CHAIN2PARALLEL: POST-HOC PARALLELIZATION OF T5 CLASSIFIER CHAINS FOR FEW-SHOT MULTI-LABEL CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-label text classification (MLTC) aims to assign a set of correlated labels to each document, a task that becomes particularly challenging when labeled data is scarce. Autoregressive classifier chains model these dependencies effectively, but their sequential label decoding incurs substantial inference cost. We propose CHAIN2PARALLEL, a post-hoc parallelization method that transforms a completed, task-specialized T5 CHAIN into an efficient, single-pass classifier. By leveraging the frozen label-conditioned encoder and extracting its label-aligned sentinel representations, CHAIN2PARALLEL predicts all labels in parallel via a lightweight shared classification head. Across three few-shot benchmarks, CHAIN2PARALLEL matches or improves upon the original chain's classification performance while substantially reducing inference latency, memory consumption, and model footprint.

1 INTRODUCTION

The goal of multi-label text classification (MLTC) is to annotate text documents with arbitrary combinations of relevant labels (Zhang & Zhou, 2014; Liu et al., 2020). In few-shot settings, the challenge is not only to recognize each label from limited supervision, but also to model dependencies among labels. Classifier chains address the latter by factorizing the joint label distribution into a sequence of conditional decisions (Read et al., 2011; 2021). Neural encoder–decoder models make this formulation expressive, but bind prediction to a serial decoding process whose cost grows with the label sequence.

Nguyen et al. (2025) apply this idea to T5 (Raffel et al., 2020). Their T5 CHAIN combines documents with textual label descriptions and learns a classifier chain under a task-level budget of $8|L|$ labeled samples. While setting a competitive few-shot benchmark, it still relies on full autoregressive decoding for every label decision. Existing non-autoregressive T5 adaptations and parallel MLTC systems instead learn a parallel predictor from the outset (Liu et al., 2021; Kementchedjhieva & Chalkidis, 2023). Crucially, it remains unclear whether a completed chain can switch from sequential to parallel prediction without discarding representations acquired under the chain objective.

Our central question is whether autoregressive chains can serve purely as training-time representation learners without imposing autoregressive inference at deployment. Specifically, we examine whether their label-conditioned representations support simultaneous label prediction, and characterize the classification and co-occurrence structure encoded within the frozen encoder states used by the parallel readout. To verify, we evaluate this conversion across classification accuracy, representation quality, inference latency and deployment cost.

We answer positively the above question by introducing CHAIN2PARALLEL, a post-hoc parallelization method for a completed T5 CHAIN. It preserves the specialized encoder and exposes its label-aligned sentinel states to a lightweight Shared MLP that scores all labels simultaneously. The encoder remains frozen, and the readout is learned from gold multi-label targets. The converted model thus leverages the label-conditioned encoder representations learned through autoregressive chain training, while predicting all labels simultaneously at deployment. We compare CHAIN2PARALLEL alongside T5 CHAIN, SetFit, and other few-shot baselines on EURLEX (Chalkidis et al., 2019), AAPD (Yang et al., 2018), and Reuters (Apté et al., 1994). CHAIN2PARALLEL matches T5 CHAIN's multi-label

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

classification performance while exceeding the reported results of the other strong baselines. On the long-tailed Reuters dataset, it achieves a marked improvement over T5 CHAIN. In addition, we train two direct-parallel baselines, DIRECTPARALLEL-BASE and DIRECTPARALLEL-GP, derived from the same T5-v1.1-base backbone. They however are obviously inferior to T5 CHAIN, let alone our method CHAIN2PARALLEL. These results show that (i) it is non-trivial to achieve a competitive performance for few-shot multi-label settings; and (ii) task-specific few-shot chain training is helpful to classification performance. Besides the gains above, deployment measurements also show roughly halved *batch-one memory* and a substantially smaller *inference bundle size*. In summary, our contributions are threefold:

- **Decoupling chain learning from serial inference.** We present CHAIN2PARALLEL, a post-hoc conversion of T5 CHAIN that separates autoregressive representation learning from parallel prediction. It trains a shared head on frozen, label-aligned sentinel states and removes the decoder from deployment. To the best of our knowledge, this is the first work to couple classifier-chain training with parallel readout for few-shot MLTC.
- **Improving the accuracy–efficiency trade-off.** Across three benchmarks and five splits, CHAIN2PARALLEL preserves aggregate classification performance on EURLEX and AAPD, and achieves a marked improvement on Reuters. Our deployment shows $50.9\times\sim 214.5\times$ faster batch-one end-to-end inference, $23.5\times\sim169.6\times$ higher throughput, and 55.7% fewer deployed parameters.
- **Insights into frozen chain-learned representations.** Matched ablations pinpoint label-aligned sentinel states as the key design factor, and diagnostics uncover shared cross-label classification directions and dataset-dependent residual co-occurrence structure.

2 RELATED WORK

Existing classification paradigms. Few-shot MLTC extends multi-label text classification to settings with limited labeled samples, making it difficult to learn both reliable label-specific decisions and dependencies among labels. Existing approaches include direct fine-tuning with encoder-based classification heads (Sun et al., 2019), prompt-based methods such as PET and PLAML (Schick & Schütze, 2021; Thaminkaew et al., 2024), contrastive methods such as SetFit (Tunstall et al., 2022), and label-wise attention models such as BERT-LWAN and LSAN (Chalkidis et al., 2020; Xiao et al., 2019). However, most of these approaches either treat labels as independent decisions or encode their dependencies only indirectly, without modeling the joint output distribution. LLM-based approaches also perform multi-label prediction (Zhu & Zamani, 2024; Xu et al., 2026). They however introduce substantial inference and context-processing costs, and remain sensitive to prompt construction, exemplar selection, and demonstration order (Zhao et al., 2021; Lu et al., 2022).

Autoregressive chains and few-shot MLTC. Classifier chains factorize the joint label distribution into an ordered sequence of conditional predictions, allowing earlier decisions to inform later ones (Read et al., 2011; 2021). Neural sequence-generation models, such as SGM, extend this principle to encoder–decoder architectures (Yang et al., 2018). T5 CHAIN further combines T5 with textual label descriptions, sentinel anchors, order-shuffled adaptation, and chain specialization for few-shot MLTC (Nguyen et al., 2025). Although these approaches capture inter-label dependencies, their predictions remain order-dependent and serial (Nam et al., 2019), and they often suffer from poor memory footprint and high inference latency. Our work uses the representations learned through chain training to predict all labels in parallel at deployment.

Non-autoregressive and parallel multi-label prediction. Parallel and non-autoregressive methods avoid sequential label generation through concurrent classifier construction (Moyano et al., 2019), label-graph message passing (Lanchantin et al., 2020), latent-variable prediction (Barezi et al., 2020), or direct set decoding (Du et al., 2024). T5-based alternatives similarly support parallel prediction: EncT5 introduces learned latent queries (Liu et al., 2021), while T5Enc maps fixed label identifiers to label-specific binary heads (Kementchedjhieva & Chalkidis, 2023). Particularly, these methods train a parallel predictor from the outset. In contrast, CHAIN2PARALLEL starts from a completed, task-specialized T5 CHAIN and reuses its chain-trained, label-conditioned representations to predict all labels simultaneously at deployment without autoregressive decoding.

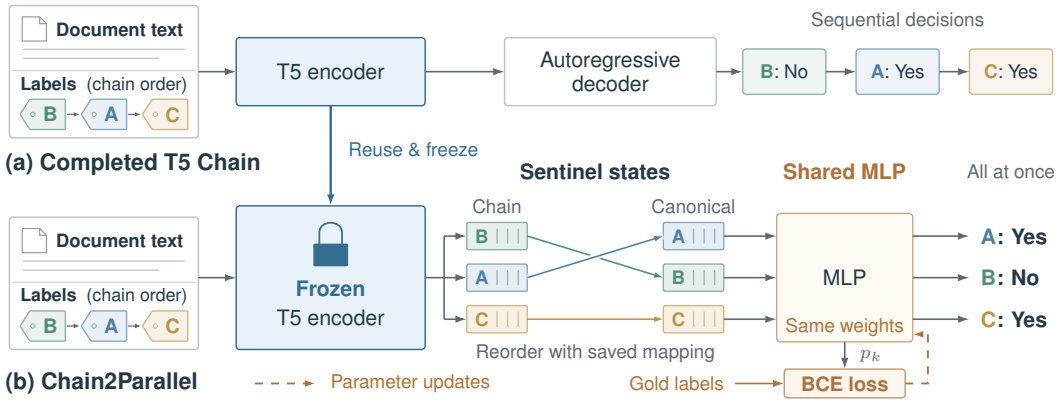


Figure 1: Post-hoc conversion from T5 CHAIN to CHAIN2PARALLEL.

3 CHAIN2PARALLEL

3.1 OVERVIEW

CHAIN2PARALLEL decouples representations learned through dependency-aware classifier-chain training from the serial prediction mechanism. Figure 1 presents the complete "conversion" workflow, where the upper part is the pipeline of T5 CHAIN. We begin with a completed T5 CHAIN and retain its tokenizer, selected label order, input format, and task-specialized encoder. However, we discard the autoregressive decoder. Given a document, the frozen encoder processes the document together with all label descriptions in a single pass. We collect the hidden state at each label's sentinel position, and use the saved chain-to-label mapping to restore the states to the canonical label order (see the middle part of this figure). Then, we replace the autoregressive decoder of T5 CHAIN with a Shared MLP trained on gold multi-label targets. The Shared MLP maps all aligned sentinel states to their corresponding label probabilities in parallel, rather than predicting the labels sequentially. Finally, we determine the predicted labels based on a fixed threshold. For ease of presentation, in what follows we first introduce T5 CHAIN as it is the footstone of our solution, and then cover other main components, respectively (Sections 3.2∼3.5).

3.2 T5 CHAIN FORMULATION

T5 CHAIN serves as the starting point of our method. It is an encoder-decoder model. The source model is initialized from T5-v1.1-base. Its encoder jointly encodes the document and an ordered set of label descriptions. Conditioned on these encoder representations and its preceding decoder outputs, the decoder then generates one binary decision for each label in sequence. We next formalize its input–output structure and training objective. Let $L = \{l_1, \ldots, l_m\}$ be the canonical label set, x the input text, and $y \in \{0,1\}^m$ its multi-label indicator vector, where $y_k = 1$ exactly when l_k is relevant to x. Let $\mathfrak{S}_m$ denote all permutations of $\{1, \ldots, m\}$; $\pi \in \mathfrak{S}_m$ maps chain position j to canonical label index $\pi(j)$. The source-chain input is

$$z^{\pi} = x\,[\mathrm{E}_1]\,l_{\pi(1)}\,[\mathrm{E}_2]\,l_{\pi(2)} \cdots [\mathrm{E}_m]\,l_{\pi(m)}. \tag{1}$$

Here each l_j is represented by its textual label description, and the sentinel tokens provide explicit, uniquely identifiable anchors for the label positions.

The output sequence v^{π} alternates sentinel markers and YES/NO decisions, with $v^{\pi}_{2j-1} = [\mathrm{E}_j]$ and v^{π}_{2j} equal to YES if $y_{\pi(j)} = 1$ and NO otherwise. With this sequence-to-sequence formulation established, we train the source model via a three-stage pipeline: generalist pre-training (GP), order-marginal domain adaptation (DA), and chain specialization (CS). After DA, chain selection retains a label order $\pi^{\star}$ on validation data. CS then minimizes the original fixed-order objective:

$$\mathcal{L}^{\mathrm{CS}} = -\mathbb{E}_{(x,y)} \sum_{j=1}^{m} \log q_{\theta}(v^{\pi^{\star}}_{2j} \mid z^{\pi^{\star}}, v^{\pi^{\star}}_{<2j}). \tag{2}$$

Here θ denotes the source encoder–decoder parameters, and $\mathbb{E}_{(x,y)}$ averages over labeled training examples. The distribution q_θ is obtained by normalizing the decoder's YES/NO logits over those two tokens; $v^{\pi^\star}_{<2j}$ denotes the output prefix preceding position $2j$. During training, the model uses the gold prefix, whereas inference conditions on previously generated decisions and their sentinel markers. At inference, the corresponding YES probability is the relevance score for a label.

The T5 CHAIN encoder computes contextualized states for all label positions in a single pass. Our key idea is to reuse the encoder of T5 CHAIN while replacing the autoregressive prediction.

3.3 ENCODER EXPORT AND SENTINEL READOUT

Following the original T5 CHAIN procedure, we first train the source model through the same three-stage pipeline described in Section 3.2. Then, we freeze this encoder and convert the encoder's token-level hidden states into aligned representations for each label. Here we adopt sentinel states, because each sentinel anchors a specific label position, and meanwhile encoder self-attention enables the sentinel's state to incorporate information from both the document and all label descriptions. This conversion involves encoding the original label-conditioned input, extracting the sentinel states, and reordering them into the canonical label order.

For the selected-chain input $z^{\pi^\star}$, the encoder returns hidden states $H = \mathrm{Enc}(z^{\pi^\star}) \in \mathbb{R}^{S\times d}$. Here Enc is the exported encoder, S is the tokenized input length including any padding, and d is the encoder hidden dimension; $H_s \in \mathbb{R}^d$ denotes the state at token position s. Each sentinel token is required to occur exactly once. If $s_j \in \{1,\dots,S\}$ is the input position of $[\mathrm{E}_j]$, we extract one state for chain position j:

$$h_j = H_{s_j} \in \mathbb{R}^d. \tag{3}$$

These states are initially sorted by chain position. Using the chain-to-label mapping stored during chain selection, we deterministically reorder these states along the canonical label axis before classification:

$$\widetilde{h}_k = h_{(\pi^\star)^{-1}(k)}, \qquad k = 1,\dots,m. \tag{4}$$

The inverse permutation $(\pi^\star)^{-1}(k)$ gives the chain position of canonical label l_k, so $\widetilde{h}_k \in \mathbb{R}^d$ is its aligned state. Stacking these states as rows yields $\widetilde{H} = [\widetilde{h}_1,\dots,\widetilde{h}_m]^\top \in \mathbb{R}^{m\times d}$. This operation retains information from the selected label-conditioned input and produces one state per label to enable parallel computation.

3.4 SHARED PARALLEL READOUT

The aligned matrix $\widetilde{H}$ obtained contains one frozen encoder state for each canonical label. Although the encoder remains frozen during conversion, these states were shaped by the source chain objective, as decoder loss was backpropagated into the encoder during chain training. Encoder self-attention also jointly contextualizes the document and all label descriptions. CHAIN2PARALLEL therefore treats each $\widetilde{h}_k$ as a fixed, label-conditioned feature and learns only a lightweight parallel readout.

The readout applies the same two-layer MLP to every aligned state. Specifically, for label k we first compute $r_k = \mathrm{LN}(\widetilde{h}_k) \in \mathbb{R}^d$, where $\mathrm{LN}(\cdot)$ denotes layer normalization, and let D_p denote dropout with probability p during training. At inference, dropout is disabled. The relevance probability for label l_k is then given by:

$$p_k = \sigma\big(w_2^\top D_p(\mathrm{GELU}(W_1 r_k + b_1)) + b_2\big)\,. \tag{5}$$

Here $W_1 \in \mathbb{R}^{d_{\mathrm{MLP}}\times d}$ and $w_2 \in \mathbb{R}^{d_{\mathrm{MLP}}}$ are weights, d_{MLP} refers to MLP hidden width, $b_1 \in \mathbb{R}^{d_{\mathrm{MLP}}}$ and $b_2 \in \mathbb{R}$ are biases, and $\top$ denotes transpose; $\mathrm{GELU}(\cdot)$ refers to the element-wise Gaussian error linear unit, and $\sigma(\cdot)$ is the logistic sigmoid. The same set of parameters is broadcast across labels, while the identity of each label is encoded within $\widetilde{h}_k$. This parameterization pools supervision into a shared evidence-to-logit function under a limited training budget. Its parameter count is independent of m, whereas the label-specific MLP weights grow as $O(md\,d_{\mathrm{MLP}})$; here $O(\cdot)$ denotes asymptotic complexity. Accordingly, we fix the Shared MLP as the primary readout prior to formal evaluation. This design distributes limited supervision across all labels, maintains a fixed head size independent of m, and enables a unified deployment configuration across different datasets.

During readout training, the encoder remains frozen. Only the head parameters ϕ—the MLP weights and biases and the layer-normalization scale and bias vectors in $\mathbb{R}^d$—are optimized. For sample x_i and label l_k, let $y_{ik} \in \{0, 1\}$ be the gold target and p_{ik} its predicted probability. Over N training samples, we minimize mean binary cross-entropy:

$$\mathcal{L}_{\text{head}}(\phi) = \frac{1}{Nm} \sum_{i=1}^{N} \sum_{k=1}^{m} \text{BCE}(y_{ik}, p_{ik}), \tag{6}$$

3.5 Inference and Deployment

At deployment, CHAIN2PARALLEL relies on the tokenizer, exported encoder, trained shared readout head, and stored chain-to-label mapping. CHAIN2PARALLEL runs the exported label-conditioned encoder once, extracts and reorders the sentinel states, and then evaluates all m head applications in parallel. No label-level autoregressive generation takes place during inference. This conversion trades label-count-dependent autoregressive depth for m parallel readouts while keeping the label-conditioned encoder computation. The encoder sequence encodes all label descriptions, and the parallel head performs m readouts, so both computations continue to scale with the label set.

4 Experiments

4.1 Experimental Setup

Datasets and evaluation metrics. We evaluate on EURLEX (Chalkidis et al., 2019), AAPD (Yang et al., 2018), and Reuters (Apté et al., 1994) using the few-shot protocol of Nguyen et al. (2025). We report micro F1 (mF1), macro F1 (MF1), and instance F1 (IF1), which respectively emphasize aggregate label decisions, performance across labels, and per-document label-set quality. Following Nguyen et al. (2025), we additionally report their arithmetic mean, AvgF1, as a summary measure.

Baselines. Our current evaluation compares T5 CHAIN, CHAIN2PARALLEL, DIRECTPARALLEL-BASE, and DIRECTPARALLEL-GP on the same five splits. For broader context, we also incorporate the few-shot results reported by Nguyen et al. (2025) for Head with RoBERTa or the T5 encoder, PET, SetFit, BERT-LWAN, and their original T5 CHAIN implementation. For a fair comparison, these systems use the same three tasks, $8|L|$ training budget, official test sets, metrics, and fixed threshold. In the following tables we report the mean with the standard deviation (SD) as a subscript.

Training and model selection. The source T5 CHAIN model uses a T5-v1.1-base backbone and follows the GP, DA, and CS pipeline. CHAIN2PARALLEL freezes the specialized encoder and trains a shared readout with gold-label binary cross-entropy loss. The Shared MLP was designated as the primary readout before official-test evaluation. DIRECTPARALLEL-BASE and DIRECTPARALLEL-GP share an identical two-stage parallel training pipeline and final topology, they differ only in initialization. (Further details are provided in Appendix A.)

4.2 Main Results

Table 1 reports comparison results. CHAIN2PARALLEL exceeds the five published non-chain baselines in every dataset–metric pair at the level of reported means. All paired statistics below use only current methods evaluated on the same splits, and we adopt the final T5 CHAIN model as the reference baseline, unless stated otherwise.

On EURLEX and AAPD, CHAIN2PARALLEL preserves aggregate performance (e.g., AvgF1 69.0 of CHAIN2PARALLEL vs. AvgF1 68.7 of T5 CHAIN on EURLEX). Notably, on Reuters it achieves a large gain: AvgF1 increases from 71.2 to 79.3, and mF1, MF1, and IF1 improve by 4.5, 16.8, and 3.1 points. This set of results actually demonstrates the advantages of CHAIN2PARALLEL.

To examine whether changes in aggregate performance are widespread across labels or driven by a small subset, Figure 2 decomposes the aggregate scores into matched per-label F1. We can see that EURLEX and AAPD lie closely along the line of equality, with Win/Tie/Loss (W/T/L) counts of 13/0/8 and 23/6/24, and MF1 changes of $+0.7$ and $+0.2$ points, respectively. On Reuters, 70 of 90 labels improve in their five-split mean, yielding the $+16.8$-point MF1 change. These results suggest that the improvement is broad at the label-mean level, rather than limited to a small subset of labels.

Table 1: Few-shot performance on the official test sets. The two provenance blocks (Nguyen et al. (2025) vs. Current evaluation) use different sampled splits and are not paired. Bold marks the highest displayed mean within each block, with ties preferentially highlighted for CHAIN2PARALLEL.

Provenance	Method	mF1	MF1	IF1	AvgF1
EURLEX					
Nguyen et al. (2025)	Head (RoBERTa)	$70.0_{0.8}$	$34.8_{1.2}$	$69.2_{1.0}$	58.0
	Head (T5 Encoder)	$47.9_{0.9}$	$11.9_{0.3}$	$47.7_{0.6}$	35.8
	PET (RoBERTa)	$70.8_{0.6}$	$45.8_{2.6}$	$70.9_{0.7}$	62.5
	SetFit	$69.9_{0.9}$	$35.7_{2.6}$	$70.6_{0.8}$	58.7
	BERT-LWAN	$69.0_{1.4}$	$35.0_{1.8}$	$68.4_{1.5}$	57.5
	T5 CHAIN (reported)	$\mathbf{73.3_{0.2}}$	$\mathbf{56.4_{0.4}}$	$\mathbf{73.8_{0.2}}$	**67.8**
Current evaluation	T5 CHAIN	$75.0_{0.8}$	$56.0_{2.2}$	$75.2_{0.9}$	$68.7_{1.0}$
	DIRECTPARALLEL-BASE	$73.6_{0.5}$	$52.5_{0.9}$	$73.9_{0.5}$	$66.7_{0.5}$
	DIRECTPARALLEL-GP	$72.8_{2.0}$	$52.4_{3.6}$	$73.1_{2.1}$	$66.1_{2.5}$
	CHAIN2PARALLEL	$\mathbf{75.0_{0.2}}$	$\mathbf{56.7_{1.2}}$	$\mathbf{75.2_{0.3}}$	$\mathbf{69.0_{0.6}}$
AAPD					
Nguyen et al. (2025)	Head (RoBERTa)	$41.5_{1.1}$	$7.0_{0.6}$	$32.0_{0.9}$	26.8
	Head (T5 Encoder)	$39.9_{2.0}$	$5.7_{1.0}$	$30.3_{2.1}$	25.3
	PET (RoBERTa)	$52.0_{1.2}$	$22.1_{2.4}$	$45.2_{2.1}$	39.8
	SetFit	$58.9_{0.7}$	$30.9_{1.4}$	$58.2_{1.0}$	49.3
	BERT-LWAN	$45.5_{3.8}$	$11.9_{5.5}$	$37.8_{5.7}$	31.7
	T5 CHAIN (reported)	$\mathbf{60.3_{0.4}}$	$\mathbf{42.6_{0.6}}$	$\mathbf{61.1_{0.5}}$	**54.7**
Current evaluation	T5 CHAIN	$61.7_{0.7}$	$40.6_{1.0}$	$\mathbf{61.9_{0.9}}$	$54.7_{0.8}$
	DIRECTPARALLEL-BASE	$52.6_{20.0}$	$30.8_{16.4}$	$50.0_{21.1}$	$44.5_{19.1}$
	DIRECTPARALLEL-GP	$61.0_{1.3}$	$39.4_{3.1}$	$59.1_{2.4}$	$53.2_{2.2}$
	CHAIN2PARALLEL	$\mathbf{62.3_{0.6}}$	$\mathbf{40.8_{1.3}}$	$61.0_{0.9}$	$\mathbf{54.7_{0.8}}$
Reuters					
Nguyen et al. (2025)	Head (RoBERTa)	$55.9_{4.7}$	$2.1_{0.3}$	$48.0_{5.4}$	35.3
	Head (T5 Encoder)	$61.7_{0.7}$	$56.6_{1.0}$	$56.3_{0.9}$	58.2
	PET (RoBERTa)	$\mathbf{85.6_{0.8}}$	$48.6_{1.2}$	$87.6_{1.1}$	73.9
	SetFit	$84.8_{0.9}$	$36.8_{2.1}$	$88.5_{0.6}$	70.0
	BERT-LWAN	$74.6_{4.1}$	$11.6_{7.1}$	$73.4_{5.5}$	53.2
	T5 CHAIN (reported)	$84.8_{0.8}$	$\mathbf{57.7_{2.8}}$	$\mathbf{88.9_{0.8}}$	**77.1**
Current evaluation	T5 CHAIN	$83.2_{1.0}$	$43.8_{1.7}$	$86.5_{1.0}$	$71.2_{0.8}$
	DIRECTPARALLEL-BASE	$64.2_{13.2}$	$17.7_{22.9}$	$62.4_{15.7}$	$48.1_{17.3}$
	DIRECTPARALLEL-GP	$87.4_{1.1}$	$\mathbf{62.1_{2.3}}$	$89.3_{1.3}$	$\mathbf{79.6_{0.8}}$
	CHAIN2PARALLEL	$\mathbf{87.7_{0.5}}$	$60.6_{3.7}$	$\mathbf{89.6_{0.6}}$	$79.3_{1.5}$

To examine the roles of *generalist pretraining* and subsequent *task-specific chain training*, we compare two direct-parallel baselines. We can see from Table 1 that on EURLEX, DIRECTPARALLEL-BASE and DIRECTPARALLEL-GP are close at 66.7 and 66.1 AvgF1. On AAPD, GP initialization raises the mean from 44.5 to 53.2; on Reuters, it raises AvgF1 from 48.1 to 79.6. these results show that *generalist pretraining* provides the positive feedback for learning the parallel encoder. On the other hand, CHAIN2PARALLEL achieves higher AvgF1 than DIRECTPARALLEL-GP on EURLEX and AAPD. The gains demonstrate the value of *task-specific chain training* for learning encoder representations that better support parallel prediction. Furthermore, we also observe that the two reach the same level on Reuters. This could be that Reuters's long-tailed labels and sparse co-occurrence leave less repeated evidence for learning further dependencies beyond GP.

4.3 ABLATION ANALYSIS

To identify the critical readout design choices, we perform ablations across representation, parameter sharing, and readout architecture. Table 2 reports the results.

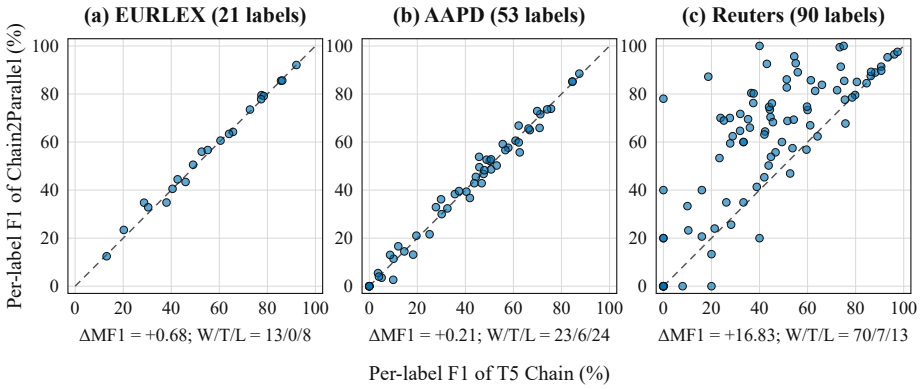


Figure 2: Each point is one canonical label; the dashed line marks equality. W/T/L counts label-wise wins, ties, and losses for CHAIN2PARALLEL under $p > 0.5$.

Table 2: Official-test AvgF1 across five few-shot splits. Each adjacent pair compares (a) representations, (b) parameter sharing, or (c) readout architecture with other settings fixed. Bold marks the higher mean within each pair. Global denotes Conditioned Global; LS denotes Label-Specific.

(a) Representation *Fixed: label-specific heads*

Head	Global	Sentinel
	EURLEX	
Linear	$56.42_{1.74}$	$\mathbf{68.92_{0.91}}$
MLP	$58.72_{0.41}$	$\mathbf{68.98_{0.74}}$
	AAPD	
Linear	$41.39_{1.19}$	$\mathbf{54.51_{0.90}}$
MLP	$44.05_{1.15}$	$\mathbf{54.04_{1.24}}$
	Reuters	
Linear	$60.26_{1.38}$	$\mathbf{75.86_{1.11}}$
MLP	$61.88_{1.14}$	$\mathbf{76.05_{0.78}}$

(b) Parameter sharing *Fixed: Sentinel states*

Head	LS	Shared
	EURLEX	
Linear	$\mathbf{68.92_{0.91}}$	$68.66_{0.82}$
MLP	$\mathbf{68.98_{0.74}}$	$68.96_{0.57}$
	AAPD	
Linear	$\mathbf{54.51_{0.90}}$	$53.99_{1.36}$
MLP	$54.04_{1.24}$	$\mathbf{54.71_{0.76}}$
	Reuters	
Linear	$75.86_{1.11}$	$\mathbf{77.54_{1.09}}$
MLP	$76.05_{0.78}$	$\mathbf{79.34_{1.49}}$

(c) Readout Architecture *Fixed: Sentinel states*

Sharing	Linear	MLP
	EURLEX	
Shared	$68.66_{0.82}$	$\mathbf{68.96_{0.57}}$
LS	$68.92_{0.91}$	$\mathbf{68.98_{0.74}}$
	AAPD	
Shared	$53.99_{1.36}$	$\mathbf{54.71_{0.76}}$
LS	$\mathbf{54.51_{0.90}}$	$54.04_{1.24}$
	Reuters	
Shared	$77.54_{1.09}$	$\mathbf{79.34_{1.49}}$
LS	$75.86_{1.11}$	$\mathbf{76.05_{0.78}}$

Effectiveness of Readout Representation. The conditioned global representation averages all encoder states — including document representations, label descriptions, sentinel tokens, and the end-of-sequence token — to form one single vector. This vector is then fed to every label, meaning label distinctions must be learned entirely by the label-specific heads. In contrast, Sentinel provides each label l_k with its aligned state $\widetilde{h}_k$. This representation incorporates document evidence and all label descriptions via bidirectional self-attention while staying anchored to its corresponding label. Table 2(a) compares these representations under matched Label-Specific Linear and MLP heads. Replacing Conditioned Global representation with Sentinel improves AvgF1 by 12.5 and 10.3 points on EURLEX, 13.1 and 10.0 points on AAPD, and 15.6 and 14.2 points on Reuters for Linear and MLP, respectively. Since the encoder input and classification-head family are held fixed, these consistent gains actually demonstrate the value of retaining one label-aligned representation per label rather than compressing the entire encoder output into a shared global mean.

Effectiveness of Parameter Sharing. A Shared head applies the same function f_ϕ to every label-aligned state $\widetilde{h}_k$. This does not produce identical label scores because the Sentinel states are already conditioned on different labels; instead, it allows supervision from all labels to update one common scoring function. A Label-Specific head assigns an independent function f_{ϕ_k} to each label, providing greater label-specific capacity but restricting each head to the limited supervision available for that label. Table 2(b) compares these schemes while holding the Sentinel representation and readout architecture fixed. Parameter sharing increases mean AvgF1 on Reuters by 1.7 and 3.3 points for Linear and MLP heads, respectively. On EURLEX, both shared heads score slightly lower; on

AAPD, sharing lowers Linear performance but improves MLP performance. On Reuters, the few-shot splits leave an average of 24.2 of the 90 labels with no positive training samples. For such labels, a Label-Specific head receives supervision only from negative examples, whereas a Shared head can also learn from positive examples of other labels. All these results demontrate that parameter sharing becomes particularly valuable especially when label-level supervision is incomplete.

Effectiveness of Readout Architecture. The two readout architectures differ in how they transform an encoder representation into a label score. A Linear head applies a single affine projection, providing a direct linear decision function over the Sentinel state. An MLP instead uses a two-layer architecture and GELU, allowing nonlinear combinations of the features contained in that state. Table 2(c) compares Linear and MLP readouts while holding both the Sentinel representation and parameter-sharing scheme fixed. MLP has a modest empirical edge at the level of five-split means. Within Shared heads, the margins between MLP and Linear heads are $+0.3$, $+0.7$, and $+1.8$ points on EURLEX, AAPD, and Reuters. Within Label-Specific heads, the differences are $+0.1$, -0.5, and $+0.2$. MLP is therefore higher in five of the six matched comparisons, although most gains are small. These ablation results directly validate our a priori design choice of the shared MLP: When using unrounded mean values, it ranks second on EURLEX and first on both AAPD and Reuters. It remains within 0.1 $\mathrm{AvgF1}$ points of the best-performing head on EURLEX and outperforms the runner-up by 0.2 and 1.8 points on AAPD and Reuters. Since sentinel states are already label-conditioned, a shared mapping aggregates limited supervision across all labels and avoids the linear growth of label-specific MLP heads with respect to label count. This design serves as a compact, consistently competitive default option.

4.4 Representations Supporting Parallel Readout

To understand why a decoder-free parallel head can reuse a frozen encoder, we examine two properties of the canonical sentinel states before head normalization. First, we test whether positive–negative classification directions are stable across training and validation and aligned across labels. Second, we investigate whether document-wise variation in these states reflects label co-occurrence. More specifically, we examine whether label-related structure exists within the chain-trained encoder.

To answer the first question, we conduct the followings: for each label we compute the difference between its mean positive- and negative-sample states. Figure 3(a) presents two comparisons. Circles compare the same label's classification direction between training and validation, yielding mean cosine similarities of 0.968, 0.931, and 0.962 on EURLEX, AAPD, and Reuters, respectively. Squares compare directions across labels within training, yielding mean pairwise cosines of 0.848, 0.743, and 0.766. This directional alignment supports a shared evidence-to-logit function: label identity is already embedded within the conditioned sentinel states, allowing the readout head to pool supervision across categories along a unified relevance direction.

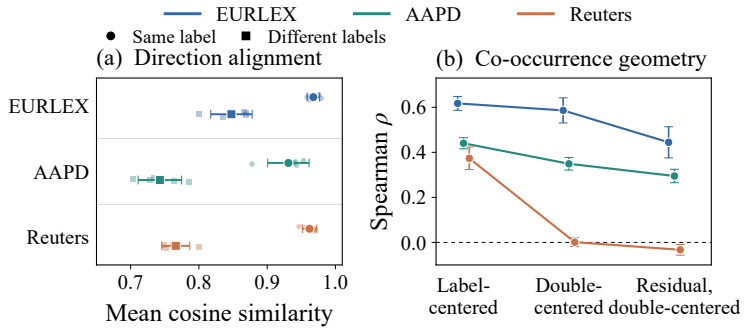


Figure 3: Sentinel geometry across five data splits. (a) Direction alignment for the same label across training and validation (circles) and between labels in training (squares). (b) Co-occurrence alignment under successive representation controls.

To answer the second question, we conduct a test by comparing the pairwise similarity of label-state variation with training-set label co-occurrence. Figure 3(b) reports this alignment under successive controls: label centering removes each label's mean state, double centering additionally removes the document-level component shared across labels, and the residual variant further removes each label's positive–negative class-mean component. The correlations after double centering are 0.586, 0.349,

Table 3: Five-split means and sample SDs as subscripts. Full-validation throughput uses batch size 2; bundle size includes pure-inference files only.

(a) Batch-one end-to-end latency

Dataset	Chain (ms)	C2P (ms)	Speedup
EURLEX	$1065.5_{5.3}$	$\mathbf{21.0_{0.3}}$	$\mathbf{50.9_{0.6}}\times$
AAPD	$2598.7_{36.4}$	$\mathbf{20.4_{0.5}}$	$\mathbf{127.7_{2.5}}\times$
Reuters	$4337.8_{25.2}$	$\mathbf{20.2_{0.1}}$	$\mathbf{214.5_{1.7}}\times$

(b) Full-validation throughput

Dataset	Chain (ex/s)	C2P (ex/s)	Speedup
EURLEX	$1.74_{0.01}$	$\mathbf{40.9_{4.1}}$	$\mathbf{23.5_{2.2}}\times$
AAPD	$0.738_{0.004}$	$\mathbf{82.6_{2.5}}$	$\mathbf{111.9_{3.1}}\times$
Reuters	$0.444_{0.002}$	$\mathbf{75.2_{1.7}}$	$\mathbf{169.6_{3.8}}\times$

(c) Batch-one peak allocated memory

Dataset	Chain (MiB)	C2P (MiB)
EURLEX	$1063.9_{8.7}$	$\mathbf{536.6_{9.2}}$
AAPD	$1054.2_{4.9}$	$\mathbf{490.8_{5.2}}$
Reuters	$1073.7_{4.9}$	$\mathbf{492.9_{7.7}}$

(d) Model footprint

Metric	Chain	C2P	Reduction
Parameters (M)	247.6	**109.7**	55.7%
Bundle (MiB)	945.2	**419.4**	55.6%

and 0.001 on EURLEX, AAPD, and Reuters, and become 0.445, 0.295, and -0.033 after the final control. The remaining positive correlations on EURLEX and AAPD indicate co-occurrence-aligned variation beyond the removed components, whereas Reuters shows little residual alignment.

Putting it all together, these results show that the frozen, chain-specialized encoder provides broadly aligned classification directions across labels and, on EURLEX and AAPD, document-dependent variation aligned with label co-occurrence. These properties offer a representation-level explanation for why its sentinel states are suitable for a shared parallel readout.

4.5 Deployment Efficiency

From Table 3(a) we can see that batch-one end-to-end inference is $50.9\times \sim 214.5\times$ faster across the three datasets; in addition, Table 3(b) shows $23.5\times \sim 169.6\times$ higher throughput over complete validation passes. This acceleration mainly stems from replacing autoregressive decoding along the label chain with a non-autoregressive shared MLP readout, which can predict all labels in parallel using their label-conditioned sentinel representations.

Removing the decoder also reduces the number of deployed parameters and the associated computation and memory overhead. With the converted inference path, batch-one allocated memory drops by approximately 50% (Table 3(c)), while parameters and pure-inference bundle size fall by 55.7% and 55.6% (Table 3(d)). These results further demonstrate the superiority of our method.

5 Conclusion

Chain2Parallel separates dependency-aware chain training from the prediction structure used at deployment. The conversion retains the specialized T5 encoder and reads its label-aligned sentinel representations in parallel. Under the $8|L|$ few-shot budget, Chain2Parallel preserves aggregate performance on EURLEX and AAPD and improves all reported F1 measures on Reuters. Deployment achieves up to $214.5\times$ speedup under batch size 1, with parameter count and model size reduced by about 56%. The ablations identify label-specific sentinel representations as the stable design factor, while Shared MLP supplies a compact uniform readout. Furthermore, DirectParallel controls demonstrate that parallel topology alone is insufficient: Generalist pretraining improves stability on AAPD and Reuters, while the DA/CS chain training provides further gains on EURLEX and AAPD, supporting the value of chain-trained encoder representations for parallel readout. Finally, internal sentinel diagnostics demonstrate that the frozen encoder supplies both unified classification directions across all benchmarks and preserved residual co-occurrence on EURLEX and AAPD. These findings link the strong performance of parallel readout to latent structure within the frozen encoder. This supports the separation of chain-based representation learning from inference-time prediction.

Reproducibility statement

Sections 3 and 4 specify the conversion, evaluation protocol, and deployment benchmark. Appendix A records the data and evaluation procedure, readout training configuration, direct-baseline configurations, and timing protocol used for the reported results.

AI use statement

Generative AI tools were used for language editing, literature discovery and synthesis, feedback on experimental and analysis design, and assistance with figures and formatting. The authors reviewed all AI-assisted content, checked numerical claims against experiment records and citations against primary sources, and take responsibility for the final content of this work.

References

Reem M. Al-Otaibi, Peter A. Flach, and Meelis Kull. Multi-label classification: A comparative study on threshold selection methods. In *First International Workshop on Learning over Multiple Contexts (LMCE) at ECML-PKDD*, Nancy, France, 2014. URL https://research-information.bris.ac.uk/en/publications/multi-label-classification-a-comparative-study-on-threshold-selec/.

Chidanand Apté, Fred Damerau, and Sholom M. Weiss. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12(3):233–251, 1994. doi: 10.1145/183422.183423. URL https://doi.org/10.1145/183422.183423.

Elham J. Barezi, Iacer Calixto, Kyunghyun Cho, and Pascale Fung. A study on the autoregressive and non-autoregressive multi-label learning. *arXiv preprint arXiv:2012.01711*, 2020. doi: 10.48550/arXiv.2012.01711. URL https://arxiv.org/abs/2012.01711.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, and Ion Androutsopoulos. Large-scale multi-label text classification on EU legislation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 6314–6322, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1636. URL https://aclanthology.org/P19-1636/.

Ilias Chalkidis, Manos Fergadiotis, Sotiris Kotitsas, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. An empirical study on large-scale multi-label text classification including few and zero-shot labels. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7503–7515, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.607. URL https://aclanthology.org/2020.emnlp-main.607/.

Xinkai Du, Quanjie Han, Yalin Sun, Chao Lv, and Maosong Sun. Label dependencies-aware set prediction networks for multi-label text classification. In *2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 11206–11210. IEEE, 2024. doi: 10.1109/ICASSP48485.2024.10447036. URL https://doi.org/10.1109/ICASSP48485.2024.10447036.

Rong-En Fan and Chih-Jen Lin. A study on threshold selection for multi-label classification. Technical report, Department of Computer Science, National Taiwan University, 2007. URL https://www.csie.ntu.edu.tw/~cjlin/papers/threshold.pdf.

Yova Kementchedjhieva and Ilias Chalkidis. An exploration of encoder-decoder approaches to multi-label classification for legal and biomedical text. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 5828–5843, Toronto, Canada, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.360. URL https://aclanthology.org/2023.findings-acl.360/.

Jack Lanchantin, Arshdeep Sekhon, and Yanjun Qi. Neural message passing for multi-label classification. In *Machine Learning and Knowledge Discovery in Databases*, volume 11907 of *Lecture Notes in Computer Science*, pp. 138–163, Cham, 2020. Springer International Publishing. doi:

10.1007/978-3-030-46147-8_9. URL https://link.springer.com/chapter/10.1007/978-3-030-46147-8_9.

Frederick Liu, Terry Huang, Shihang Lyu, Siamak Shakeri, Hongkun Yu, and Jing Li. EncT5: A framework for fine-tuning T5 as non-autoregressive models. *arXiv preprint arXiv:2110.08426*, 2021. URL https://arxiv.org/abs/2110.08426.

Han Liu, Caixia Yuan, and Xiaojie Wang. Label-wise document pre-training for multi-label text classification. In *Proc. NLPCC*, pp. 641–653, 2020.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8086–8098, Dublin, Ireland, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.556. URL https://aclanthology.org/2022.acl-long.556/.

Jose M. Moyano, Eva L. Gibaja, Sebastian Ventura, and Alberto Cano. Speeding up classifier chains in multi-label classification. In *Proceedings of the 4th International Conference on Internet of Things, Big Data and Security*, pp. 29–37. SciTePress, 2019. doi: 10.5220/0007614200290037. URL https://www.scitepress.org/PublishedPapers/2019/76142/.

Jinseok Nam, Young-Bum Kim, Eneldo Loza Mencía, Sunghyun Park, Ruhi Sarikaya, and Johannes Fürnkranz. Learning context-dependent label permutations for multi-label classification. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 4733–4742. PMLR, 2019. URL https://proceedings.mlr.press/v97/nam19a.html.

Quang Anh Nguyen, Nadi Tomeh, Mustapha Lebbah, Thierry Charnois, and Hanane Azzag. Leveraging text-to-text transformers as classifier chain for few-shot multi-label classification. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 26941–26950, Suzhou, China, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.1368. URL https://aclanthology.org/2025.emnlp-main.1368/.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL https://jmlr.org/papers/v21/20-074.html.

Jesse Read, Bernhard Pfahringer, Geoffrey Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Machine Learning*, 85(3):333–359, 2011. doi: 10.1007/s10994-011-5256-5. URL https://doi.org/10.1007/s10994-011-5256-5.

Jesse Read, Bernhard Pfahringer, Geoffrey Holmes, and Eibe Frank. Classifier chains: A review and perspectives. *Journal of Artificial Intelligence Research*, 70:683–718, 2021. doi: 10.1613/JAIR.1.12376. URL https://doi.org/10.1613/JAIR.1.12376.

Timo Schick and Hinrich Schütze. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 255–269, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.20. URL https://aclanthology.org/2021.eacl-main.20/.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune BERT for text classification? In *Chinese Computational Linguistics*, pp. 194–206. Springer International Publishing, 2019. doi: 10.1007/978-3-030-32381-3_16. URL https://doi.org/10.1007/978-3-030-32381-3_16.

Thanakorn Thaminkaew, Piyawat Lertvittayakumjorn, and Peerapon Vateekul. Prompt-based label-aware framework for few-shot multi-label text classification. *IEEE Access*, 12:28310–28322, 2024. doi: 10.1109/ACCESS.2024.3367994. URL https://doi.org/10.1109/ACCESS.2024.3367994.

Lewis Tunstall, Nils Reimers, Unso Eun Seo Jo, Luke Bates, Daniel Korat, Moshe Wasserblat, and Oren Pereg. Efficient few-shot learning without prompts. *arXiv preprint arXiv:2209.11055*, 2022. doi: 10.48550/arXiv.2209.11055. URL https://arxiv.org/abs/2209.11055.

Lin Xiao, Xin Huang, Boli Chen, and Liping Jing. Label-specific document representation for multi-label text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 466–475, Hong Kong, China, 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1044. URL https://aclanthology.org/D19-1044/.

Pengyu Xu, JingRen Hou, Liping Jing, and Jian Yu. Structured confidence–guided online adaptation for LLM-based multi-label classification. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 16671–16686, San Diego, California, United States, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.822. URL https://aclanthology.org/2026.findings-acl.822/.

Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu, and Houfeng Wang. SGM: Sequence generation model for multi-label classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 3915–3926, Santa Fe, New Mexico, USA, 2018. Association for Computational Linguistics. URL https://aclanthology.org/C18-1330/.

Min-Ling Zhang and Zhi-Hua Zhou. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819–1837, 2014. doi: 10.1109/TKDE.2013.39. URL https://doi.org/10.1109/TKDE.2013.39.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12697–12706. PMLR, 2021. URL https://proceedings.mlr.press/v139/zhao21c.html.

Yaxin Zhu and Hamed Zamani. ICXML: An in-context learning framework for zero-shot extreme multi-label classification. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 2086–2098, Mexico City, Mexico, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.134. URL https://aclanthology.org/2024.findings-naacl.134/.

A Reproducibility Details

Data and evaluation. EURLEX contains European Union legislation annotated with top-level EUROVOC concepts (Chalkidis et al., 2019); AAPD contains scientific abstracts labeled with arXiv subject categories (Yang et al., 2018); and Reuters-21578 is a financial-newswire collection evaluated with its ModApte partition (Apté et al., 1994). To avoid the confounding effects of threshold selection and focus on architectural comparisons, we use a fixed classification threshold of 0.5 (Fan & Lin, 2007; Al-Otaibi et al., 2014). Label is assigned whenever its predicted probability exceeds 0.5. Table 4 summarizes the dataset statistics and available supervision per split.

Table 4: Dataset statistics and per-split few-shot sample sizes. Training and validation each contain $8|L|$ samples. Test gives the size of the unchanged official test set. L/D and W/D denote average labels and words per document.

		Few-shot			Average	
Dataset	$\|L\|$	Train	Valid	Test	L/D	W/D
EURLEX	21	168	168	6,000	3.24	583
AAPD	53	424	424	1,000	2.39	163
Reuters	90	720	720	3,019	1.23	130

Local reproduction of T5 Chain. We do not repeat generalist pre-training (GP). Instead, domain adaptation (DA) is initialized from the GP model already trained and released by the authors of T5 Chain. For every access to a DA training example, the implementation draws a fresh permutation of the canonical labels with NumPy's global random generator and constructs the input–target pair in that order; DA validation instead uses the fixed canonical order. Both DA and chain specialization (CS) update the full T5 encoder–decoder with a restricted two-class cross-entropy computed only from the NO/YES logits at answer positions under gold decoder prefixes; the losses are summed across labels and averaged across examples. After DA, the model with the lowest validation loss is restored and saved as the DA-stage output. Chain selection loads this saved DA model and performs no parameter updates: it evaluates 50 randomly generated candidate permutations on the validation split, ranks them by mean per-example NO/YES loss, and breaks exact ties by the smaller chain seed.

CS then reloads the same DA-stage output and continues full encoder–decoder training with the selected permutation fixed for both training and validation; the selected CS model is therefore not obtained by continuing from the chain-scoring pass, which is evaluation only. DA and CS use learning rate 10^{-4} with linear decay and no warm-up, micro-batch size 2, eight gradient-accumulation steps (effective batch size 16 on one GPU), validation batch size 2, optimization seed 42, and maximum input length 2,000. Their maximum budgets are 5,000 and 1,000 optimizer updates, respectively. Validation and checkpoint saving occur every 50 updates; the validation-loss-best model is restored at the end of each stage, and training stops after 15 DA or 10 CS consecutive validation evaluations without improvement. Our locally trained source chain underperforms the published Reuters result, although the two evaluations use different sampled few-shot splits and are not paired. We explored alternative learning rates and random seeds but did not recover the published Reuters performance. These attempts did not establish a definitive explanation for the discrepancy.

Parallel readout training. We export the specialized T5-v1.1-base encoder and cache its frozen sentinel states in FP32. For the Shared MLP in Equation 5, we set $d_{\mathrm{MLP}} = 128$ and the dropout probability $\rho = 0.1$. The head is trained with AdamW, a learning rate of 10^{-3}, weight decay of 10^{-4}, and batch size 16 for at most 200 epochs; training stops after 15 epochs without validation-$\mathrm{AvgF1}$ improvement. The implementation uses PyTorch 2.3.0, Transformers 4.41.2, and Datasets 2.19.2.

Direct baselines. The DIRECTPARALLEL-BASE and DIRECTPARALLEL-GP use the same selected-chain label prompt and non-autoregressive encoder-plus-Shared-MLP topology as CHAIN2PARALLEL. They differ only in encoder initialization. DIRECTPARALLEL-BASE starts from T5-v1.1-base, whereas DIRECTPARALLEL-GP starts from the T5 Generalist checkpoint after GP. Neither uses DA, CS, completed T5 CHAIN weights, or a decoder. In Step 1 of Figure 4, the encoder and a temporary Shared MLP are jointly trained with binary cross entropy (BCE); the checkpoint with the lowest validation BCE supplies the encoder for Step 2. In Step 2, we discard the temporary head, freeze the selected encoder, and cache its training and validation sentinel states. We then train only a fresh Shared MLP on these features using BCE, and select its checkpoint by the highest validation $\mathrm{AvgF1}$, breaking ties by lower validation BCE.

Direct-baseline configuration. During encoder adaptation in both baselines, AdamW uses a learning rate of 10^{-4} and zero weight decay for the encoder, and a learning rate of 10^{-3} and weight decay of 10^{-4} for the temporary head. Training uses micro-batch size 2, eight gradient-accumulation steps, and validation batch size 2. The encoder learning rate decays linearly over a 6,000-update horizon with no warm-up, while the temporary head learning rate remains constant. DIRECTPARALLEL-BASE and DIRECTPARALLEL-GP both have a 6,000-update cap and patience of 15. Validation occurs every 50 updates, and early stopping has no minimum-step protection. Encoder adaptation uses BF16 with TF32 enabled, gradient checkpointing, maximum input length 2,000, and gradient-norm clipping at 1.0; validation and feature caching use FP32 with TF32 disabled. The fresh head uses width 128, dropout 0.1, learning rate 10^{-3}, weight decay 10^{-4}, training/evaluation batch sizes 16/256, and at most 200 epochs with patience 15, restoring the best weights before evaluation.

Statistical Reporting. Each method is evaluated on matched test splits within a dataset. Tables report the mean with the sample standard deviation as a subscript. For targeted comparisons, we additionally report the mean paired $\mathrm{AvgF1}$ difference across splits and a descriptive two-sided 90% confidence interval, computed from the split-level paired differences using Student-t interval. Published $\mathrm{AvgF1}$ values are derived from the three reported F1 means and have no split-level SD.

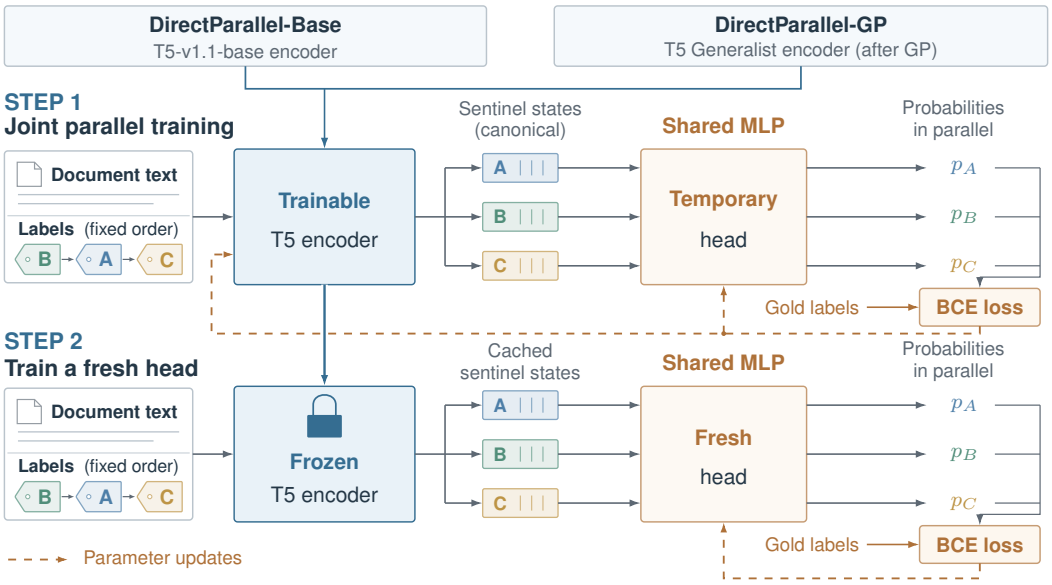


Figure 4: Direct parallel training.

Efficiency measurement. We benchmark the final T5 CHAIN model and its converted CHAIN2PARALLEL counterpart on validation data using an NVIDIA GeForce RTX 4090D in FP32 with TF32 disabled. We report batch-one end-to-end latency, full-validation throughput, peak allocated GPU memory, parameter count, and pure-inference bundle size. End-to-end measurements include preprocessing, host–device transfers, model computation, canonical probability output, and result assembly, but exclude model loading and data-file reading. Batch-one latency is the median of 100 timed passes after 20 warm-up passes, with CUDA synchronization around each measurement. Full-validation throughput uses batch size 2 after one warm-up batch.

B ABLATION DESIGN DETAILS

Recall Section 4.3, we have studied the effectiveness of readout representation, parameter sharing, and readout architecture based on Table 2. To specify how these factors are separated in the ablations, we provide the representation definitions, head parameterizations, and fixed comparison settings below.

Within each data split, all six configurations use the same frozen encoder, gold-label binary cross-entropy objective, and fixed decision threshold $p_k > 0.5$. The configurations comprise four Sentinel readouts (Shared or Label-Specific, each with Linear or MLP architecture) and two Label-Specific Conditioned Global readouts. A Shared head applied to the same global vector for every label would produce identical label scores at inference, so we exclude Shared Conditioned Global configurations and compare six readouts in total.

Readout Representation. Both representations use the identical document-plus-label-prompt encoder input. Sentinel extracts the hidden state at each sentinel token $[\mathrm{E}_j]$ and restores the canonical label order, assigning state $\widetilde{h}_k$ to label l_k as defined in Equations 3 and 4. For attention mask $a_s \in \{0, 1\}$, the global state averages all non-padding encoder states:

$$g = \frac{\sum_{s=1}^{S} a_s H_s}{\sum_{s=1}^{S} a_s}, \qquad g_k = g \;\; (k = 1, \ldots, m). \tag{7}$$

Here $g \in \mathbb{R}^d$ includes document, label-prompt, and tokenizer-added end-of-sequence token states. The same vector $g_k = g$ is supplied to every label l_k, without a label-specific readout position. Representation comparisons use Label-Specific heads and keep the Linear or MLP architecture fixed.

Parameter Sharing. Both configurations take the aligned Sentinel states $\widetilde{h}_k$ as input. The Shared head applies one scoring function f_ϕ to all labels, whereas the Label-Specific head assigns an independent function f_{ϕ_k} to each label l_k. For each comparison, the representation and Linear or MLP architecture are fixed; only parameter sharing changes.

Readout Architecture. Linear heads use a single affine projection; MLP heads use the two-layer form in Equation 5, with shared or independent weights and biases according to the parameter-sharing setting. Architecture comparisons keep the Sentinel states and sharing setting fixed, replacing only the Linear or MLP readout.

C Sentinel Representation Diagnostics

Recall Section 4.4, we have examined classification-direction alignment and label co-occurrence geometry in frozen sentinel states. Here we cover the details for the state selection, diagnostic definitions, representation controls, and randomization procedures used in that analysis.

States and label eligibility. All diagnostics use the same cached sentinel states and eligible labels within each split. The states come from the frozen source-chain encoders underlying the five-split main evaluation, after canonical reordering and before the head's layer normalization. Comparing class means requires both positive and negative examples in each of training and validation; we denote the labels satisfying this condition by $\mathcal{K} \subseteq \{1, \ldots, m\}$, with $K = |\mathcal{K}|$. This gives 20–21 eligible labels on EURLEX, 52–53 on AAPD, and 52–62 on Reuters. Training and validation contain distinct documents; validation also serves source-chain model selection. We calculate each statistic within a split, then report its mean and sample SD across five splits. (Table 5)

Table 5: Frozen sentinel diagnostics across five data splits, with means and sample SDs as subscripts. Direction metrics use cosine similarity; geometry metrics use Spearman correlation with training label co-occurrence.

Statistic	EURLEX	AAPD	Reuters
Classification directions			
Training–validation direction alignment	$0.968_{0.009}$	$0.931_{0.030}$	$0.962_{0.011}$
Training–validation alignment, double-centered	$0.965_{0.012}$	$0.931_{0.027}$	$0.950_{0.016}$
Between-label direction alignment	$0.848_{0.030}$	$0.743_{0.032}$	$0.766_{0.020}$
Alignment with other labels' training directions	$0.888_{0.015}$	$0.828_{0.026}$	$0.850_{0.018}$
Geometry–co-occurrence alignment			
Static label means	$0.131_{0.053}$	$0.037_{0.057}$	$0.130_{0.087}$
Label-centered states	$0.617_{0.031}$	$0.440_{0.025}$	$0.374_{0.050}$
Double-centered states	$0.586_{0.056}$	$0.349_{0.028}$	$0.001_{0.019}$
Own class-mean component	$0.551_{0.062}$	$0.566_{0.035}$	$0.700_{0.042}$
Within-class residual	$0.524_{0.028}$	$0.366_{0.016}$	$0.240_{0.071}$
Double-centered residual	$0.445_{0.069}$	$0.295_{0.030}$	$-0.033_{0.024}$
Own class-mean energy fraction (%)	$21.74_{1.43}$	$13.09_{0.73}$	$14.92_{0.72}$

Classification directions. To determine whether the direction from irrelevant to relevant document states is stable across partitions and similar across labels, we perform the followings. For partition $b \in \{\mathrm{tr}, \mathrm{va}\}$ (training or validation), let n_b be its number of documents, $\widetilde{h}^b_{ik} \in \mathbb{R}^d$ the canonical state for document i and label $k \in \mathcal{K}$, and $y^b_{ik} \in \{0, 1\}$ its target. The positive–negative mean difference and its unit-normalized direction are

$$\Delta^b_k = \frac{\sum_i y^b_{ik} \widetilde{h}^b_{ik}}{\sum_i y^b_{ik}} - \frac{\sum_i (1 - y^b_{ik}) \widetilde{h}^b_{ik}}{\sum_i (1 - y^b_{ik})}, \qquad u^b_k = \frac{\Delta^b_k}{\|\Delta^b_k\|_2}. \tag{8}$$

Here, the summation over i ranges over partition b. The first and second terms denote the mean representation vectors of documents with and without label k, respectively; consequently, Δ^b_k defines the difference vector pointing from the negative-class centroid to the positive-class centroid. With $\|\cdot\|_2$ denoting the Euclidean norm, dividing Δ^b_k by its magnitude yields the unit vector

u_k^b, enabling orientation comparisons unweighted by the magnitude of class-mean discrepancies. Training–validation alignment is $K^{-1}\sum_{k\in\mathcal{K}}(u_k^{\mathrm{tr}})^\top u_k^{\mathrm{va}}$; between-label alignment averages $(u_k^{\mathrm{tr}})^\top u_\ell^{\mathrm{tr}}$ over unordered eligible label pairs $k<\ell$. The other-label statistic excludes the target label from the training reference. We normalize $\sum_{\ell\in\mathcal{K}\setminus\{k\}} u_\ell^{\mathrm{tr}}$, take its cosine with u_k^{va}, and average over k.

Static and document-dependent geometry. Beyond individual classification directions, the geometric analysis links label-pair similarity to label co-occurrence. We separate fixed label means from document-dependent variation and control for components shared across labels within a document. With $\mu_k^b = n_b^{-1}\sum_i \widetilde{h}_{ik}^b$ denoting label k's mean state, label-centering gives A^b; subtracting its within-document average gives double-centered states B^b:

$$A_{ik}^b = \widetilde{h}_{ik}^b - \mu_k^b, \qquad B_{ik}^b = A_{ik}^b - \frac{1}{K}\sum_{\ell\in\mathcal{K}} A_{i\ell}^b. \tag{9}$$

We measure whether labels vary together across documents by comparing their sequences of states. For a validation-state tensor $V\in\mathbb{R}^{n_{\mathrm{va}}\times K\times d}$, the label-pair geometry is

$$G_{k\ell}(V) = \cos(\mathrm{vec}(V_{\cdot k}), \mathrm{vec}(V_{\cdot \ell}))\,, \qquad k,\ell\in\mathcal{K}, \tag{10}$$

where the dot selects all documents, vec concatenates one label's states in document order, and cos denotes cosine similarity. We apply G to A^{va} and B^{va}; static geometry instead uses $\cos(\mu_k^{\mathrm{va}}, \mu_\ell^{\mathrm{va}})$.

Training-label co-occurrence provides a common reference for these geometries. We define $C_{k\ell} = \mathrm{Corr}(y_{\cdot k}^{\mathrm{tr}}, y_{\cdot \ell}^{\mathrm{tr}})$, where Corr denotes Pearson correlation of binary label indicators. For each geometry, we report the Spearman correlation of its upper-triangular entries with those of C, assigning average ranks to ties. For the double-centered direction diagnostic, we apply Equation 9 separately to training and validation, then compute classification directions and their alignment from the resulting states.

Removing each label's own class-mean component. A further control checks whether co-occurrence alignment persists beyond each label's own positive–negative class means. Co-occurring labels can align simply because both move between their positive and negative class means. Let $\bar{y}_k^{\mathrm{va}} = n_{\mathrm{va}}^{-1}\sum_i y_{ik}^{\mathrm{va}}$ be label k's positive frequency in validation. We separate the class-mean component Q from residual variation R and its double-centered counterpart $\overline{R}$:

$$Q_{ik} = (y_{ik}^{\mathrm{va}} - \bar{y}_k^{\mathrm{va}})\Delta_k^{\mathrm{va}}, \qquad R_{ik} = A_{ik}^{\mathrm{va}} - Q_{ik}, \qquad \overline{R}_{ik} = R_{ik} - \frac{1}{K}\sum_{\ell\in\mathcal{K}} R_{i\ell}. \tag{11}$$

Here R contains deviations from each label's positive- or negative-class mean, and $\overline{R}$ additionally removes their within-document average. Table 5 reports the co-occurrence alignment of $G(Q)$, $G(R)$, and $G(\overline{R})$ as "Own class-mean component," "Within-class residual," and "Double-centered residual," respectively, distinguishing the contribution of class means from the remaining variation. To quantify the removed component, we report its mean energy fraction across labels, $K^{-1}\sum_{k\in\mathcal{K}}\sum_i \|Q_{ik}\|_2^2 / \sum_i \|A_{ik}^{\mathrm{va}}\|_2^2$. The residual co-occurrence signal is clearest on EURLEX and AAPD.

Randomization references. We perform randomization tests to examine whether comparably strong alignments can emerge if documents are paired with incorrect label vectors, or if representation geometry is matched to incorrect label identities. These comparisons help assess whether the preceding evidence for stable classification directions and co-occurrence structure depends on the actual correspondences in the data. Within each split, we compare training–validation direction alignment with 199 permutations of whole rows of the validation label matrix, preserving label frequencies and co-occurrence while shuffling document assignments. For geometry alignment, we use 1,999 simultaneous row-and-column permutations of each geometry matrix to shuffle label identities relative to the co-occurrence matrix while preserving the geometry's internal structure. Direction alignment exceeded all row-permutation values in all 15 splits. Double-centered residual alignment exceeded all label-permutation values in every EURLEX and AAPD split, but showed no positive excess on Reuters.

D Deployment Results Across Five Data Splits

The deployment benchmark compares the complete inference paths of the source chain and the parallel classifier. We first specify generation and caching, then explain what is timed and how the results are summarized. Tables 6–8 report means with sample SDs in subscripts across five splits of all three datasets; Chain denotes T5 CHAIN and C2P denotes CHAIN2PARALLEL. Fixed-batch results cover batch sizes 1, 2, and 4; full-validation results use batch size 2.

Table 6: Full-validation results at batch size 2 and deployment footprint across five data splits; means with sample SDs as subscripts.

		(a) Elapsed time and latency			(b) Throughput and peak memory		
Dataset	Method	Elapsed (s)	Batch (ms)	Example (ms)	Rate (ex/s)	Alloc. (MiB)	Reserv. (MiB)
EURLEX	Chain	$96.433_{0.535}$	$1148.01_{6.37}$	$574.00_{3.18}$	$1.742_{0.010}$	$2470.52_{0.88}$	$3587.60_{732.19}$
	C2P	$4.144_{0.446}$	$49.33_{5.30}$	$24.66_{2.65}$	$40.897_{4.100}$	$1656.85_{0.75}$	$2844.00_{827.80}$
AAPD	Chain	$574.718_{3.155}$	$2710.93_{14.88}$	$1355.47_{7.44}$	$0.738_{0.004}$	$1333.43_{25.63}$	$2136.40_{181.69}$
	C2P	$5.138_{0.158}$	$24.24_{0.75}$	$12.12_{0.37}$	$82.587_{2.518}$	$696.05_{24.05}$	$1347.60_{179.45}$
Reuters	Chain	$1623.352_{6.807}$	$4509.31_{18.91}$	$2254.66_{9.45}$	$0.444_{0.002}$	$1996.04_{74.48}$	$3425.60_{752.23}$
	C2P	$9.576_{0.214}$	$26.60_{0.59}$	$13.30_{0.30}$	$75.219_{1.673}$	$1294.21_{99.86}$	$2565.60_{757.45}$

(c) Paired speedup and pure-inference bundle size

Dataset	**Speedup** ($\times$)	Chain (MiB)	C2P (MiB)
EURLEX	$\mathbf{23.47_{2.24}}$	$945.23_{0.00}$	$419.36_{0.00}$
AAPD	$\mathbf{111.94_{3.11}}$	$945.23_{0.00}$	$419.37_{0.00}$
Reuters	$\mathbf{169.59_{3.82}}$	$945.24_{0.00}$	$419.37_{0.00}$

Source-chain generation. The Chain baseline follows constrained autoregressive generation, including the steps needed to emit label markers. It loads the full `T5ForConditionalGeneration` checkpoint and calls Transformers `generate` with `num_beams=1`, `do_sample=False`, and a `prefix_allowed_tokens_fn` constraint. For m labels, the constraint alternates a forced sentinel with a single-token No/Yes answer, followed by a final sentinel and EOS. Thus, `max_new_tokens` is $2m + 2$: m answers, $m + 1$ sentinels, and EOS, excluding the decoder-start token. Each step uses the standard full-vocabulary T5 language-model head followed by token constraints; forced-token forward passes are retained, without a specialized two-token projection. Generation returns per-step scores (`output_scores=True`, `return_dict_in_generate=True`); a two-way softmax over No/Yes scores at each answer position yields Yes probabilities, which are reordered into canonical label order.

Encoder reuse and decoder caching. To distinguish inference-time reuse from precomputed features, both methods run a fresh encoder pass in every timed call. Chain reuses its encoder outputs throughout generation and enables KV caching, with `use_cache=True` in the effective generation configuration.[1] T5 retains decoder self-attention keys and values, processes only the unprocessed decoder token at subsequent steps, and reuses encoder-side cross-attention key/value projections. This avoids repeated prefix projections while retaining sequential decoding and attention over the cached prefix. C2P instead extracts and reorders sentinel states from the encoder pass, then applies the trained head and sigmoid.

Generation checks within timing. The two timing settings measure different scopes of inference. Fixed-batch timing measures probability-output paths; Full-validation timing measures the implemented evaluation paths. The fixed-batch and full-validation paths differ in the checks and output handling included in timing. Fixed-batch Chain timing sets `validate_output=False`, disabling the wrapper's post-generation sequence and score checks. The full-validation path uses

[1] Versioned implementation: `https://github.com/huggingface/transformers/tree/v4.41.2/src/transformers/generation`.

Table 7: Model-only fixed-batch results across five data splits; means with sample SDs as subscripts.

			(a) Batch latency (ms)			(b) Throughput and peak memory		
Dataset	Batch	Method	Median	IQR	p95	Rate (ex/s)	Alloc. (MiB)	Reserv. (MiB)
EURLEX	1	Chain	$1063.97_{5.02}$	$15.59_{2.76}$	$1087.21_{8.78}$	$0.940_{0.004}$	$1063.88_{8.71}$	$1161.60_{24.02}$
		C2P	$16.83_{0.29}$	$0.13_{0.05}$	$17.39_{0.65}$	$59.431_{1.020}$	$536.60_{9.15}$	$583.60_{24.67}$
	2	Chain	$1108.66_{9.44}$	$19.22_{8.17}$	$1130.66_{16.65}$	$1.804_{0.015}$	$1224.06_{36.60}$	$1388.80_{45.27}$
		C2P	$20.00_{2.43}$	$0.11_{0.03}$	$20.49_{2.19}$	$101.186_{12.000}$	$696.61_{36.63}$	$790.80_{36.24}$
	4	Chain	$1154.39_{12.97}$	$19.28_{11.83}$	$1181.44_{17.11}$	$3.465_{0.039}$	$1743.37_{155.27}$	$1970.40_{176.60}$
		C2P	$59.08_{14.60}$	$0.09_{0.05}$	$59.25_{14.64}$	$70.880_{16.243}$	$1215.76_{155.59}$	$1339.20_{176.12}$
AAPD	1	Chain	$2588.80_{27.97}$	$48.10_{26.13}$	$2654.28_{52.13}$	$0.386_{0.004}$	$1054.20_{4.93}$	$1122.00_{1.41}$
		C2P	$16.79_{0.30}$	$0.11_{0.08}$	$16.97_{0.35}$	$59.585_{1.046}$	$490.74_{5.21}$	$513.60_{5.90}$
	2	Chain	$2693.26_{22.71}$	$27.44_{7.84}$	$2744.22_{29.06}$	$0.743_{0.006}$	$1111.07_{6.62}$	$1228.40_{9.74}$
		C2P	$17.06_{0.25}$	$0.18_{0.14}$	$17.44_{0.45}$	$117.226_{1.728}$	$542.49_{12.33}$	$582.00_{16.49}$
	4	Chain	$2741.95_{21.62}$	$36.50_{12.68}$	$2813.10_{33.01}$	$1.459_{0.012}$	$1274.73_{7.98}$	$1372.00_{26.76}$
		C2P	$20.33_{1.24}$	$0.08_{0.07}$	$20.47_{1.27}$	$197.341_{11.881}$	$658.91_{18.64}$	$695.60_{24.63}$
Reuters	1	Chain	$4335.72_{28.09}$	$35.75_{7.01}$	$4399.56_{32.63}$	$0.231_{0.001}$	$1073.68_{4.93}$	$1152.40_{3.29}$
		C2P	$16.61_{0.08}$	$0.06_{0.01}$	$16.98_{0.55}$	$60.194_{0.299}$	$492.87_{7.67}$	$518.80_{9.55}$
	2	Chain	$4494.90_{22.95}$	$45.97_{18.23}$	$4560.70_{32.26}$	$0.445_{0.002}$	$1178.80_{8.84}$	$1287.20_{30.68}$
		C2P	$17.04_{0.12}$	$0.06_{0.02}$	$17.15_{0.14}$	$117.405_{0.803}$	$547.18_{20.38}$	$589.20_{28.52}$
	4	Chain	$4560.98_{4.41}$	$28.77_{4.72}$	$4613.06_{7.65}$	$0.877_{0.001}$	$1347.82_{8.10}$	$1454.00_{8.12}$
		C2P	$22.22_{1.15}$	$0.06_{0.06}$	$22.32_{1.17}$	$180.369_{9.435}$	$690.10_{18.49}$	$729.60_{32.54}$

(c) Paired speedup (×)

Batch	EURLEX	AAPD	Reuters
1	$\mathbf{63.23_{0.84}}$	$\mathbf{154.24_{2.07}}$	$\mathbf{260.98_{1.37}}$
2	$\mathbf{56.11_{6.84}}$	$\mathbf{157.85_{2.13}}$	$\mathbf{263.86_{1.94}}$
4	$\mathbf{20.43_{4.59}}$	$\mathbf{135.28_{8.32}}$	$\mathbf{205.66_{10.68}}$

`teacher_raw_outputs`, retaining those checks, including per-step finite-score checks with GPU-to-host scalar reads. It also extracts Yes-minus-No logits alongside probabilities and transfers both to the CPU. Full-validation speedups therefore characterize the complete implemented evaluation paths, including this additional validation and postprocessing work.

Measurement protocol. We control hardware, precision, process isolation, and input batches to keep the two methods comparable. All runs use an RTX 4090D, FP32, PyTorch 2.3.0/CUDA 12.1, and Transformers 4.41.2, with TF32 and autocast disabled. Each method–batch configuration runs in a separate process, with method order alternating across batch sizes. Length-quantile batches are matched between methods; each batch size has its own selection and padding. Fixed-batch timing uses 20 warm-up and 100 timed passes with CUDA synchronization; full-validation timing covers one complete pass after one warm-up batch.

Timing and statistics. The timing boundaries separate model computation from surrounding input/output work. Model-only timing maps tokenized GPU inputs to canonical GPU probabilities; end-to-end timing adds prompt construction, tokenization, and CPU–GPU transfers, while full-validation timing also includes output assembly. Model loading and data-file reading precede timing. Fixed-batch latency reports the median, interquartile range (IQR), and 95th percentile (p95) within each run; throughput is batch size divided by median latency in seconds. Full-validation throughput divides example count by elapsed time; batch and example latencies divide elapsed time by the respective counts. Speedups preserve the pairing between methods within each split. We average Chain/C2P latency ratios across splits, using medians for fixed batches and elapsed times for full validation. The memory figures capture resource use alongside runtime. Allocated memory covers tensors; reserved memory is the CUDA allocator reservation. Both peaks are measured after warm-up with the model retained; MiB denotes 2^{20} bytes.

Table 8: End-to-end fixed-batch results across five data splits; means with sample SDs as subscripts.

			(a) Batch latency (ms)			(b) Throughput and peak memory		
Dataset	Batch	Method	Median	IQR	p95	Rate (ex/s)	Alloc. (MiB)	Reserv. (MiB)
EURLEX	1	Chain	$1065.53_{5.32}$	$12.39_{5.35}$	$1086.46_{6.81}$	$0.939_{0.005}$	$1063.89_{8.71}$	$1161.60_{24.02}$
		C2P	$20.95_{0.32}$	$0.61_{0.28}$	$22.02_{0.47}$	$47.733_{0.717}$	$536.61_{9.15}$	$583.60_{24.67}$
	2	Chain	$1115.13_{6.09}$	$18.55_{6.91}$	$1144.96_{16.11}$	$1.794_{0.010}$	$1224.08_{36.60}$	$1388.80_{45.27}$
		C2P	$27.90_{2.78}$	$0.31_{0.22}$	$28.80_{2.41}$	$72.237_{7.068}$	$696.64_{36.64}$	$790.80_{36.24}$
	4	Chain	$1164.48_{17.30}$	$17.46_{6.71}$	$1189.73_{18.46}$	$3.436_{0.051}$	$1743.44_{155.28}$	$1970.40_{176.60}$
		C2P	$75.47_{15.56}$	$0.27_{0.10}$	$75.97_{15.68}$	$54.751_{10.663}$	$1215.83_{155.60}$	$1339.20_{176.12}$
AAPD	1	Chain	$2598.69_{36.39}$	$51.00_{35.87}$	$2661.02_{67.53}$	$0.385_{0.005}$	$1054.21_{4.93}$	$1122.00_{1.41}$
		C2P	$20.35_{0.49}$	$0.17_{0.12}$	$20.70_{0.61}$	$49.154_{1.150}$	$490.75_{5.21}$	$513.60_{5.90}$
	2	Chain	$2700.22_{30.20}$	$31.82_{12.25}$	$2740.30_{34.64}$	$0.741_{0.008}$	$1111.09_{6.62}$	$1228.40_{9.74}$
		C2P	$23.81_{0.45}$	$0.23_{0.17}$	$24.37_{0.83}$	$84.018_{1.586}$	$542.51_{12.33}$	$582.00_{16.49}$
	4	Chain	$2756.03_{21.64}$	$34.66_{13.24}$	$2803.47_{24.53}$	$1.451_{0.011}$	$1274.76_{7.98}$	$1372.00_{26.76}$
		C2P	$33.21_{1.58}$	$0.34_{0.23}$	$33.66_{1.45}$	$120.675_{5.707}$	$658.95_{18.65}$	$695.60_{24.63}$
Reuters	1	Chain	$4337.83_{25.25}$	$35.97_{6.45}$	$4401.54_{22.39}$	$0.231_{0.001}$	$1073.68_{4.93}$	$1152.40_{3.29}$
		C2P	$20.23_{0.14}$	$0.10_{0.06}$	$21.12_{1.64}$	$49.437_{0.350}$	$492.88_{7.68}$	$518.80_{9.55}$
	2	Chain	$4488.06_{10.60}$	$39.97_{8.05}$	$4567.81_{21.30}$	$0.446_{0.001}$	$1178.81_{8.84}$	$1287.20_{30.68}$
		C2P	$23.80_{0.24}$	$0.07_{0.01}$	$23.93_{0.25}$	$84.028_{0.829}$	$547.19_{20.38}$	$589.20_{28.52}$
	4	Chain	$4581.62_{12.47}$	$40.46_{7.12}$	$4643.94_{14.01}$	$0.873_{0.002}$	$1347.85_{8.10}$	$1454.00_{8.12}$
		C2P	$35.60_{1.33}$	$0.14_{0.10}$	$36.20_{1.90}$	$112.478_{4.175}$	$690.14_{18.49}$	$729.60_{32.54}$

(c) Paired speedup ($\times$)

Batch	EURLEX	AAPD	Reuters
1	$\mathbf{50.86_{0.63}}$	$\mathbf{127.72_{2.52}}$	$\mathbf{214.45_{1.72}}$
2	$\mathbf{40.28_{4.00}}$	$\mathbf{113.43_{2.10}}$	$\mathbf{188.56_{1.65}}$
4	$\mathbf{15.90_{2.89}}$	$\mathbf{83.16_{4.28}}$	$\mathbf{128.83_{4.72}}$

Deployment footprint. Beyond runtime and peak memory, the deployment footprint captures the parameters and files needed for inference. Chain and C2P contain 247,577,856 and 109,728,641 deployed parameters, respectively. Bundles include inference weights, tokenizer, label metadata, and configuration, plus the C2P head.

E Large Language Model Baseline

The LLM baseline tests an alternative use of the few-shot training data: constructing demonstrations for inference. We use the qwen3.8-flash model identifier on EURLEX. For each of the five data splits, the complete $8|L| = 168$-document training set is used to construct label-wise demonstrations. For every label, all positive training documents in that split are supplied to the model, which produces a compact decision rule and a compressed representative document while retaining its complete gold label set. At classification time, the prompt contains the task definition, all 21 legal label identifiers and natural-language names, the 21 refined demonstrations, and the target document. As shown in Table 9, Chain2Parallel exceeds the ICL baseline by 8.1 mF1, 6.3 MF1, 8.0 IF1, and 7.5 AvgF1 points.

Table 9: EURLEX comparison with a LLM in-context-learning baseline. Values are means with sample SDs.

Method	mF1	MF1	IF1	AvgF1
Chain2Parallel	$\mathbf{75.0_{0.2}}$	$\mathbf{56.7_{1.2}}$	$\mathbf{75.2_{0.3}}$	$\mathbf{69.0_{0.6}}$
Qwen3.8-Flash ICL	$66.9_{2.3}$	$50.4_{2.1}$	$67.2_{2.4}$	$61.5_{2.2}$

Table 10: Time and token usage for Qwen3.8-Flash ICL. Refine is the one-time demonstration-construction time, and classify is client-observed wall-clock time. Input and output tokens are classification-only; amortized tokens also include refinement usage.

Refine (s)	Classify (s)	Docs/s	Input tok/doc	Output tok/doc	Amortized tok/doc
113.8 ± 10.4	266.3 ± 64.3	5.89 ± 1.41	$5{,}262 \pm 63$	26.5 ± 1.6	$5{,}648 \pm 49$

The cost analysis separates one-time demonstration construction from repeated classification. Table 10 summarizes time and token usage across the five splits. Refinement took 113.8 seconds per split on average, while classifying 1,500 documents took 266.3 seconds with eight concurrent API workers, corresponding to 5.89 documents per second. Each classification request used an average of 5,262.3 input tokens and 26.5 output tokens. After amortizing refinement over the 1,500 evaluated documents, total usage was 5,648.1 tokens per document.

The results illustrate the benefit of a task-specialized parallel classifier for this fixed-label task. By reusing its trained encoder, CHAIN2PARALLEL predicts labels without demonstration prompts at inference. In the local deployment benchmark, CHAIN2PARALLEL reaches 40.9 examples per second on an RTX 4090D (Table 6); this GPU measurement and the API timings above describe different execution environments.

F DISCUSSION AND FUTURE WORK

The current evaluation of CHAIN2PARALLEL focuses on T5 CHAIN, where the conversion uses label-aligned sentinel states and a chain-to-label mapping. Future work could explore alternative label anchors and alignment schemes for other chain architectures.

The sentinel diagnostics characterize the geometry of encoder representations used for parallel readout. Targeted interventions on the sentinel states could help distinguish causal effects from the observed correlations. Because the parallel readout is not explicitly conditioned on preceding decoded decisions, it may not retain dependency information represented only in the decoder trajectory. Lightweight interactions among label states in the readout could be explored to capture additional dependency information while retaining parallel prediction.

Including all label descriptions in one encoder input makes sequence length and encoding cost grow with the label set. Exploring label selection or grouped encoding could help extend the input design to larger label sets, while changing label sets would also require updating the label alignment. Finally, the conversion inherits the source chain's representation quality and potential biases, while absolute efficiency remains dependent on the hardware and serving implementation. Broader evaluations could assess inherited biases and test the portability of efficiency gains across hardware and serving configurations.

G NOTATION SUMMARY

Table 11 collects the notation used in the method and ablations. Standard arithmetic operators are omitted unless they carry a paper-specific meaning.

Table 11: Notation used in the method and readout ablations.

Symbol	Domain or shape	Meaning
Task and classifier-chain notation		
L	label set	Canonically ordered label set $L = \{l_1, \ldots, l_m\}$.
l_k	label	Canonical label k; it is represented in the encoder prompt by its textual label description.
$m = \lvert L \rvert$	scalar	Number of labels.
$x,\ x_i$	text	An input document and training document i, respectively.
$y,\ y_k,\ y_{ik}$	$\{0,1\}^m$, $\{0,1\}$	Multi-label target vector, its component for label k, and the target for document i and label k.

Continued on the next page

Table 11 continued

Symbol	Domain or shape	Meaning
i, j, k, s	indices	Document, chain-position, canonical-label, and token-position indices, respectively.
$\mathfrak{S}_m$	permutation set	All permutations of the m canonical label indices.
$\pi,\ \pi^\star$	$\mathfrak{S}_m$	A chain permutation and the validation-selected permutation retained for chain specialization.
$\pi(j)$	$\{1, \ldots, m\}$	Canonical label index placed at chain position j.
$(\pi^\star)^{-1}(k)$	$\{1, \ldots, m\}$	Chain position occupied by canonical label k under the selected permutation.
$[\mathrm{E}_j]$	sentinel token	Hard anchor associated with chain position j.
z^π	token sequence	Encoder input containing x, the sentinels, and the permuted textual label descriptions.
v^π	token sequence	Alternating decoder output of sentinel markers and YES/NO decisions.
$v^\pi_{2j-1},\ v^\pi_{2j}$	tokens	Sentinel at output position $2j-1$ and the corresponding YES/NO decision at position $2j$.
$v^\pi_{<2j}$	token prefix	Decoder output preceding the decision at position $2j$.
q_θ	distribution	Source-chain YES/NO conditional distribution.
θ	parameters	Parameters of the source T5 encoder–decoder.
$\mathbb{E}_{(x,y)}$	operator	Expectation over labeled training examples (x, y).
$\mathcal{L}^{\mathrm{CS}}$	scalar	Fixed-order chain-specialization loss.
Exported encoder and parallel readout		
Enc	function	Frozen encoder exported from the completed source chain.
S	scalar	Tokenized encoder-input length, including padding.
d	scalar	Encoder hidden dimension.
H	$\mathbb{R}^{S\times d}$	Sequence of encoder hidden states for $z^{\pi^\star}$.
H_s	$\mathbb{R}^d$	Encoder state at token position s.
s_j	$\{1, \ldots, S\}$	Input-token position at which sentinel $[\mathrm{E}_j]$ occurs; it is a position index, not a count.
h_j	$\mathbb{R}^d$	Sentinel state H_{s_j} associated with chain position j.
$\widetilde{h}_k$	$\mathbb{R}^d$	Sentinel state reordered to canonical label k.
$\widetilde{H}$	$\mathbb{R}^{m\times d}$	Matrix obtained by stacking all canonical label-aligned sentinel states.
r_k	$\mathbb{R}^d$	Layer-normalized state $\mathrm{LN}(\widetilde{h}_k)$.
d_{MLP}	scalar	Shared-MLP hidden width, fixed to 128.
$D_{0.1}$	function	Dropout with probability 0.1 during training and the identity at inference.
$W_1,\ b_1$	$\mathbb{R}^{d_{\mathrm{MLP}}\times d}$, $\mathbb{R}^{d_{\mathrm{MLP}}}$	First-layer weight matrix and bias of the shared MLP.
$w_2,\ b_2$	$\mathbb{R}^{d_{\mathrm{MLP}}}$, $\mathbb{R}$	Output-layer weight vector and scalar bias of the shared MLP.
$p_k,\ p_{ik}$	$(0, 1)$	Predicted relevance probability for label k, for one document and for training document i, respectively.
N	scalar	Number of documents used to train the parallel head.
ϕ	parameters	Trainable readout parameters: layer-normalization parameters and shared-MLP weights and biases.
$\mathcal{L}_{\mathrm{head}}$	scalar	Mean binary cross-entropy objective over N documents and m labels.
$\widehat{y}_k$	$\{0, 1\}$	Predicted label indicator $\mathbb{1}[p_k > 0.5]$.
a_s	$\{0, 1\}$	Encoder attention mask at token position s; 1 denotes a non-padding token.
$g,\ g_k$	$\mathbb{R}^d$	Mean-pooled non-padding encoder state and its identical copy assigned to label k in the Conditioned Global ablation.
Operators		
$\mathrm{LN}, \mathrm{GELU}, \sigma$	functions	Layer normalization, Gaussian error linear unit, and logistic sigmoid.
$\mathrm{BCE}(t, p)$	scalar function	Binary cross entropy between binary target t and probability p.

Continued on the next page

Table 11 continued

Symbol	Domain or shape	Meaning
$(\cdot)^\top$	operator	Transpose.
$O(\cdot)$	asymptotic notation	Growth rate used for parameter-count or computational scaling.