processing of a set of text segments occurs as part of a larger task such as document comparison, document analysis, or claim verification.

Recently, a number of datasets (Schuster et al., 2022a; Koreeda and Manning, 2021; Petroni et al., 2022b) have suggested the usefulness of natural language inference in large-scale real-world reasoning tasks. In one such dataset, ContractNLI (Koreeda and Manning, 2021), a fixed set of 17 claims are evaluated against different legal contracts. In other scenarios (Schuster et al., 2022a; Gu et al., 2020), the contents of multiple documents within a cluster of related documents must be compared.

In both scenarios, a typical approach would require comparing each sentence within a document with each other sentence, leading to a complexity that scales quadratically with the size of the document cluster, the size of the documents, and the length of the sentences. But with LAIT, the bulk of the work will be performed only once. Because each document or claim can be encoded independently for most of the layers of the model, the latency improvement offered by LAIT in these

Dataset	FSA	Sparse	LAIT-12	LAIT-95%
MNLI - Full	167.9 (1.3)	275.3 (0.96)	111.3 (1.4)	116.02 + ϵ
BoolQ-S - Full	54.40 (0.43)	87.72 (0.38)	37.51 (0.21)	41.73 + ϵ
ContractNLI - Single	0.0071 (0.0012)	0.0094 (0.0005)	0.0004 (0.0000)	-
ContractNLI - Full	25.03 (0.58)	34.28 (0.46)	0.0593 (0.0008)	-
WikiClusters - Single	1390. (6.0)	1871. (7.1)	1.086 (0.03)	-
WikiClusters - Full	4805. (32.)	5451. (15.)	87.79 (0.78)	-

Table 5: Encoding latency (and standard deviation) comparison in seconds between fully self-attentive T5 (FSA), LongT5 with local attention (Sparse), and LAIT. ϵ represents system-dependent processing of a LAIT cache. Measurements were performed with a 2080Ti GPU, using the Hugging Face (Wolf et al., 2019) implementation of T5 and LongT5.

settings is related to the overall redundancy and duplication of text segments within the task.

Table 5 demonstrates the savings possible for both popular academic tasks, and two realistic settings: ContractNLI (Koreeda and Manning, 2021), and WikiClusters (Schuster et al., 2022a). For MNLI and BoolQ, we measure the time to encode the entire dataset. For WikiClusters and ContractNLI, we both measure the time to encode the entire dataset and the time to encode a single document (in the case of ContractNLI) or cluster (in the case of WikiClusters). We compare a standard fully self-attentive model (T5), a sparse model (LongT5 with local attention), and LAIT. For MNLI and BoolQ, we estimate the latency of the LAIT-95% model for that task, as a weighted average of FSA and LAIT layers.

Even without a custom kernel, LAIT's independent processing of input segments enables significant speedups for processing real-world data. Interestingly, the sparse transformer demonstrates slightly *increased* latency, likely because the the input sizes are relatively short. However, even when enabled by a sparse transformer, processing larger chunks of data – such as an entire ContractNLI contract alongside each of the 17 claims – will not fully alleviate the problem, as the contracts must still be processed 17 times, rather than just once as in LAIT. In these situations, LAIT may be able to complement a sparse transformer; this would require further study.

5.3 Robustness

A potential concern with an approach like LAIT is that it may be more susceptible to reported biases in sentence-level models (Poliak et al., 2018; Schuster et al., 2021). We test LAIT's effect on the model's robustness to domain shifts, and to biases in the training data such as over-relying on clues in one of the segments instead of performing cross-segment reasoning.

Model	FEVER	VitaminC	MNLI
Training Data: FEVER-train			
LAIT-0	97.33	65.12	47.93
LAIT-3	97.31	64.73	45.85
LAIT-6	97.10	63.62	35.15
LAIT-9	96.82	62.97	33.82
LAIT-12	88.35	49.91	34.29
Training Data: VitaminC-train			
LAIT-0	78.54	88.07	80.37
LAIT-3	78.96	87.96	80.01
LAIT-6	78.72	87.46	78.74
LAIT-9	77.70	86.26	76.74
LAIT-12	54.04	57.00	43.38

Table 6: Accuracy of FEVER- and VitaminC-trained LAIT models on FEVER, VitaminC, and MNLI.

Schuster et al. (2021) found that in FEVER, when evidence text in a claim-evidence pair was revised in a way that would reverse the semantic relationship (e.g., f_{revision}(Claim, Evidence, REFUTES) $\rightarrow$ (Claim, Evidence$'$, SUPPORTS), models trained on FEVER would only make the correct prediction 56% of the time. Table 6 summarizes our robustness experiments using zero-shot transfer from FEVER and VitaminC.

We find that when LAIT is trained on on FEVER, the transfer performance drops faster than the in-domain performance as independence is increased. However, when training on VitaminC, the decrease in accuracy as a function of P is more correlated with the in-domain trend. This suggests that LAIT models can be robust against domain shifts and contrastive adversarial examples when trained with appropriate data.

6 Related Work

Sentence encoders. Modern representation learning systems at the sentence level have rapidly risen in popularity, starting with InferSent (Conneau et al., 2017), ESIM (Cer et al., 2018), and USE (Chen et al., 2017). Following the inception of Transformer (Vaswani et al., 2017), new sentence encoders (see e.g., Gao et al., 2021; Ni et al., 2022; Reimers and Gurevych, 2019) demonstrated improved performance on many sentence-pair benchmarks. Other work extended this approach to document encoders by hierarchically encoding sentences independently before combining them into a pooled document embedding (Wu et al., 2021; Yang et al., 2020). Yet, unlike previous work, LAIT effectively breaks a pretrained Transformer into a hybrid of multiple parallel segment encoders and powerful fully-attentive layers to match state-of-the-art performance across many NLP tasks.

Efficient text classifiers Dual encoder architectures, originally dating back to the Siamese architecture of (Bromley et al., 1993), were proposed for efficient retrieval in (Gillick et al., 2018). (Ni et al., 2021) and (Menon et al., 2022) significantly broaden the range of tasks efficiently served by dual encoders.

Building on the Transformer architecture, LAIT can also readily leverage many other known efficiency solutions (Tay et al., 2022) such as distillation (Sanh et al., 2019; Jiao et al., 2020), quantization (Shen et al., 2020; Zafrir et al., 2019), and early exiting (Schuster et al., 2022b; Xin et al., 2020).

Sparse attention. Sparse attention architectures have demonstrated that not all attention connections within a Transformer are necessary, and that impressive performance can be achieved even when removing a large number of the cross-token attention. Examples such as BigBird, Longformer, and LongT5 (Zaheer et al., 2020; Beltagy et al., 2020; Guo et al., 2021) use local attention windows and some form of global attention to reduce the attention complexity. Other approaches dynamically skip certain computations (Tay et al., 2020). Unlike these approaches, here we impose the sparsity on top of known input segments, which preserves segment-level semantics and supports parallel computing and caching of segments. Despite their benefits, sparse transformers still include cross-segment attention at every layer of the model, and as such they cannot encode segments independently.

Late interaction. Some recent work has considered precomputing full-token representations of some, but not all, text segments, as well as late interaction between queries and documents (Lu et al., 2020; Xiong et al., 2017). ColBERT (Khattab and Zaharia, 2020; Santhanam et al., 2022) uses precomputed token representations as part of a DE retrieval framework. These architectures, however, are tailored for retrieval tasks that use embedding similarity scores, and generally under-perform in classification tasks like NLI. The fully-attentive layers in LAIT allow bridging this performance gap while still providing efficiency gains. Our caching variant also relates to other recent parallel work on precomputing and reusing representations of repeated passages to speed up computation (Saad-Falcon et al., 2023; de Jong et al., 2023; Li et al., 2022). Hui et al. (2022) develop a fully parallel encoder for documents and queries, where both encodings are fed to a joint decoder for reranking. Most similar to our work is MacAvaney et al. (2020) that study a hybrid Transformer architecture for ranking. In this work, we focus on general NLP tasks with an arbitrary number of segments, and unconstrained output space.

7 Conclusion

We present Layer-Adjustable Interactions in Transformers (LAIT) to allow simple-but-effective efficiency gains over a wide range of NLP tasks. The LAIT framework leverages existing pretrained Transformers such as T5, and converts them during finetuning into a hybrid model that combines parallel independent encoding of multiple segments, followed by fully-attentive layers to allow cross-segment reasoning.

We evaluate LAIT on a large set of 10 well-known datasets, involving different examined capabilities, number of segments, input lengths, output spaces, and difficulty levels. We find LAIT to consistently provide significant reduction in encoder attention complexity while preserving high accuracy. Furthermore, we show that the parallel independent segment encoding of LAIT enables additional inference-time compute savings by caching representations of repeated segments in large scale real-world settings.

LAIT demonstrates that transformers can achieve high performance even without cross-segment interaction at every layer; essentially, that sentences can be just as effectively encoded if first processed separately, and then processed jointly.

Limitations

While the LAIT framework can significantly reduce the computation required for large-scale sentence-level reasoning and classification tasks, we do foresee some limitations in its use. Caching per-token representations for large numbers of text segments leads to a dramatic increase in memory requirements, which could be prohibitive for extremely low-compute end users. We also note that LAIT can further exacerbate segment-level bias in datasets. While we believe that careful data curation approaches ameliorate this issue, the risk of bias is not always known to downstream users and as such corrective datasets may not always be available. Finally, LAIT can increase the cost of training because the optimal degree of independence is not known until all LAIT-p models are evaluated, though in practical settings (1) it is possible to perform a binary search of LAIT configurations because performance generally decreases monotonically as p increases; (2) even a naive rule of setting p to a quarter of the model's depth seems to provide some immediate gains while preserving 99% of the accuracy in all our evaluated tasks; and (3) inference-time cost improvements will far outweigh training costs.

References

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second pascal recognising textual entailment challenge. In *Proceedings of the second PASCAL challenges workshop on recognising textual entailment*, volume 6, pages 6–4. Venice.

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

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. In *TAC*.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a "siamese" time delay neural network. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, page 737–744, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Jannis Bulian, Christian Buck, Wojciech Gajewski, Benjamin Boerschinger, and Tal Schuster. 2022. Tomayto, tomahto. beyond token-level answer equivalence for question answering evaluation. *arXiv preprint arXiv:2202.07654*.

Iñigo Casanueva, Tadas Temcinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. 2020. Efficient intent detection with dual sentence encoders. pages 38–45.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium. Association for Computational Linguistics.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668, Vancouver, Canada. Association for Computational Linguistics.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2924–2936.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

Michiel de Jong, Yury Zemlyanskiy, Nicholas FitzGerald, Joshua Ainslie, Sumit Sanghai, Fei Sha, and William Cohen. 2023. Pre-computed memory or on-the-fly encoding? a hybrid approach to retrieval augmentation makes the most of your compute.

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.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9. Association for Computational Linguistics.

Daniel Gillick, Alessandro Presta, and Gaurav Singh Tomar. 2018. End-to-end retrieval in continuous space. *arXiv preprint arXiv:1811.08008*.

Xiaotao Gu, Yuning Mao, Jiawei Han, Jialu Liu, You Wu, Cong Yu, Daniel Finnie, Hongkun Yu, Jiaqi Zhai, and Nicholas Zukoski. 2020. Generating representative headlines for news stories. In *Proceedings of The Web Conference 2020*, pages 1773–1784.

Mandy Guo, Joshua Ainslie, David Uthus, Santiago Ontanon, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. 2021. Longt5: Efficient text-to-text transformer for long sequences. *arXiv preprint arXiv:2112.07916*.

Kai Hui, Honglei Zhuang, Tao Chen, Zhen Qin, Jing Lu, Dara Bahri, Ji Ma, Jai Gupta, Cicero Nogueira dos Santos, Yi Tay, and Donald Metzler. 2022. ED2LM: Encoder-decoder to language model for faster document re-ranking inference. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3747–3758, Dublin, Ireland. Association for Computational Linguistics.

Shankar Iyer, Nikhil Dandekar, and Kornel Csernai. 2017. First quora dataset release: Question pairs.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling bert for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface:a challenge set for reading comprehension over multiple sentences. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL)*.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48.

Yuta Koreeda and Christopher Manning. 2021. ContractNLI: A dataset for document-level natural language inference for contracts. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1907–1919, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zonglin Li, Ruiqi Guo, and Sanjiv Kumar. 2022. Decoupled context processing for context augmented language modeling. In *Advances in Neural Information Processing Systems*.

Wenhao Lu, Jian Jiao, and Ruofei Zhang. 2020. Twinbert: Distilling knowledge to twin-structured bert models for efficient retrieval.

Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020. Efficient document re-ranking for transformers by precomputing term representations. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

Aditya Menon, Sadeep Jayasumana, Ankit Singh Rawat, Seungyeon Kim, Sashank Reddi, and Sanjiv Kumar. 2022. In defense of dual-encoders for neural ranking. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 15376–15400. PMLR.

Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. 2022. Sentence-T5: Scalable sentence encoders from pre-trained text-to-text models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1864–1874, Dublin, Ireland. Association for Computational Linguistics.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Ábrego, Ji Ma, Vincent Y. Zhao, Yi Luan, Keith B. Hall, Ming-Wei Chang, and Yinfei Yang. 2021. Large dual encoders are generalizable retrievers.

Fabio Petroni, Samuel Broscheit, Aleksandra Piktus, Patrick Lewis, Gautier Izacard, Lucas Hosseini, Jane Dwivedi-Yu, Maria Lomeli, Timo Schick, Pierre-Emmanuel Mazaré, et al. 2022a. Improving wikipedia verifiability with ai. *arXiv preprint arXiv:2207.06220*.

Fabio Petroni, Samuel Broscheit, Aleksandra Piktus, Patrick Lewis, Gautier Izacard, Lucas Hosseini, Jane Dwivedi-Yu, Maria Lomeli, Timo Schick, Pierre-Emmanuel Mazaré, et al. 2022b. Improving wikipedia verifiability with ai. *arXiv preprint arXiv:2207.06220*.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. Wic: the word-in-context dataset

for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1267–1273.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191, New Orleans, Louisiana. Association for Computational Linguistics.

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

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Adam Roberts, Hyung Won Chung, Anselm Levskaya, Gaurav Mishra, James Bradbury, Daniel Andor, Sharan Narang, Brian Lester, Colin Gaffney, Afroz Mohiuddin, Curtis Hawthorne, Aitor Lewkowycz, Alex Salcianu, Marc van Zee, Jacob Austin, Sebastian Goodman, Livio Baldini Soares, Haitang Hu, Sasha Tsvyashchenko, Aakanksha Chowdhery, Jasmijn Bastings, Jannis Bulian, Xavier Garcia, Jianmo Ni, Andrew Chen, Kathleen Kenealy, Jonathan H. Clark, Stephan Lee, Dan Garrette, James Lee-Thorp, Colin Raffel, Noam Shazeer, Marvin Ritter, Maarten Bosma, Alexandre Passos, Jeremy Maitin-Shepard, Noah Fiedel, Mark Omernick, Brennan Saeta, Ryan Sepassi, Alexander Spiridonov, Joshua Newlan, and Andrea Gesmundo. 2022. Scaling up models and data with `t5x` and `seqio`. *arXiv preprint arXiv:2203.17189*.

Jon Saad-Falcon, Amanpreet Singh, Luca Soldaini, Mike D'Arcy, Arman Cohan, and Doug Downey. 2023. Embedding recycling for language models.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

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

Tal Schuster, Sihao Chen, Senaka Buthpitiya, Alex Fabrikant, and Donald Metzler. 2022a. Stretching sentence-pair nli models to reason over long documents and clusters. *arXiv preprint arXiv:2204.07447*.

Tal Schuster, Adam Fisch, and Regina Barzilay. 2021. Get your vitamin c! robust fact verification with contrastive evidence. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 624–643.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q Tran, Yi Tay, and Donald Metzler. 2022b. Confident adaptive language modeling. In *Advances in Neural Information Processing Systems*.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2020. Q-bert: Hessian based ultra low precision quantization of bert. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8815–8821.

Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. 2020. Sparse sinkhorn attention.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2022. Efficient transformers: A survey. *ACM Comput. Surv.*, 55(6).

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 809–819.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019a. Superglue: A stickier benchmark for general-purpose language understanding systems. *arXiv preprint arXiv:1905.00537*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019b. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In the Proceedings of ICLR.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL-HLT*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers:

State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2021. Hi-transformer: Hierarchical interactive transformer for efficient and effective long document modeling. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 848–853, Online. 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.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic coattention networks for question answering. In *International Conference on Learning Representations*.

Liu Yang, Mingyang Zhang, Cheng Li, Michael Bendersky, and Marc Najork. 2020. Beyond 512 tokens: Siamese multi-depth transformer-based hierarchical encoder for long-form document matching. In *Proceedings of the 29th ACM International Conference on Information &amp Knowledge Management*. ACM.

Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. 2019. Q8bert: Quantized 8bit bert. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)*, pages 36–39. IEEE.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33:17283–17297.

A Segment Preprocessing

For each task, we must prepare the text segments for processing by either the Dual Encoder, Fully Self-attentive, or LAIT models. Here, we report the preprocessing and concatenation strategy used. For the FSA models, we concatenate each segment. For the DE and LAIT models, we treat each segment as a separate input.

MNLI

- `hypothesis: <hypothesis text>`
- `premise: <premise text>`

WiC

- `<key word>: <sentence1>`
- `<key word>: <sentence2>`

STSB

- `sentence1: <sentence1>`
- `sentence2: <sentence2>`

BoolQ

- `question: <question>`
- `passage: <passage>`

RTE

- `hypothesis: <hypothesis>`
- `premise: <premise>`

QQP

- `question1: <question1>`
- `question2: <question2>`

FEVER

- `hypothesis: <claim>`
- `premise: <evidence>`

VitaminC

- `hypothesis: <claim>`
- `premise: <evidence>`

Answer Equivalence

- `question: <question>`
- `answer1: <answer1>`
- `answer2: <answer2>`

MultiRC

- `question: <question>`
- `answer: <answer>`
- `paragraph: <paragraph>`

BoolQ-Split

- `question: <question>`
- `passage1: <passage1>`
- `passage2: <passage2>`
- `passage3: <passage3>`
- `passage4: <passage4>`
- `passage5: <passage5>`

B Additional Results

B.1 Full Decoder and T5-Large Models

For our experiments in the main paper we used a T5-base model with only a single decoder layer. Using only one decoder layer is faster at inference time enforces the model to more heavily rely on the encoder stack, and therefore the strong results of LAIT in that setting are even more encouraging. We also experiment with a LAIT on top of a T5-Base with all 12 decoder layers and with a larger T5-Large that has 24 layers in both encoder and decoder stacks.

Table 7 and Table 8 present the results for T5-Base and T5-Large, respectively. LAIT shows similar trends for these different configurations, indicating that our approach is general and translates to different model configurations. Also, as expected, larger decoder allows LAIT to further postpone the cross-segment interactions (larger P) without loosing accuracy.

B.2 Generalization of LAIT configuration

Here, we report additional results using our split of the existing validation sets into a synthetic validation set and a heldout test set.

Figure 5 reports the decrease in model performance as the number of parallel encoder layers increases. Table 9 reports the heldout test results for the LAIT models with the best synthetic validation performance. Table 11 includes the tasks with more than two segments. Table 10 reports the cost of both full encoding and partially-cached encoding for LAIT$\star$ models identified from Tables 9 and 11.

P	MNLI		QQP		WiC		WiC		MultiRC	
	Accuracy	Relative	Accuracy	Relative	Accuracy	Relative	Accuracy	Relative	F1	Relative
0	86.92		91.86		72.73		83.64		80.26	
1	86.90	99.98%	91.89	100.03%	72.57	99.78%	83.46	99.78%	79.74	99.35%
2	87.05	100.15%	91.90	100.04%	72.88	100.21%	83.49	99.82%	79.95	99.61%
3	87.17	100.29%	91.93	100.08%	73.51	101.07%	83.49	99.82%	79.80	99.43%
4	86.93	100.01%	91.87	100.01%	72.88	100.21%	83.64	100.00%	79.69	99.29%
5	86.60	99.63%	91.94	100.09%	73.51	101.07%	83.15	99.41%	78.92	98.33%
6	86.61	99.64%	91.72	99.85%	73.67	101.29%	82.97	99.20%	78.37	97.65%
7	86.30	99.29%	91.66	99.78%	73.82	101.50%	82.45	98.58%	78.03	97.22%
8	86.15	99.11%	91.73	99.86%	73.67	101.29%	82.48	98.61%	78.13	97.35%
9	86.13	99.09%	91.61	99.73%	73.82	101.50%	82.35	98.46%	77.96	97.13%
10	84.97	97.76%	91.45	99.55%	71.32	98.06%	77.13	92.22%	67.07	83.57%
11	84.17	96.84%	90.98	99.04%	67.87	93.32%	74.74	89.36%	59.06	73.59%
12	83.22	95.74%	89.55	97.49%	64.89	89.22%	73.73	88.15%	58.18	72.49%

Table 7: Results for different number of parallel layers P of LAIT using the same setting as Table 2, but with 12 decoder layers instead of a single decoder layer. Hence, $P = 0$ is similar to T5-base setting from Table 2 (numbers are not identical due to different training runs). The extra decoding layers allows further increasing P compared to single decoder-layer while maintaining similar performance. The relative column shows the accuracy or F1 change compared to $P = 0$.

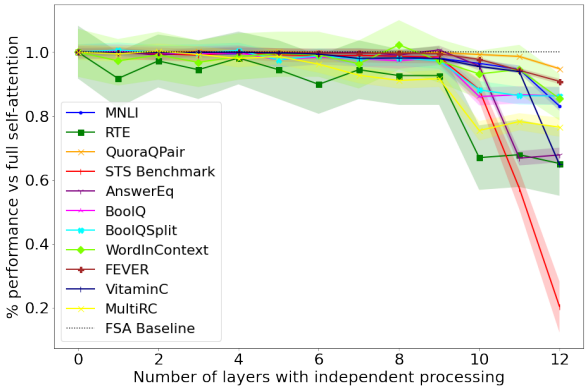

Figure 5: Relative performance of LAIT vs. T5 Fully self-attentive baseline on a variety of multi-segment natural language processing tasks. For all tasks, we report performance on a held-out portion of the validation set. Performance only degrades after 8-10 layers of independent segment processing. 95% confidence interval via bootstrapping on the evaluation data.

P	MNLI		WiC		BoolQ		MultiRC	
	Accuracy	Relative	Accuracy	Relative	Accuracy	Relative	F1	Relative
0	90.19		73.82		86.88		84.16	
1	90.01	99.80%	73.35	99.36%	86.88	100.00%	84.03	99.85%
2	90.16	99.97%	73.82	100.00%	86.76	99.86%	83.76	99.52%
3	90.10	99.90%	73.35	99.36%	86.85	99.97%	84.04	99.86%
4	89.97	99.76%	73.51	99.58%	87.25	100.43%	84.20	100.05%
5	90.09	99.89%	74.14	100.43%	87.19	100.36%	84.26	100.12%
6	89.97	99.76%	74.29	100.64%	87.09	100.24%	84.19	100.04%
7	90.39	100.22%	74.14	100.43%	87.22	100.39%	83.75	99.51%
8	90.15	99.96%	74.45	100.85%	86.88	100.00%	84.04	99.86%
9	90.07	99.87%	73.98	100.22%	87.22	100.39%	83.86	99.64%
10	89.87	99.65%	74.29	100.64%	86.94	100.07%	84.00	99.81%
11	89.84	99.61%	74.45	100.85%	87.03	100.17%	83.82	99.60%
12	90.13	99.93%	74.92	101.49%	87.06	100.21%	83.97	99.77%
13	89.75	99.51%	74.29	100.64%	86.88	100.00%	83.54	99.26%
14	89.59	99.33%	73.82	100.00%	86.45	99.51%	83.11	98.75%
15	89.86	99.63%	72.73	98.52%	86.94	100.07%	82.80	98.38%
16	89.81	99.58%	73.04	98.94%	86.70	99.79%	82.44	97.96%
17	89.50	99.23%	73.98	100.22%	86.09	99.09%	81.85	97.26%
18	89.37	99.09%	73.51	99.58%	86.02	99.01%	81.57	96.92%
19	88.66	98.30%	74.14	100.43%	84.89	97.71%	78.99	93.86%
20	88.50	98.13%	72.88	98.73%	83.33	95.91%	76.66	91.09%
21	88.39	98.00%	73.82	100.00%	82.45	94.90%	74.67	88.72%
22	88.16	97.75%	72.26	97.89%	81.83	94.19%	73.02	86.76%
23	86.93	96.39%	71.16	96.40%	79.24	91.21%	61.11	72.61%
24	85.83	95.17%	68.03	92.16%	76.88	88.49%	59.34	70.51%

Table 8: Results for different number of parallel layers P of LAIT with a T5-Large backbone model, using all 24 decoder layers. The relative column shows the accuracy or F1 change compared to $P = 0$.

Model	MNLI Accuracy	RTE Accuracy	QQP Accuracy	STSB Spearman	WiC Accuracy	BoolQ Accuracy	FEVER Accuracy	VitaminC Accuracy
LAIT-0	86.86± 0.93	**78.42 ± 6.47**	91.57 ± 0.37	89.75± 1.64	68.97± 5.17	81.65± 1.87	97.01± 0.44	87.95± 0.36
LAIT-1	86.86± 0.94	71.94 ± 7.19	91.61 ± 0.39	89.53± 1.68	67.08± 5.17	81.96± 1.81	96.90± 0.46	87.80± 0.37
LAIT-2	86.37± 0.94	76.26 ± 6.51	91.43 ± 0.37	89.24± 1.82	68.34± 5.02	81.59± 1.87	96.92± 0.45	87.83± 0.37
LAIT-3	86.29± 0.93	74.1 ± 7.19	91.87 ± 0.35	88.91± 1.85	66.77± 5.49	81.41± 1.93	96.94± 0.44	87.87± 0.34
LAIT-4	86.43± 0.93	76.98 ± 6.47	91.64 ± 0.37	89.67± 1.63	68.03± 5.49	**81.59± 1.81**	97.01± 0.44	87.92± 0.34
LAIT-5	**86.51± 0.93**	74.1 ± 7.19	91.65 ± 0.38	**88.99± 1.88**	68.65± 5.18	79.82± 1.87	96.71± 0.45	87.73± 0.36
LAIT-6	85.84± 1.01	70.5 ± 7.23	91.53 ± 0.4	88.73± 1.79	68.34± 5.02	80.49± 1.93	96.68± 0.45	**87.41± 0.34**
LAIT-7	85.94± 0.91	74.1 ± 7.19	91.37 ± 0.4	88.82± 1.82	66.46± 5.02	80.06± 1.87	96.68± 0.47	86.21± 0.39
LAIT-8	85.80± 1.00	72.66 ± 7.19	91.4 ± 0.39	88.60± 1.85	**70.53± 4.86**	79.57± 1.99	96.70± 0.47	86.35± 0.39
LAIT-9	85.19± 0.99	72.66 ± 7.19	91.44 ± 0.38	88.38± 1.79	67.08± 5.17	80.37± 1.96	**96.52± 0.48**	86.18± 0.39
LAIT-10	83.80± 1.01	52.52 ± 7.91	**90.89 ± 0.42**	79.21± 2.90	64.26± 5.02	70.40± 2.23	94.80± 0.54	84.00± 0.42
LAIT-11	82.17± 1.10	53.24 ± 7.91	90.33 ± 0.41	51.49± 5.44	65.20± 4.86	70.89± 2.05	91.48± 0.70	82.60± 0.42
LAIT-12	72.19± 1.27	51.08 ± 7.91	86.81 ± 0.47	18.30± 6.94	58.93± 5.49	70.58± 2.08	88.14± 0.83	57.00± 0.54

Table 9: Results comparing LAIT configurations. We perform a split of the validation sets to form synthetic validation and test sets; we report the test-set score corresponding to the checkpoint with the best validation performance.

Task	Full Encoding ↓	with Caching ↓
MNLI	83.33%	69.85%
STSB	63.07%	62.78%
AnswerEq	54.94%	41.21%
BoolQ	89.72%	83.53%
BoolQ-S	48.69%	47.12%
WiC	55.85%	55.85%
FEVER	78.19%	47.74%
VitaminC	80.42%	70.67%
MultiRC	94.93%	59.02%
RTE	82.49%	82.04%
QQP	64.05%	61.85%
Potential practical settings:		
ContractNLI	99.46%	60.75%
WikiClusters	81.51%	58.47%

Table 10: Cost of encoder attention FLOPs (vs. `T5-base`) when using LAIT⋆ model for each task to process the entire validation set. LAIT⋆ selection is based on results in tables 9 and 11.

Model	AnswerEq Accuracy	BoolQ-split Accuracy	MultiRC F1
LAIT-0	90.55± 1.21	81.22± 1.90	78.55 ± 1.94
LAIT-1	90.73± 1.17	81.77± 1.87	78.13 ± 1.91
LAIT-2	89.65± 1.19	81.16± 1.90	78.81 ± 1.89
LAIT-3	90.69± 1.19	81.04± 1.96	**77.97** ± 1.9
LAIT-4	90.55± 1.22	**81.65± 1.83**	76.98 ± 2.15
LAIT-5	90.46± 1.21	79.27± 1.90	77.48 ± 1.97
LAIT-6	90.37± 1.19	80.31± 1.96	75.6 ± 2.12
LAIT-7	90.51± 1.22	79.45± 1.96	72.87 ± 1.99
LAIT-8	89.74± 1.26	79.76± 1.99	71.58 ± 1.97
LAIT-9	**91.18± 1.15**	79.02± 2.02	72.03 ± 2.04
LAIT-10	86.68± 1.44	71.62± 2.11	59.29 ± 2.46
LAIT-11	60.50± 1.91	70.15± 2.20	61.47 ± 2.13
LAIT-12	61.40± 2.02	70.09± 2.11	60.11 ± 2.34

Table 11: Results for tasks with more than two segments. **Bold**, underline, and box indicate model performance as in Table 9.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
We discuss limitations in section 5.2, as well as an un-numbered final "Limitations" section.

☑ A2. Did you discuss any potential risks of your work?
Yes, in the final un-numbered "Limitations" section.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract is un-numbered; main claims are in sections: 1, 3, 3.1, 5, 5.1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Datasets and pretrained models. Section 4.1, Section 4.2, Section 5.2

☑ B1. Did you cite the creators of artifacts you used?
Section 4.1, Section 4.2, Section 5.2

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
All datasets and checkpoints are public; we cite the relevant papers and repositories in 4.1 and 4.2 and 5.2 for more details.

☒ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
We use common benchmarks and model checkpoints. We believe that our use of these data clearly falls within the intended use of those resources.

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
The datasets we use are standards of the field for model evaluation; we do not believe our work introduces any new concerns regarding offensiveness, identification, or anonymization.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Section 4.2

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Table 1

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

C ☑ Did you run computational experiments?

Section 4, Section 5

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 4.1, Section 5.1, Section 5.2

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
In section 4.1 we describe the search, the evaluation strategy (select by either best performance on validation or on a synthetic dev/test split), the models used, and the tasks used.

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 5

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 4.1

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.