Systematic Design Choices for Fine-Tuning Text-Classification Models: Projection Space, Task Instructions, and Label Encoding

Jian Lyu[a,*], Jingfeng Xue[a], Weizhi Meng[b], Weijie Han[a], Junbao Chen[a], Zeyang Liu[a]

[a]*School of Computer Science and Technology, Beijing Institute of Technology, 100081, Beijing, China*
[b]*School of Computing and Communications, Lancaster University, LA1 4YW, Lancaster, United Kingdom*

Abstract

Text classification (TC) is a foundational task in natural language processing (NLP), where supervised fine-tuning (SFT) of pre-trained language models (PLMs) has become the dominant paradigm. However, systematic investigations of three key design choices within fine-tuning frameworks for TC are still lacking: (1) using a classification head projecting to the label space versus the vocabulary space, (2) augmenting input text with task instructions, and (3) integrating label text directly into training sequences. To address this gap, we introduce a principled $2 \times 2 \times 2$ design matrix and conduct an empirical study grounded in this unified methodological framework across four core benchmark datasets (two Chinese and two English) using both encoder-only and decoder-only PLMs, and further validate our findings on multi-label and long-document benchmarks. Results indicate that classification heads projecting to the label space or the vocabulary space achieve comparable performance. Explicit task instructions, while effective in few-shot in-context learning (ICL), do not consistently improve performance in supervised fine-tuning. Notably, although decoder-only models exhibit the capability to learn from label-appended sequences, this behavior superficially resembles ICL and fundamentally arises from architectural alignment between label placement and causal next-token supervision rather than genuine reasoning over label semantics. These results provide both analytical insight into PLM supervision dynamics and actionable design guidelines for efficient TC workflows.[1]

Keywords: Text classification, Fine-tuning, Projection space, Task instructions, Label encoding

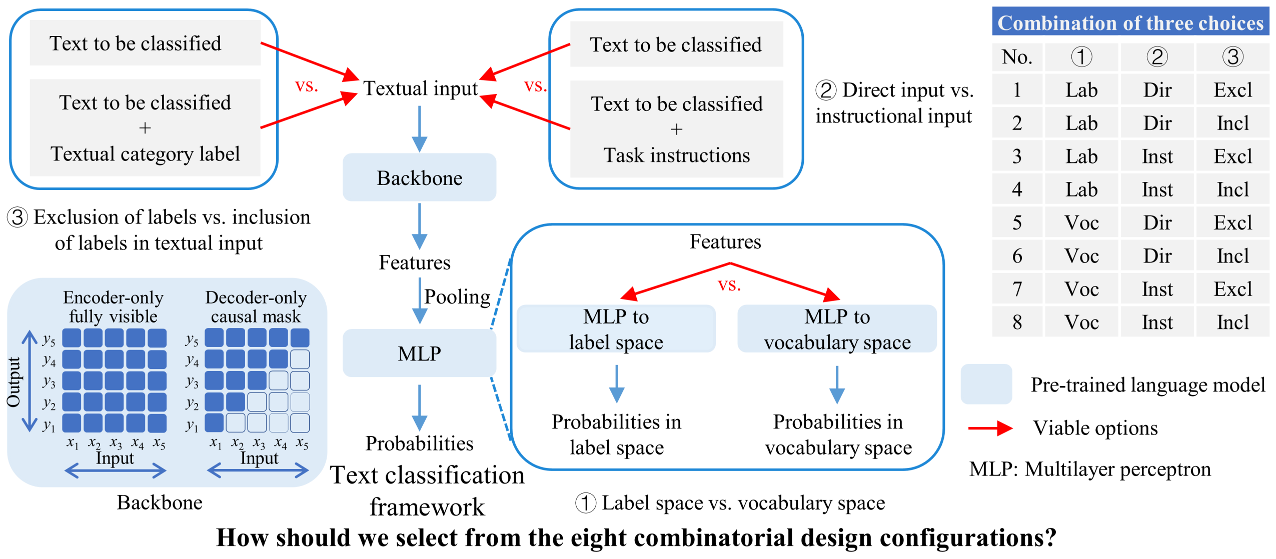
1. Introduction

Text classification (TC), alternatively referred to as text categorization, is a classical problem in natural language processing (NLP). The fundamental objective of TC lies in the attribution of distinctive tags or labels to textual entities encompassing sentences, queries, paragraphs, and entire documents [1]. It has found widespread applications in sentiment analysis, news categorization, spam detection, intent recognition, content moderation, information retrieval, and many other NLP domains [2]. The rise of Transformer-based [3]

[*]Corresponding author
[1]All source code and processed datasets are publicly available at `https://github.com/JianLyu07/design_choices`.

pre-trained language models (PLMs) like BERT [4] and GPT [5] has established supervised fine-tuning (SFT) as a prevalent method for TC [6].

Figure 1 illustrates a commonly used SFT framework for TC based on PLMs, where contextualized representations from the PLM backbone are passed to a task-specific multilayer perceptron (MLP) for final classification. Despite its simplicity, the framework incorporates three fundamental design choices that warrant attention. First, the decision between label-space projection [4, 7] and vocabulary-space projection [8, 9] constitutes a key architectural trade-off in classification head design. Second, another important choice lies in the use of direct input [10] versus instructional input [11]. Third, the explicit inclusion [12] or exclusion [13] of labels within the input context during training introduces a third critical design dimension.



Text to be classified
Text to be classified + Textual category label
vs.
Textual input
vs.
Text to be classified
Text to be classified + Task instructions
② Direct input vs. instructional input
③ Exclusion of labels vs. inclusion of labels in textual input
Backbone
Features
Pooling
MLP
Probabilities
Text classification framework
Encoder-only fully visible
Decoder-only causal mask
Output
Input
y_5 y_4 y_3 y_2 y_1
x_1 x_2 x_3 x_4 x_5
Backbone
Features
vs.
MLP to label space
MLP to vocabulary space
Probabilities in label space
Probabilities in vocabulary space
① Label space vs. vocabulary space
Pre-trained language model
Viable options
MLP: Multilayer perceptron
How should we select from the eight combinatorial design configurations?

Combination of three choices

No.	①	②	③
1	Lab	Dir	Excl
2	Lab	Dir	Incl
3	Lab	Inst	Excl
4	Lab	Inst	Incl
5	Voc	Dir	Excl
6	Voc	Dir	Incl
7	Voc	Inst	Excl
8	Voc	Inst	Incl

Figure 1: Text classification framework using pre-trained language models (PLMs) and three critical design choices. The term ”backbone” refers exclusively to the transformer layers of PLMs (e.g., BERT encoder blocks), excluding any pretraining or downstream task heads.

The standard fine-tuning configuration typically adopts label-space projection, direct input, and label exclusion. In contrast, the alternative choices—vocabulary-space projection, instructional input, and explicit label inclusion—are commonly employed in unified NLP frameworks (e.g., T5) [8], instruction tuning paradigms (e.g., FLAN) [11], and few-shot TC or in-context learning (ICL) settings [12, 14, 15]. These three design dimensions can significantly impact performance and computational efficiency, making them critical targets for systematic investigation. However, prior research has predominantly addressed few-shot, zero-shot, or ICL settings. A systematic investigation into how these three factors affect performance in data-rich supervised fine-tuning scenarios (with hundreds of examples or more) remains largely absent in the current literature.

To address this gap, we conducted a systematic comparative study based on the framework in Figure 1. The experimental design employed six benchmark datasets (four core single-label tasks plus two additional challenging benchmarks—one multi-label and one long-document—across English and Chinese) and a set of encoder-only and decoder-only PLMs spanning from base-scale models to a 14B-parameter LLM. We evaluated eight combinatorial strategies, each representing a unique configuration across the three design dimensions (two options per dimension). These strategies were assessed as reciprocal baselines over a comprehensive set of dataset–model–strategy combinations. Our study establishes, through controlled ablation, that: (1) Label-space and vocabulary-space projections perform comparably in supervised TC across datasets, architectures, and model scales; (2) Explicit task instructions offer no consistent performance benefit in SFT, in contrast to their importance in few-shot ICL; (3) Exposing label tokens in the input creates architecture-dependent positional shortcuts. Encoder-only models degrade sharply under label inclusion, while decoder-only models' apparent label comprehension arises from architectural alignment between label placement and causal next-token supervision rather than genuine in-context learning. Beyond empirical metrics, our work contributes the following conceptual advances:

- We present the first theoretically-grounded, factorial evaluation of three orthogonal design choices—projection space, instructional context, and label encoding—for supervised TC.
- We uncover the mechanistic basis underlying decoder-only models' robustness to label-appended training, linking it to offset alignment rather than emergent reasoning.
- We distill formally-supported, architecture-aware guidelines: employ label-space heads for standalone classifiers, reuse the model's native vocabulary-space head (causal or masked LM) only when already present, and exclude both instructional templates and label-included inputs during supervised fine-tuning.

The remainder of this paper is organized as follows: Section 2 reviews related work and identifies research gaps; Section 3 presents the combined training strategies; Section 4 details the experimental configurations and reports results and analysis; and Section 5 concludes with key findings and future directions.

2. Related work

2.1. Research gaps

TC fine-tuning predominantly follows the BERT-derived paradigm: label-space projection, direct raw text input, and label exclusion during encoding [4, 7]. More broadly, contemporary PLM adaptation regimes differ along two key dimensions: parameter updating and instructional integration. Specifically, zero-shot prompting (including in-context learning) operates with a frozen PLM guided by natural-language instructions and demonstrations at inference time; supervised fine-tuning (SFT) updates parameters on labeled examples for a single target task; and instruction tuning extends this by performing multi-task fine-tuning on large collections of (instruction, output) pairs to instill generic instruction-following ability prior to task-specific adaptation [16, 17]. Empirically, in text classification, moderately sized models trained via SFT frequently match or outperform few-shot prompting with much larger PLMs once sufficient labeled data are available [18, 19, 20]. Our work focuses specifically on the supervised fine-tuning regime, examining how architectural heads, input formulations, and label encodings within the SFT pipeline influence performance and computational efficiency for fixed downstream tasks, thereby providing a methodological baseline for PLM-based text classification workflows.

Nevertheless, within SFT, the dominant BERT-derived configuration has not been systematically benchmarked against alternative designs employed in unified and instruction-oriented frameworks. Unified frameworks, such as T5-style models, leverage vocabulary-space projection and templated inputs [8, 21]; yet their efficacy in standard supervised TC settings remains underexplored. Similarly, while instruction tuning demonstrates substantial gains in zero-shot and cross-task generalization [11], the utility of explicit task instructions within single-task SFT pipelines—particularly in data-rich settings—remains unclear and under-explored. Complementary application-driven work on neural text processing, for example spelling correction, typically concentrates on a single task and a fixed training setup rather than systematically varying supervision design [2].

A deeper theoretical tension concerns the role of label text. ICL studies suggest that appending labels to the input text (e.g., "Text: ... Label: ...") might induce implicit reasoning behaviors [22, 23]. In contrast, encoder architectures trained with MLM inherently isolate label processing from text encoding [4], treating labels as physically separate prediction targets. This dichotomy remains unresolved, particularly regarding whether models genuinely leverage label semantics or merely exploit positional patterns. Existing studies address these design choices in isolation—across few-shot vs. full-data settings and encoder vs. decoder architectures [14, 24]—yet they lack a unified evaluation.

The three design dimensions we investigate—projection space, instructional context, and label integration—thus constitute an underexplored design space within SFT. Their collective impact across diverse datasets and model architectures has not been systematically characterized, motivating the methodological baseline study presented here.

2.2. Technical background

Projection Space. Label-space projection follows the BERT-style [CLS] + MLP architecture, widely validated in industrial-scale applications [7]. Vocabulary-space projection builds on T5-style generative heads [8], framing label prediction as a token-level language modeling task via likelihood maximization. Theoretically, both approaches are grounded in representation learning principles: label-space projection aims to encourage intra-class compactness in the learned representation space [25], while vocabulary-space projection exploits the semantic structure of the pre-trained token embedding distribution [26].

Instruction. The use of instructional inputs in our study is based on FLAN's structured prompting paradigm [11], but we adapt these inputs for supervised fine-tuning settings, beyond their typical use in few-shot inference. This differs from "instruction tuning" in the LLM literature, where models are fine-tuned on heterogeneous mixtures of instruction–response pairs to acquire generic instruction-following abilities across tasks [16]. In our experiments, FLAN-style instructions serve only as optional natural-language prefixes within a single-task SFT pipeline.

Label encoding. Label serialization in our experiments mirrors the formatting used in ICL [12], but our setting introduces a key point of methodological scrutiny—testing whether models rely on semantic interpretation of labels or simply exploit positional heuristics. We employ a controlled ablation design to isolate the effect of each variable, validating results across standard TC benchmarks in both English and Chinese, using macro-averaged F1 as the primary evaluation metric.

2.3. Differentiation from prior work

Our work diverges from existing studies in three key aspects: (1) Rather than isolating single factors (e.g., instruction tuning for zero-shot transfer), we propose a joint evaluation framework that systematically assesses the potentially interdependent effects of projection space, instructional context, and label integration; (2) In contrast to prevailing ICL interpretations, we demonstrate empirically that decoder-based models' apparent label comprehension during fine-tuning primarily arises from the interaction between causal attention and logit–label alignment mechanisms, rather than from genuine semantic reasoning; (3) We identify that under specific conditions, certain configurations—such as vocabulary-space and label-space projections, as well as direct and instructional inputs—can achieve statistically comparable performance. This finding supports the adoption of simplified implementation strategies for TC, addressing practical efficiency concerns often overlooked in theoretical analyses.

3. Methodology

In this section, we present a unified abstraction of Transformer-based PLM backbones. Based on this abstraction, we analyze three core design dimensions in the TC framework and derive eight training strategies from their combinatorial configurations.

3.1. Unified input-output mapping for Transformer backbones

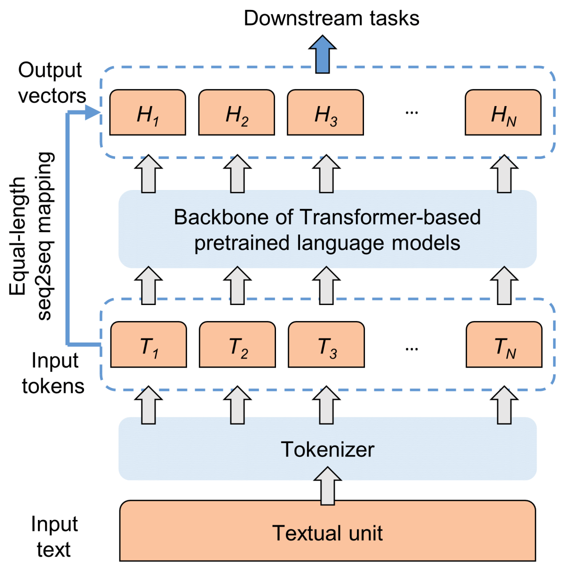


Figure 2: Illustration of equal-length input-output mapping in Transformer backbones. Regardless of whether the model is encoder-only or decoder-only, each forward pass produces a hidden vector for each input token, enabling a consistent representation mechanism across architectures.

Viewed as black boxes, encoder-only and decoder-only Transformer backbones share a fundamental property: each forward pass implements an equal-length sequence-to-sequence (seq2seq) mapping (Figure 2). Concretely, a token sequence (potentially including special markers such as [CLS], [SEP], or [EOS]) is mapped to a vector sequence of identical length in a single forward pass, regardless of internal architectural differences. Although the output sequence expands progressively during autoregressive decoding, each forward pass operates on a fixed-length input to produce hidden states. This unified viewpoint abstracts away architectural differences in attention mechanisms (bidirectional vs. causal) and establishes the foundational consistency necessary for our comparative study.

We formalize this unified mapping as follows. Given an input text sequence x, the Transformer backbone defines a parameterized function f_θ that generates contextualized representations:

$$\mathbf{H} = f_\theta(x) = [H_1, H_2, \ldots, H_N] \tag{1}$$

where $H_i \in \mathbb{R}^h$ denotes the hidden state vector for the i-th token in the tokenized sequence of length N, and h is the hidden dimension. This formulation implicitly handles tokenization and supports both encoder-only models with full self-attention and decoder-only models with causal masking via appropriate constraints on the attention masks in f_θ.

This abstraction creates a clean separation of concerns: the backbone exposes a single interface $\mathbf{H} = f_\theta(x)$, where x is treated as a unified input string independent of tokenization details. The resulting hidden sequence $\mathbf{H}$ then provides a common feature space on which downstream classification heads operate.

3.2. Definitions of the three design dimensions

Text classification (TC) tasks can be formulated in terms of conditional probabilities. Let x denote the textual input and y the target class. The objective of TC is to identify a label y that maximizes the conditional probability $P(y \mid x)$, which can be written as:

$$\hat{y} = \arg\max_y P(y \mid x) \tag{2}$$

where $\hat{y}$ denotes the predicted class label.

Building upon the unified seq2seq mapping established in Section 3.1, we formulate the text classification framework as a composition of transformations. The complete pipeline comprises two sequential components: the Transformer backbone f_θ that maps input text x to contextualized representations $\mathbf{H}$, and the classification head g_ϕ that maps these representations to real-valued scores over classes. To obtain probabilities, we apply a task-dependent normalization function ρ to these scores, yielding

$$P(y \mid x) = \rho\big(g_\phi(f_\theta(x))\big) \tag{3}$$

where $g_\phi(f_\theta(x)) \in \mathbb{R}^n$ denotes a vector of class-wise scores and $\rho(\cdot)$ maps these scores to a probability distribution over labels (e.g., via a softmax mapping; the concrete instantiation of ρ is specified in Section 3.3). The projection space determines the output domain of g_ϕ and consequently the nature of the target variable y, while the input paradigm and label context govern the construction of the input x. We now elaborate on each of these three design dimensions.

First, projection space. The projection space defines the output domain of the classification head g_ϕ in Equation (3), and thus determines whether the backbone representation is mapped to the label set or to the model vocabulary. For a classification problem with n classes and label set $\mathcal{Y} = \{1, 2, \ldots, n\}$, we consider two instantiations of g_ϕ as illustrated in Figure 3.

Label-space projection uses a pooled representation of the backbone outputs $\mathbf{H} = f_\theta(x)$. Let

$$H_{\text{pool}}^{\text{lab}} = \text{Pool}_{\text{lab}}(\mathbf{H}) \in \mathbb{R}^h \tag{4}$$

denote a pooled feature vector, where $\mathrm{Pool}_{\mathrm{lab}}(\cdot)$, for example, may take the hidden state of a special token (such as [CLS]), the last token, or the mean of all token representations. The classification head then applies an affine projection to the label space:

$$g_\phi^{\mathrm{lab}}(\mathbf{H}) = \mathbf{W}_{\mathrm{lab}} H_{\mathrm{pool}}^{\mathrm{lab}} + \mathbf{b}_{\mathrm{lab}} \in \mathbb{R}^n, \tag{5}$$

where $\mathbf{W}_{\mathrm{lab}} \in \mathbb{R}^{n\times h}$ and $\mathbf{b}_{\mathrm{lab}} \in \mathbb{R}^n$ correspond to the $n \times h$ head in Figure 3(a).

Vocabulary-space projection uses a (possibly different) pooling operator that aggregates one or more token representations before applying the vocabulary head:

$$H_{\mathrm{pool}}^{\mathrm{voc}} = \mathrm{Pool}_{\mathrm{voc}}(\mathbf{H}) \in \mathbb{R}^h, \tag{6}$$

where $\mathrm{Pool}_{\mathrm{voc}}(\cdot)$ may, for example, select the hidden state at a [MASK] token or combine several relevant positions. A vocabulary-level projection of size $V = |\mathcal{V}|$ is then applied, where $\mathcal{V}$ denotes the PLM's token vocabulary:

$$\mathbf{u} = \mathbf{W}_{\mathrm{vocab}} H_{\mathrm{pool}}^{\mathrm{voc}} + \mathbf{b}_{\mathrm{vocab}} \in \mathbb{R}^V, \qquad g_\phi^{\mathrm{voc}}(\mathbf{H}) = \mathrm{Extract}_{\mathcal{W}}(\mathbf{u}) \in \mathbb{R}^n, \tag{7}$$

with $\mathbf{W}_{\mathrm{vocab}} \in \mathbb{R}^{V\times h}$ and $\mathbf{b}_{\mathrm{vocab}} \in \mathbb{R}^V$. Here $\mathcal{W} = \{w_1, \ldots, w_n\} \subset \mathcal{V}$ denotes the set of label tokens, and $\mathrm{Extract}_{\mathcal{W}}(\cdot)$ is an operator that selects the n components of $\mathbf{u}$ associated with the tokens in $\mathcal{W}$.

For notational simplicity, we write both heads as single affine maps on top of pooled features; in practice, the corresponding MLP implementations may include standard non-linear activations and dropout, which we omit in the notation to focus on the choice of projection space. In summary, label-space projection maps pooled features directly to an n-dimensional label space, whereas vocabulary-space projection first projects pooled features into the V-dimensional vocabulary space and then uses only the n components associated with the label-token subset $\mathcal{W}$.

Second, input paradigm. The input paradigm specifies how the observable input string x in Equation (3) is constructed from the raw task text and an optional natural-language instruction. Let x_0 denote the raw text to be classified (e.g., the news sentence in Figure 4) and let I denote an instruction string describing the task and label set (the underlined prefix in Figure 4(b)). We use $\mathrm{Concat}(\cdot,\cdot)$ to denote concatenation in token space.

Direct input feeds the model with a templated version of the raw text only:

$$x_{\mathrm{dir}} = \tau_{\mathrm{dir}}(x_0), \tag{8}$$

where $\tau_{\mathrm{dir}}(\cdot)$ is a deterministic template function that, for example, may insert special tokens such as [CLS] or [MASK] before or after x_0. In the AG's News example of Figure 4(a), $\tau_{\mathrm{dir}}(x_0)$ instantiates the pattern "x_0. The category is [MASK]".

Instructional input is motivated by recent instruction-tuning and prompt-design practices, where models are conditioned on natural-language descriptions of the task and label space by augmenting the raw text with explicit task descriptions or label definitions. Formally, instructional input augments the same template with an explicit instruction prefix:

$$x_{\mathrm{inst}} = \tau_{\mathrm{inst}}(I, x_0) = \mathrm{Concat}\big(I,\, \tau_{\mathrm{dir}}(x_0)\big) = \mathrm{Concat}(I,\, x_{\mathrm{dir}}), \tag{9}$$

so that the model receives both the task description I and the templated raw text. Figure 4(b) illustrates this setting, where I explains that a category must be chosen and enumerates the label options, followed by the same pattern as in the direct-input case.

Under these two paradigms, the backbone operates on either x_{dir} or x_{inst} to produce contextual representations $\mathbf{H} = f_\theta(x)$. The treatment of the label text L—that is, whether labels are excluded from or included in the input—is orthogonal to this choice and is formalized separately under the label-context dimension.

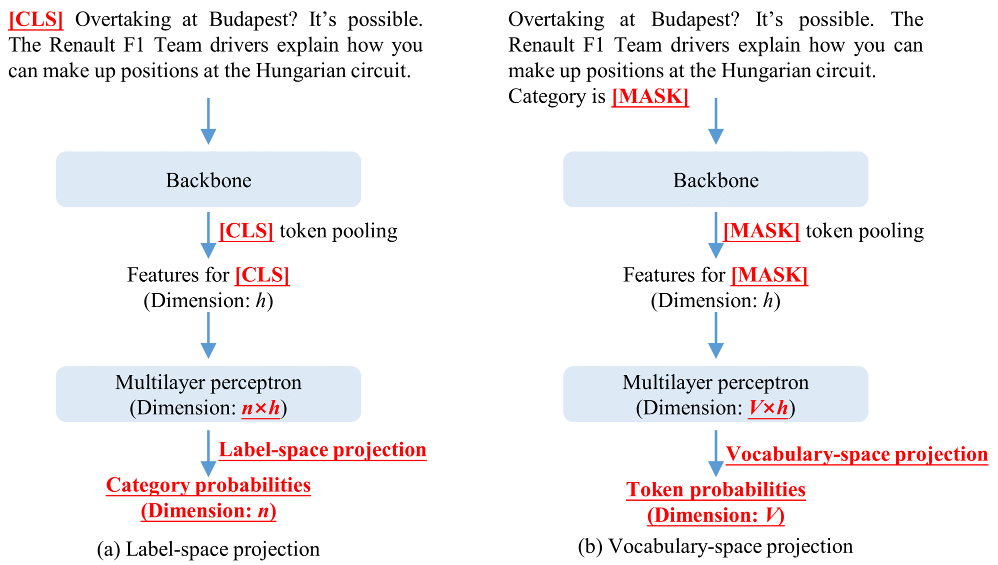

[CLS] Overtaking at Budapest? It's possible. The Renault F1 Team drivers explain how you can make up positions at the Hungarian circuit.
Backbone
[CLS] token pooling
Features for [CLS] (Dimension: h)
Multilayer perceptron (Dimension: n×h)
Label-space projection
Category probabilities (Dimension: n)
(a) Label-space projection
Overtaking at Budapest? It's possible. The Renault F1 Team drivers explain how you can make up positions at the Hungarian circuit. Category is [MASK]
Backbone
[MASK] token pooling
Features for [MASK] (Dimension: h)
Multilayer perceptron (Dimension: V×h)
Vocabulary-space projection
Token probabilities (Dimension: V)
(b) Vocabulary-space projection

Figure 3: Label-space versus vocabulary-space projection for text classification. Red underlining indicates key differences. h represents the feature dimension from the PLM backbone, n is the number of classes, and V denotes the PLM's vocabulary size. In label-space projection, the [CLS] token (commonly used in BERT) captures the sentence's contextualized representation. However, alternatives such as mean pooling or last-token pooling can also be used for sentence-level representations. In contrast, vocabulary-space projection uses the [MASK] token for predicting class labels at the token level, with its position and usage determined by the specific task and PLM design.

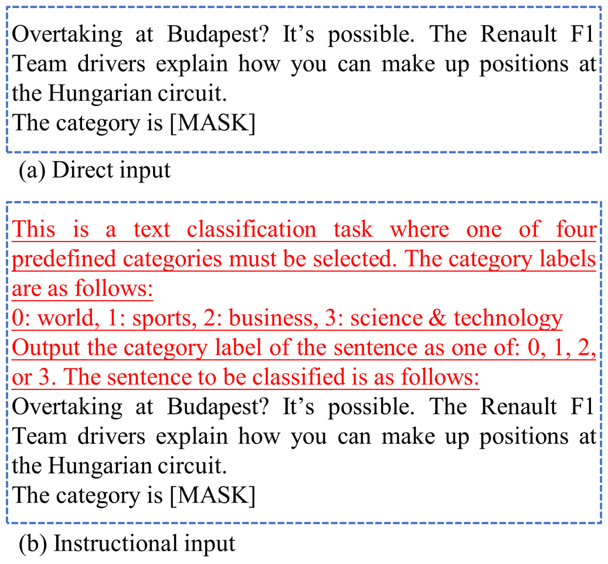
Third, label context. Label context is motivated by in-context learning (ICL), where large PLMs condition on sequences of labeled demonstrations and each label is written explicitly as text in the prompt (Figure 5(a)). We therefore ask whether, during supervised fine-tuning, exposing or hiding label text in the input context affects performance; this presence or absence of label text defines our third design dimension.

Let (x_0, y) be a supervised training pair, where x_0 is the raw text and $y \in \mathcal{Y}$ is the class index. Let $L = \ell(y)$ denote a textual realization of the label (e.g., "Positive"), obtained from a deterministic mapping $\ell : \mathcal{Y} \to \Sigma^*$. Given an input paradigm (direct or instructional), we first construct a base template $x_{\text{base}} \in \{x_{\text{dir}}, x_{\text{inst}}\}$ as defined above, and then choose one of two label-context schemes corresponding to Figure 5(b,c).

Label exclusion keeps the label text outside the model input. The backbone only receives the base template:

$$x_{\text{excl}} = x_{\text{base}}, \tag{10}$$

and the label information y (or L) is used solely on the output side when defining the training loss. This corresponds to the standard supervised fine-tuning setup illustrated in Figure 5(b), where the input contains a [MASK] (or equivalent prediction slot) but no explicit label string.



Overtaking at Budapest? It's possible. The Renault F1 Team drivers explain how you can make up positions at the Hungarian circuit.
The category is [MASK]

(a) Direct input

This is a text classification task where one of four predefined categories must be selected. The category labels are as follows:
0: world, 1: sports, 2: business, 3: science & technology
Output the category label of the sentence as one of: 0, 1, 2, or 3. The sentence to be classified is as follows:
Overtaking at Budapest? It's possible. The Renault F1 Team drivers explain how you can make up positions at the Hungarian circuit.
The category is [MASK]

(b) Instructional input

Figure 4: Direct input versus instructional input for text classification. Key differences are highlighted with red underlining; the format shown is the template used in this study and serves as an illustration.

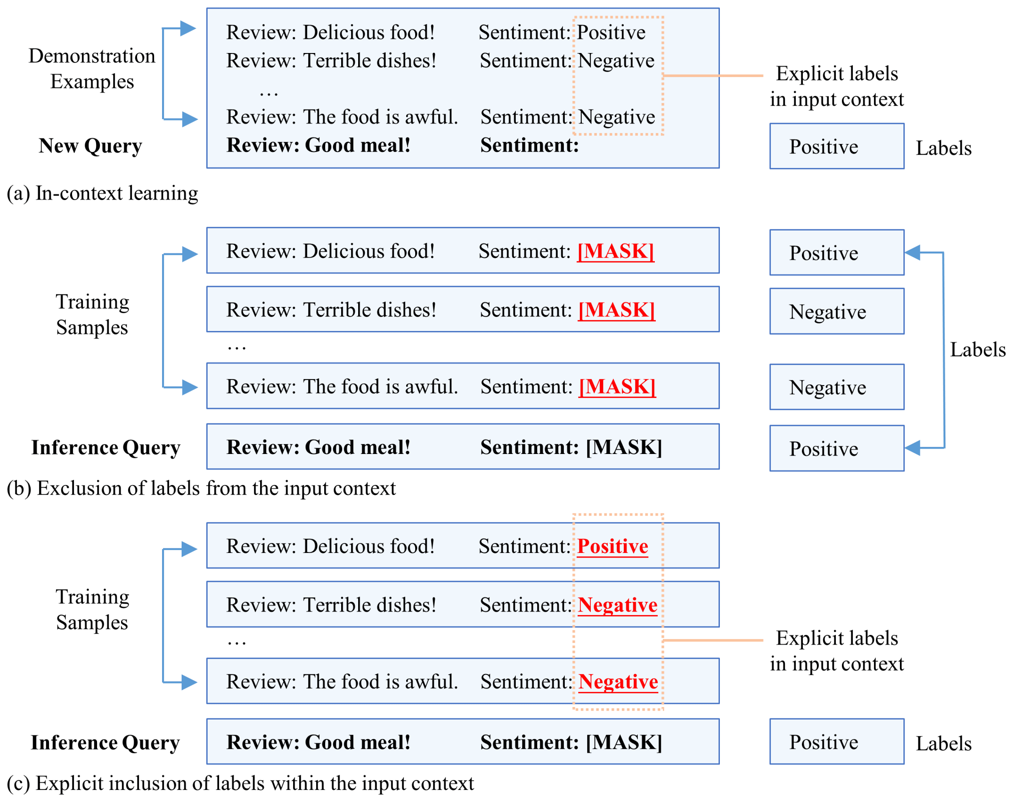

Demonstration Examples
Review: Delicious food! Sentiment: Positive
Review: Terrible dishes! Sentiment: Negative
…
Review: The food is awful. Sentiment: Negative
New Query
Review: Good meal! Sentiment:
Explicit labels in input context
Positive
Labels
(a) In-context learning
Training Samples
Review: Delicious food! Sentiment: [MASK]
Review: Terrible dishes! Sentiment: [MASK]
…
Review: The food is awful. Sentiment: [MASK]
Inference Query
Review: Good meal! Sentiment: [MASK]
Positive
Negative
Negative
Positive
Labels
(b) Exclusion of labels from the input context
Training Samples
Review: Delicious food! Sentiment: Positive
Review: Terrible dishes! Sentiment: Negative
…
Review: The food is awful. Sentiment: Negative
Explicit labels in input context
Inference Query
Review: Good meal! Sentiment: [MASK]
Positive
Labels
(c) Explicit inclusion of labels within the input context

Figure 5: Label exclusion versus explicit label inclusion in supervised fine-tuning, contextualized within the in-context learning (ICL) framework. (a) **In-context learning:** The model receives a new query appended to a sequence of demonstration examples with explicit labels in the input. (b) **Standard supervised fine-tuning:** Labels are excluded from the input; the model predicts a label from the input alone and is trained via a loss computed against gold labels. (c) **Label-included fine-tuning:** A hybrid approach that incorporates ICL-style formatting by including labels in the training input, potentially influencing model behavior via contextual alignment. Key differences between (b) and (c) are highlighted with red underlining.

Label inclusion appends a textual form of the gold label to the base template, yielding an ICL-style training input:

$$x_{\text{incl}} = \tau_{\text{label}}(x_{\text{base}}, L) = \text{Concat}(x_{\text{base}},\ L), \tag{11}$$

where $\tau_{\text{label}}(\cdot, \cdot)$ denotes a label-templating function. For example, Figure 5(c) illustrates an ICL-style pattern in which the raw text and its label are serialized as "Text: x_0 Label: L". In our experiments, τ_{label} is instantiated by such concatenations of the base template and the label text.

In all cases, Equation (3) is evaluated with x set to either x_{excl} or x_{incl}, independently of the projection space and input paradigm. This allows us to isolate the effect of exposing label text in the input context and to compare label-excluded fine-tuning with ICL-inspired, label-included fine-tuning in a controlled manner.

3.3. Combinatorial training strategies

Given the three orthogonal design dimensions formalized in Section 3.2—projection space (label vs. vocabulary), input paradigm (direct vs. instructional), and label context (exclusion vs. inclusion)—each with two alternatives, we obtain a total of $2 \times 2 \times 2 = 8$ training strategies. In all cases, the backbone f_θ maps the chosen input string x to contextual representations $\mathbf{H} = f_\theta(x)$, and the classification head g_ϕ produces an n-dimensional score vector that is turned into probabilities via Equation (3). The three design dimensions determine: (1) the choice of projection head, i.e., $g_\phi \in \{g_\phi^{\text{lab}}, g_\phi^{\text{voc}}\}$; (2) the choice of input paradigm, selecting either x_{dir} or x_{inst}; and (3) the treatment of label context, determining whether the final serialized input is x_{excl} or x_{incl}.

To instantiate this framework for our experiments, we specify loss functions for both single-label and multi-label text classification. For single-label TC, the target $y \in \mathcal{Y} = \{1, \ldots, n\}$ is a single class index. Given scores

$$\mathbf{s}(x) = g_\phi\big(f_\theta(x)\big) \in \mathbb{R}^n, \tag{12}$$

we instantiate the normalization function ρ in Equation (3) as a softmax over classes and use the standard cross-entropy loss

$$\mathcal{L}_{\text{SL}}(x, y) = -\log \Big[\rho\big(\mathbf{s}(x)\big)\Big]_y, \tag{13}$$

where $[\cdot]_y$ denotes the y-th component. This formulation applies uniformly to both projection spaces: for label-space projection we set $g_\phi = g_\phi^{\text{lab}}$, and for vocabulary-space projection we set $g_\phi = g_\phi^{\text{voc}}$ as defined in Section 3.2.

For multi-label TC, the target becomes a binary vector $\mathbf{y} \in \{0, 1\}^n$ indicating the presence or absence of each label. Let $p_k(x)$ denote the predicted probability that label k is active for input x. We then use a factorized binary cross-entropy loss

$$\mathcal{L}_{\text{ML}}(x, \mathbf{y}) = -\sum_{k=1}^{n} \Big[y_k \log p_k(x) + (1 - y_k) \log \big(1 - p_k(x)\big)\Big]. \tag{14}$$

The difference between projection spaces lies in how $p_k(x)$ is parameterized. For label-space projection, the head produces an n-dimensional score vector $\mathbf{s}(x) = g_\phi^{\text{lab}}(f_\theta(x)) \in \mathbb{R}^n$ and we set

$$p_k(x) = \sigma\big(s_k(x)\big), \quad k = 1, \dots, n, \tag{15}$$

i.e., a coordinate-wise sigmoid over the label-space scores. For vocabulary-space projection, the input template contains n prediction slots (e.g., n `[MASK]` positions), one per candidate label. At the k-th slot the vocabulary head produces logits $\mathbf{u}^{(k)}(x) \in \mathbb{R}^V$ over the vocabulary. We designate two special tokens $w^{(0)}, w^{(1)} \in \mathcal{V}$ to represent the negative and positive state of label k and compute

$$p_k(x) = \frac{\exp\left(u^{(k)}_{w^{(1)}}(x)\right)}{\exp\left(u^{(k)}_{w^{(0)}}(x)\right) + \exp\left(u^{(k)}_{w^{(1)}}(x)\right)}, \quad k = 1, \dots, n, \tag{16}$$

i.e., a two-way softmax between $\{w^{(0)}, w^{(1)}\}$ at each prediction position. Substituting these $p_k(x)$ into Equation (14) yields the multi-label loss for vocabulary-space projection. In our experiments (Section 4.1), these losses are combined with standard ℓ_2 weight decay on the trainable parameters (including the head parameters), but no additional regularizers are introduced that distinguish label-space from vocabulary-space parameterizations.

Table 1: Taxonomy of text classification strategies along orthogonal design dimensions. Each configuration specifies the serialized input x and the choice of projection head g_ϕ.

Configuration			Input x	Scores $g_\phi(\mathbf{H})$
ProjSp	InParad	LblCtx		
Lab	Dir	Excl	x_{dir}	$g_\phi^{\text{lab}}(\mathbf{H}) \in \mathbb{R}^n$
Lab	Dir	Incl	$x_{\text{incl}} = \tau_{\text{label}}(x_{\text{dir}}, L)$	$g_\phi^{\text{lab}}(\mathbf{H}) \in \mathbb{R}^n$
Lab	Inst	Excl	x_{inst}	$g_\phi^{\text{lab}}(\mathbf{H}) \in \mathbb{R}^n$
Lab	Inst	Incl	$x_{\text{incl}} = \tau_{\text{label}}(x_{\text{inst}}, L)$	$g_\phi^{\text{lab}}(\mathbf{H}) \in \mathbb{R}^n$
Voc	Dir	Excl	x_{dir}	$g_\phi^{\text{voc}}(\mathbf{H}) \in \mathbb{R}^n$
Voc	Dir	Incl	$x_{\text{incl}} = \tau_{\text{label}}(x_{\text{dir}}, L)$	$g_\phi^{\text{voc}}(\mathbf{H}) \in \mathbb{R}^n$
Voc	Inst	Excl	x_{inst}	$g_\phi^{\text{voc}}(\mathbf{H}) \in \mathbb{R}^n$
Voc	Inst	Incl	$x_{\text{incl}} = \tau_{\text{label}}(x_{\text{inst}}, L)$	$g_\phi^{\text{voc}}(\mathbf{H}) \in \mathbb{R}^n$

Note: ProjSp = projection space (Lab: label-space head g_ϕ^{lab}; Voc: vocabulary-space head g_ϕ^{voc}); InParad = input paradigm (Dir: direct input $x_{\text{dir}} = \tau_{\text{dir}}(x_0)$; Inst: instructional input $x_{\text{inst}} = \tau_{\text{inst}}(I, x_0)$); LblCtx = label context (Excl: label exclusion $x_{\text{excl}} = x_{\text{base}}$; Incl: label inclusion $x_{\text{incl}} = \tau_{\text{label}}(x_{\text{base}}, L)$ as defined in Section 3.2). Here x_0 is the raw text, I is the task instruction, and $L = \ell(y)$ is the textual form of the gold label. In all configurations, $g_\phi(\mathbf{H}) \in \mathbb{R}^n$ denotes the n-dimensional score vector used in Equation (3).

Combining the three binary design dimensions yields the eight strategies summarized in Table 1. Each row fixes (1) the projection space (label vs. vocabulary), (2) the input paradigm (direct vs. instructional), and (3) the label context (exclusion vs. inclusion), thereby determining both the serialized input x passed to the backbone and the form of the head g_ϕ used in Equation (3). These eight configurations are evaluated uniformly across encoder-only and decoder-only PLMs in Section 4, where they serve as reciprocal baselines for assessing the impact of each design choice. We adopt an operational notion of equivalence between configurations. For a fixed model–dataset pair and under the loss formulations and regularization scheme defined in this section, two configurations are said to behave equivalently if they are trained using the same hyperparameter search space and number of epochs, and the resulting evaluation metrics are statistically indistinguishable under the significance test described in Section 4.1.3. We apply this criterion when comparing the eight strategies in Section 4.

4. Experiments

This section presents a comprehensive empirical evaluation of three design dimensions in supervised text classification fine-tuning: projection space, input paradigm, and label context. We begin by detailing the experimental setup (Section 4.1). Our core analysis systematically compares the eight combinatorial strategies across four standard benchmarks to establish their relative performance (Section 4.2). We then validate the observed relationships across diverse model architectures, including larger-scale and alternative backbones (Section 4.3), and further assess their generalization to complex tasks such as multi-label and long-document classification (Section 4.4). Subsequent analyses investigate the role of instructional input by varying template styles and training data scales (Section 4.5) and provide a mechanistic analysis of label-appended supervision (Section 4.6). The findings are then synthesized and discussed (Section 4.7). All models are fine-tuned via LoRA and evaluated using macro-F1 as the primary metric, with accuracy and micro-F1 as complementary metrics for single- and multi-label tasks, respectively.

4.1. Experimental setup

4.1.1. Datasets

Table 2: Dataset summary (language, domain, task, and data splits).

Dataset	Language	Domain	Task	Train / Dev / Test	Total
AG's News	EN	News	Topic	120k / – / 7.6k	127.6k
SST-5	EN	Reviews	Sentiment	8.5k / 1.1k / 2.2k	11.8k
TNEWS	ZH	News	Topic	53.4k / 10k / –	63.4k
AEC_TC	ZH	Construction	Explainability	1.2k / 145 / 148	1.5k
HoC	EN	Biomedical	Hallmark (multi-label)	12.1k / 1.8k / 3.6k	17.5k
HPN	EN	News	Hyperpartisan (long-doc)	515 / 65 / 65	645

Note: EN = English, ZH = Chinese. Counts are in thousands (k), except where shown as raw integers. '–' indicates unavailable splits: AG's News lacks an official development set; TNEWS test set ground truth labels are not publicly released for evaluation. 'long-doc' denotes long-document classification.

Core benchmarks. We evaluate our framework on six text classification datasets selected to assess generalization across languages, domains, label structures, and input lengths. For the core analysis in Section 4.2, we use four established single-label benchmarks: AG's News [27] and SST-5 [28] (English; topic classification and fine-grained sentiment analysis), TNEWS [29] (Chinese; news categorization), and AEC_TC [30] (Chinese; domain-specific explainability classification in construction engineering).

Additional challenging benchmarks. To further assess robustness in more complex scenarios (Section 4.4), we include two specialized datasets: the Hallmarks of Cancer (HoC) corpus [31, 32], an English biomedical multi-label classification dataset with highly imbalanced label distributions where PubMed abstracts are annotated with cancer hallmarks, and

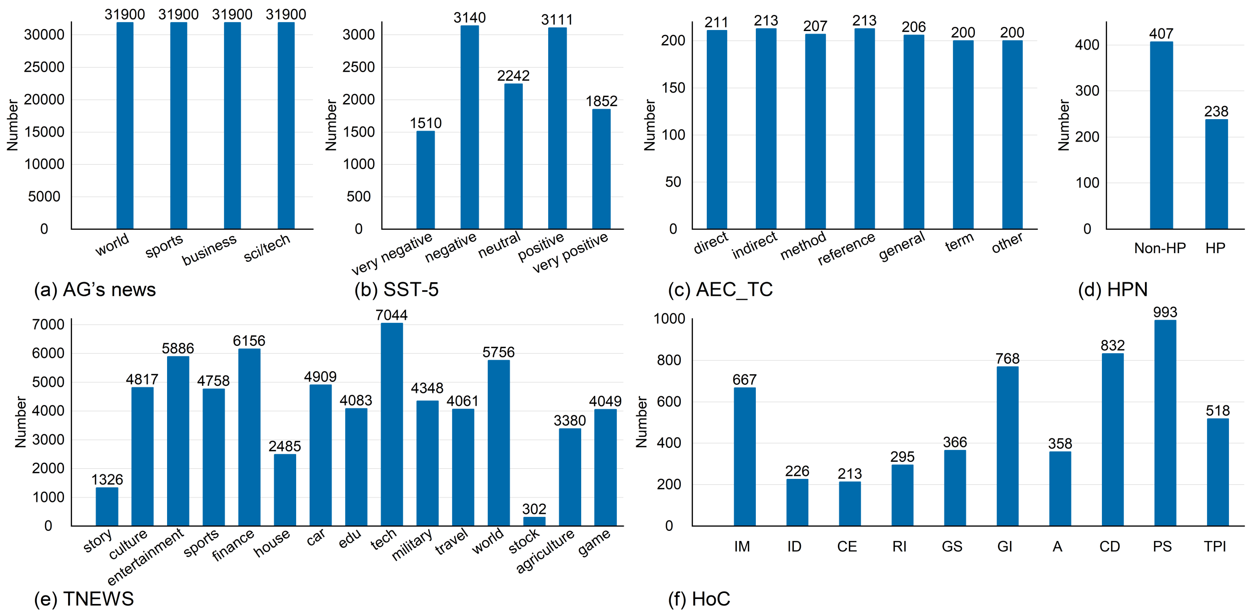


Figure 6: Category distribution across datasets. Each subfigure reports the number of samples per class label for AG's News (a), SST-5 (b), AEC_TC (c), HPN (d), TNEWS (e), and HoC (f). The plots highlight differences in label cardinality and class imbalance across datasets.

the Hyperpartisan News (HPN) dataset [33], an English long-document benchmark comprising 645 political news articles labeled for hyperpartisan bias, which we randomly split into train/validation/test subsets (515/65/65).

Collectively, these datasets cover binary to 15-way classification, single- and multi-label settings, balanced to highly imbalanced label distributions, and sentence- to document-level inputs, enabling comprehensive evaluation across diverse classification regimes. Table 2 summarizes key statistics, and Figure 6 visualizes category distributions.

4.1.2. Models and baselines

Primary configurations. Our evaluation employs all eight combinatorial strategies as reciprocal baselines using representative PLMs. Primary experiments employ BERT [4] as the encoder-only architecture (bert-base-uncased for English datasets; bert-base-chinese for Chinese datasets) and Qwen-0.5B [34] as the decoder-only representative.

Robustness verification. To ensure architectural robustness and probe scaling effects, we additionally evaluate mGTE (gte-multilingual-mlm-base) [35] as an alternative encoder-only model on AEC_TC, MiniCPM (MiniCPM-1B-sft-bf16) [36] as an additional decoder-only model on SST-5, and Qwen-14B [34] as a larger-scale decoder-only model on AEC_TC, extending our analysis to a more representative scale of contemporary LLMs. This model set provides a principled scaling trajectory while maintaining experimental feasibility.

Model specifications are summarized in Table 3, including Transformer-layer parameters (excluding embeddings and classification heads), hidden dimensions, layer depths, and Hugging Face identifiers [37]. The selected models span 85 million to 14 billion parameters, covering practical deployment scales while preserving architectural consistency within encoder-only and decoder-only categories and providing substantial scale diversity for robust comparisons.

Table 3: Specifications of pre-trained language models (PLMs) used in this study.

Model	Hugging Face ID	Params (M)	Hidden size	Layers	Architecture
BERT	google-bert/ bert-base-uncased	85.1	768	12	Encoder-only
BERT	google-bert/ bert-base-chinese	85.1	768	12	Encoder-only
Qwen-0.5B	Qwen/ Qwen2.5-0.5B	357.9	896	24	Decoder-only
mGTE	Alibaba-NLP/ gte-multilingual-mlm-base	113.3	768	12	Encoder-only
MiniCPM	openbmb/ MiniCPM-1B-sft-bf16	1247.4	1536	52	Decoder-only
Qwen-14B	Qwen/ Qwen2.5-14B	13212.9	5120	48	Decoder-only

Note: "Params" denotes the number of Transformer-layer parameters in millions (M), excluding embedding layers and task-specific MLP heads. We use bert-base-uncased for English datasets and bert-base-chinese for Chinese datasets.

4.1.3. Implementation and training details

Architecture instantiation. All models are implemented using the Hugging Face Transformers library [38]. In our implementation, only the projection space determines the model interface; the input paradigm and label context affect solely the input construction. For label-space projection, we use the sequence classification interface (AutoModelForSequenceClassification), with encoder-only architectures deriving features from the [CLS] token and decoder-only architectures from the final token. For vocabulary-space projection, encoder-only models employ AutoModelForMaskedLM and decoder-only models use AutoModelForCausalLM. During template construction, we place prediction slots at the end of the sequence. During training, these slots are filled either with a mask/placeholder token when labels are excluded or with the gold label text L when labels are included; at inference, all slots are filled with the appropriate mask/placeholder token. For encoder-only models the prediction slots are implemented using `[MASK]` tokens, whereas for decoder-only models we use analogous dedicated label-prediction tokens. We use a single slot for single-label tasks and multiple slots (one per candidate label) for multi-label classification (HoC). Figure 7 provides implementation examples for all eight configurations.

Training configuration. Fine-tuning employs LoRA [39] on the query/value projection matrices (W_q/W_v). We perform grid searches over LoRA ranks $\{8, 16, 32\}$, learning rates $\{5\times10^{-5}, 1\times10^{-4}, 3\times10^{-4}, 5\times10^{-4}, 1\times10^{-3}\}$, and batch sizes $\{4, 8, 16\}$. Fixed parameters include LoRA $\alpha = 32$, dropout $= 0.1$, and weight decay $= 0.01$. All other hyperparameters follow the defaults of Transformers v4.46.2. Each hyperparameter configuration is trained for 10 epochs, extended to 30 epochs for HoC and HPN due to their increased task complexity (multi-label classification with high class imbalance and long-document inputs, respectively). We save two checkpoints per epoch and, for each configuration, report results from the checkpoint that achieves the best validation macro-F1. The same hyperparameter search space, training schedule, and model-selection procedure are applied to all eight strategies for a given model–dataset pair, avoiding reliance on early-stopping heuristics and ensuring

	Label-space projection (Lab)	Vocabulary-space projection (Voc)
Encoder-only PLMs	**Input:** [CLS] (+ I) + x_0 (+ L) + [SEP] **Interface:** AutoModelForSequenceClassification **Prediction:** [CLS] last hidden state → logits over classes $1,\ldots,n$	**Input:** [CLS] (+ I) + x_0 + [Y] + [SEP] **Interface:** AutoModelForMaskedLM **Prediction:** [Y] last hidden state → logits over vocabulary tokens $w_1,\ldots, w_n$
Decoder-only PLMs	**Input:** (+ I) + x_0 (+ L) **Interface:** AutoModelForSequenceClassification **Prediction:** final-token last hidden state → logits over classes $1,\ldots,n$	**Input:** (+ I) + x_0 + [Y] **Interface:** AutoModelForCausalLM **Prediction:** [Y] last hidden state → logits over vocabulary tokens $w_1,\ldots, w_n$

Figure 7: Schematic mapping from projection space and model architecture to interfaces and input templates. Parentheses indicate optional segments controlled by the input paradigm and label-context settings in Table 1. In the vocabulary-space setting, $[Y]$ denotes a label-prediction slot: during training it is instantiated as a `[MASK]` token for encoder-only models (when labels are excluded) or as the gold label text L when labels are included, and as an analogous dedicated prediction token for decoder-only models; at inference, all prediction slots are filled with the appropriate mask or placeholder token. Gold label text L is used only during training and never appears in the inference inputs. For multi-label TC we use $[Y_1], \ldots, [Y_n]$ (one slot per label).

that comparisons are based on validation performance rather than training duration. For the convergence and efficiency analyses in Section 4.2.2, we further fix a single representative configuration per model–dataset pair and apply it to both label-space and vocabulary-space heads.

Evaluation protocol. Performance is evaluated using macro-F1 (M-F1) as the primary metric due to its robustness to class imbalance. For single-label tasks, we additionally report accuracy; for the multi-label HoC dataset, we also report micro-F1 (μ-F1). All results are averaged over five runs with different random seeds (from 42 to 46). Experiments primarily use NVIDIA L20 GPUs (48 GB memory), with Qwen-14B and the long-document HPN experiments run on RTX PRO 6000 GPUs (96 GB memory). Unless otherwise stated, reported values denote means; mean $\pm$ standard deviation indicates statistical dispersion. Statistical significance is assessed via paired t-tests (df $= 4$), and strategies are treated as indistinguishable from the best-performing configuration when $p > 0.005$ (significance level 0.005).

4.2. Main results

4.2.1. Overall performance across strategies

Table 4 presents macro-F1 and accuracy results under a full factorial design, enabling main-effect and interaction analyses across all eight combinatorial strategies over four benchmark datasets. The results are based on encoder-only (BERT) and decoder-only (Qwen-0.5B) architectures. Configurations that exhibit no statistically significant difference from the best-performing strategy (paired t-test, $p > 0.005$) are highlighted in bold. Within these top-performing setups, three consistent trends emerge.

First, label-space (Lab) and vocabulary-space (Voc) projections deliver statistically equivalent results. For instance, BERT achieves 94.2% macro-F1 with Lab-Dir-Excl versus 94.1% with Voc-Dir-Excl on AG's News, while Qwen scores 89.9% and 91.9% respectively on AEC_TC test, with no significant difference. This parity, observed across datasets and architectures, suggests that both projection paradigms are functionally interchangeable when sufficient supervision is available.

Second, instructional input (Inst) provides no consistent advantage over direct input (Dir) across high-performing configurations. For example, on SST-5 test, BERT yields 52.4% macro-F1 under Lab-Inst-Excl compared to 51.9% under Lab-Dir-Excl, and on TNEWS dev, Qwen achieves 56.2% under Voc-Inst-Excl versus 56.8% under Voc-Dir-Excl. These negligible differences contrast with the prominent role of instructions in few-shot ICL, indicating that explicit task prompts are largely redundant in full-data fine-tuning. We explore whether this pattern persists under reduced training data in Section 4.5.2.

Table 4: Overall performance of BERT and Qwen across eight combinatorial strategies on four datasets.

			ProjSp	Lab	Lab	Lab	Lab	Voc	Voc	Voc	Voc
Model	Dataset	Split	InParad	Dir	Dir	Inst	Inst	Dir	Dir	Inst	Inst
			LblCtx	Excl	Incl	Excl	Incl	Excl	Incl	Excl	Incl
BERT	AG's News	Test	M-F1%	**94.2±0.3**	81.7±0.6	**94.2±0.5**	84.6±0.8	**94.1±0.5**	24.9±1.8	**94.2±0.4**	16.3±1.8
			Acc.%	**94.2±0.4**	81.6±0.6	**94.2±0.5**	84.5±0.7	**94.1±0.6**	30.6±1.6	**94.2±0.5**	24.5±1.6
	SST-5	Dev	M-F1%	**51.7±0.3**	23.7±2.0	**51.7±0.4**	20.9±1.1	**51.4±0.3**	8.1±1.7	**51.2±0.6**	8.3±1.6
			Acc.%	**52.6±0.3**	30.6±1.8	**53.0±0.3**	29.1±1.2	**51.5±0.3**	25.3±1.6	**51.5±0.5**	26.3±1.6
		Test	M-F1%	**51.9±0.4**	20.5±1.9	**52.4±0.4**	19.7±1.6	**51.6±0.5**	7.5±1.5	**51.9±0.4**	8.9±1.7
			Acc.%	**52.8±0.4**	26.7±1.8	**53.2±0.3**	28.6±1.4	**52.7±0.5**	7.5±1.4	**52.9±0.5**	28.6±1.5
	TNEWS	Dev	M-F1%	**55.3±0.5**	18.5±1.9	**55.9±0.7**	19.5±1.5	**55.7±0.7**	4.2±1.3	**55.6±0.6**	3.8±1.7
			Acc.%	**56.0±0.8**	26.7±1.8	**56.5±0.7**	23.4±1.8	**56.2±0.7**	10.5±1.2	**56.1±0.5**	8.4±1.6
	AEC_TC	Dev	M-F1%	**89.7±0.6**	51.5±1.2	**89.6±0.4**	50.9±1.0	**90.3±0.3**	13.7±1.5	**91.5±0.8**	11.9±1.3
			Acc.%	**89.7±0.5**	57.2±1.2	**89.7±0.5**	55.9±0.8	**90.3±0.6**	17.2±1.4	**91.7±0.5**	17.2±1.8
		Test	M-F1%	**90.6±0.5**	48.2±1.8	**91.6±0.6**	48.8±1.7	**92.4±0.6**	10.7±1.3	**89.9±0.4**	9.6±1.4
			Acc.%	**90.5±0.3**	55.4±1.9	**91.6±0.4**	54.1±1.7	**92.4±0.5**	12.5±1.8	**89.9±0.5**	13.5±1.6
Qwen-0.5B	AG's News	Test	M-F1%	**94.8±0.5**	18.4±1.8	**94.6±0.4**	10.8±2.0	**94.3±0.5**	**94.2±0.4**	**94.3±0.4**	**94.1±0.5**
			Acc.%	**94.8±0.4**	29.3±1.6	**94.6±0.3**	21.1±1.3	**94.3±0.4**	**94.2±0.4**	**94.3±0.4**	**94.1±0.5**
	SST-5	Dev	M-F1%	**54.2±0.4**	12.6±1.8	**54.9±0.5**	10.4±0.6	**54.0±0.4**	**53.3±0.5**	**54.2±0.5**	**53.9±0.7**
			Acc.%	**56.2±0.4**	18.8±1.8	**55.7±0.6**	18.8±1.7	**56.3±0.4**	**55.3±0.4**	**54.0±0.4**	**54.1±0.8**
		Test	M-F1%	**54.2±0.5**	14.0±1.6	**54.7±0.6**	10.6±1.9	**53.9±0.4**	**54.9±0.6**	**54.6±0.5**	**54.3±0.6**
			Acc.%	**56.5±0.6**	19.6±1.8	**55.4±0.6**	19.8±1.7	**54.6±0.4**	**55.9±0.5**	**55.2±0.7**	**56.5±0.6**
	TNEWS	Dev	M-F1%	**56.8±0.7**	4.9±1.8	**57.3±0.8**	5.1±1.9	**56.8±0.6**	**56.6±0.5**	**56.2±0.3**	**56.1±0.4**
			Acc.%	**57.3±0.3**	13.4±0.7	**57.8±0.6**	9.2±1.7	**58.3±0.8**	**57.6±0.4**	**57.1±0.7**	**56.9±0.3**
	AEC_TC	Dev	M-F1%	**89.6±0.5**	16.6±1.7	**90.3±0.4**	17.2±1.2	**89.5±0.7**	**90.5±0.6**	**89.4±0.5**	**90.3±0.5**
			Acc.%	**89.7±0.5**	24.1±2.0	**90.3±0.5**	23.5±1.7	**89.7±0.8**	**90.7±0.5**	**89.7±0.6**	**90.3±0.7**
		Test	M-F1%	**89.9±0.5**	17.7±1.2	**89.4±0.5**	15.2±1.4	**91.9±0.5**	**91.6±0.7**	**91.9±0.5**	**90.1±0.6**
			Acc.%	**89.9±0.6**	27.0±1.9	**89.2±0.7**	23.2±1.4	**91.9±0.5**	**91.6±0.6**	**91.9±0.5**	**90.2±0.4**

Note: M-F1% denotes macro-averaged F1; Acc.% denotes accuracy. Strategies correspond to all 2×2×2 combinations of projection space (ProjSp $\in$ Lab, Voc), input paradigm (InParad $\in$ Dir, Inst), and label context (LblCtx $\in$ Excl, Incl), as defined in Table 1. Underlined values indicate the best-performing configuration among the eight strategies for each fixed combination of model and data split. Bolded strategies are statistically indistinguishable from the underlined best-performing configurations ($p > 0.005$, paired t-test over 5 runs, significance level 0.005).

Third, label inclusion (Incl) triggers markedly divergent behaviors between architectures, particularly under different projection spaces. In the label-space setting (Lab-Incl), BERT significantly outperforms Qwen, though both degrade relative to their optimal configurations. For instance, on AG's News, BERT scores 81.7% versus Qwen's 18.4% in macro-F1. In contrast, under vocabulary-space projection with label inclusion (Voc-Incl), Qwen maintains performance (94.1–94.2%) comparable to its optimum (94.8%), while BERT collapses to 16.3–24.9% macro-F1. This performance inversion reveals fundamental differences in how encoder-only and decoder-only architectures process label-appended sequences across projection paradigms. We further dissect these mechanisms in Section 4.6.

In summary, the results demonstrate that: (1) Projection space (Lab vs. Voc) yields equivalent performance under supervised settings; (2) Instructional input fails to offer stable benefits; and (3) The effectiveness of label inclusion is highly architecture-dependent, with opposing trends observed between BERT and Qwen. These findings underscore the need for architecture-aware alignment of design choices when fine-tuning PLMs for text classification.

4.2.2. Training convergence and computational efficiency across projection spaces

Guided by the overall comparison in Section 4.2.1, we focus here on the two configurations that consistently appear among the top-performing strategies across models and datasets, namely the label-space/direct/label-excluded (Lab/Dir/Excl) and vocabulary-space/direct/label-excluded (Voc/Dir/Excl) settings. These configurations thus serve as our final algorithmic variants when analyzing optimization dynamics and convergence behavior. To isolate the effect of the classification head on convergence and efficiency, we analyze representative model–dataset pairs under fixed training schedules, varying only the projection space, as shown in Figure 8 and Table 5. For each backbone–dataset combination, label-space and vocabulary-space heads share exactly the same training configuration; all hyperparameters follow the setup described in Section 4.1.3.

Figure 8 shows validation macro-F1, validation loss, and training global gradient norm over 10 epochs for BERT and Qwen-0.5B under Lab/Dir/Excl and Voc/Dir/Excl configurations. On both datasets, label-space and vocabulary-space variants of each backbone follow closely aligned macro-F1 trajectories and reach performance plateaus within 3–4 epochs, indicating similar convergence speeds and comparable final performance. Loss curves show dataset- and model-specific patterns (e.g., stronger overfitting for Qwen-0.5B on SST-5), but within each backbone, Lab and Voc curves remain closely aligned. Gradient norms decay from initial peaks and stabilize within comparable ranges for both projection types, demonstrating equally stable optimization dynamics.

Table 5 summarizes efficiency metrics over the same 10-epoch runs. For each model–dataset combination, training and dev throughput are of similar magnitude between Lab and Voc projections, and total wall-clock times differ only modestly within each pair. However, vocabulary-space heads consistently require more GPU memory during training and inference. This overhead is expected, as vocabulary-space projection computes logits over the full vocabulary, while label-space projection operates over the much smaller label set. Overall, these results demonstrate that for a fixed PLM backbone, the choice between label-space and vocabulary-space projection has minimal impact on convergence speed or throughput, but vocabulary-space heads incur a consistent memory penalty due to their larger output dimensionality.

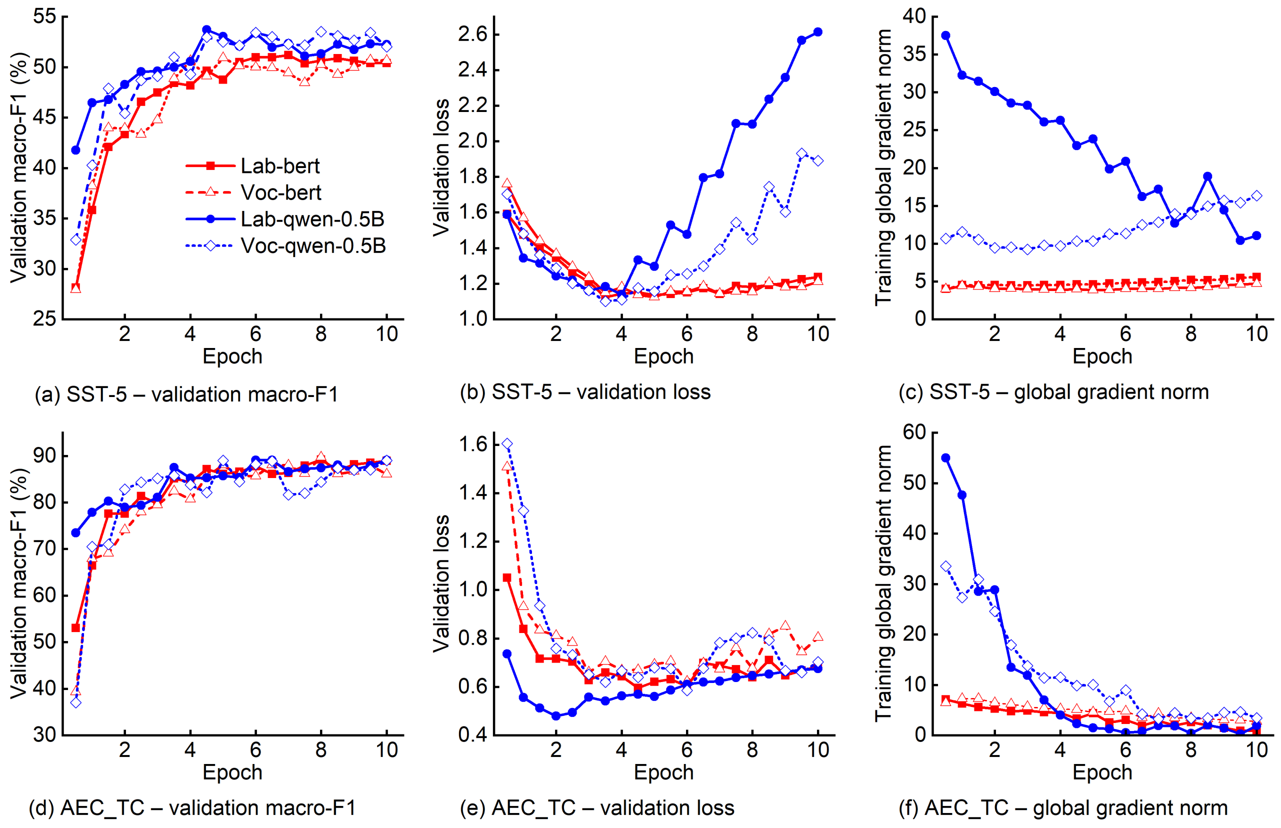


Figure 8: Training convergence of label-space vs. vocabulary-space projections on SST-5 and AEC_TC. (a–c) SST-5: validation macro-F1, validation loss, and training global gradient norm over 10 epochs. (d–f) AEC_TC: the same metrics. Each subplot compares BERT and Qwen-0.5B with label-space (Lab) and vocabulary-space (Voc) heads under identical training schedules, varying only the projection space. Validation metrics are computed on the dev split at two checkpoints per epoch; gradient norms are averaged over training steps between consecutive checkpoints.

Table 5: Training and validation efficiency of label-space vs. vocabulary-space projections on SST-5 and AEC_TC (10-epoch runs).

Dataset	Model	ProjSp	Train thr. (tokens/s)	Train mem. (GB)	Dev thr. (tokens/s)	Dev mem. (GB)	Wall time (s)
SST-5	BERT	Lab	15336	0.65	43579	0.26	336
	BERT	Voc	15746	1.04	39370	0.47	349
	Qwen-0.5B	Lab	7289	2.85	20529	1.04	660
	Qwen-0.5B	Voc	7628	5.87	20445	3.64	664
AEC_TC	BERT	Lab	11569	0.95	36642	0.26	99
	BERT	Voc	13161	1.32	39539	0.45	92
	Qwen-0.5B	Lab	5519	3.23	16387	1.07	150
	Qwen-0.5B	Voc	5874	6.31	15989	4.83	159

Note: Throughput (thr.) and memory (mem.) measurements for training (Train) and development (Dev) phases; Wall time = total wall-clock time over 10 epochs.

4.3. Cross-model validation with alternative and larger backbones

To assess the robustness of our findings beyond the BERT and Qwen-0.5B backbones, we repeat the full $2 \times 2 \times 2$ strategy evaluation on held-out datasets using alternative encoder-only and decoder-only PLMs. Specifically, we consider mGTE on AEC_TC, MiniCPM-1B on SST-5, and a larger decoder-only model, Qwen-14B, on AEC_TC. These models differ substantially in architecture, parameter count, and pretraining pipelines, and thus serve as a stress test for the generality of the trends observed in Section 4.2.

Table 6: Cross-model validation on held-out datasets using an alternative encoder-only backbone (mGTE), a mid-scale decoder-only backbone (MiniCPM), and a larger decoder-only backbone (Qwen-14B).

Model/ Dataset	Split	ProjSp InParad LblCtx	Lab Dir Excl	Lab Dir Incl	Lab Inst Excl	Lab Inst Incl	Voc Dir Excl	Voc Dir Incl	Voc Inst Excl	Voc Inst Incl
mGTE/ AEC_TC	Dev	M-F1%	**90.2**	76.7	**89.6**	73.4	**90.3**	9.2	**91.0**	9.9
		Acc.%	**90.3**	77.2	**89.7**	73.8	**90.3**	16.6	**91.0**	19.3
	Test	M-F1%	**91.9**	78.3	**91.3**	76.3	**91.1**	7.3	**91.2**	8.7
		Acc.%	**91.9**	78.4	**91.2**	76.3	**91.2**	13.5	**91.2**	12.2
MiniCPM/ SST-5	Dev	M-F1%	**56.9**	8.5	**56.8**	8.3	**56.2**	**56.4**	**58.0**	**56.8**
		Acc.%	**56.9**	26.3	**57.1**	26.2	**57.0**	**56.5**	**58.3**	**58.2**
	Test	M-F1%	**58.5**	7.4	**57.7**	8.9	**59.3**	**58.6**	**58.9**	**59.4**
		Acc.%	**59.5**	28.6	**58.4**	28.6	**60.3**	**59.5**	**60.0**	**59.7**
Qwen-14B/ AEC_TC	Dev	M-F1%	**90.5**	17.5	**91.8**	18.4	**91.5**	**90.6**	**90.9**	**91.7**
		Acc.%	**90.7**	26.5	**92.0**	29.3	**90.7**	**90.7**	**91.0**	**91.7**
	Test	M-F1%	**92.1**	17.7	**91.8**	19.5	**92.2**	**92.2**	**92.1**	**92.1**
		Acc.%	**91.9**	24.0	**91.9**	30.1	**92.2**	**92.2**	**92.2**	**92.1**

Note: See Table 4 for metric definitions and annotation conventions.

Table 6 reports macro-F1 and accuracy for all eight strategies on these backbones. Across mGTE and MiniCPM, we recover the same three core conclusions as before: label-space and vocabulary-space projections attain very similar performance among the top configurations; instructional input does not provide a consistent benefit over direct input; and label inclusion remains strongly architecture-dependent, with encoder-only models degrading under all Incl settings, while decoder-only models are robust under Voc/·/Incl but still fail under Lab/·/Incl, mirroring the behaviors observed for BERT and Qwen-0.5B. Qwen-14B extends this picture to a substantially larger decoder-only LLM: on AEC_TC, all label-excluded configurations (Lab/·/Excl and Voc/·/Excl) achieve comparable macro-F1 and accuracy, with no consistent advantage of one projection space over the other, whereas the characteristic collapse of Lab/Dir/Incl persists and vocabulary-space variants with label inclusion remain competitive.

From a representation-learning perspective, these results are consistent with viewing both heads as shallow maps on top of the same contextual representations: once the backbone is fine-tuned, most task-specific capacity resides in the shared transformer layers, so the precise parameterization of the final affine map (label logits versus vocabulary logits restricted to class tokens) has limited impact on achievable accuracy. This aligns with prior large-scale text-to-text and instruction-tuned models that cast classification as generation over verbalized labels [8, 11], yet empirically match dedicated classification heads when sufficient supervision is available. Taken together, our cross-model and cross-scale experiments suggest that the main conclusions about projection space, instruction usage, and label inclusion are intrinsic to encoder-only and decoder-only PLMs, rather than artifacts of a particular backbone size or implementation.

4.4. *Cross-dataset validation on multi-label and long-document benchmarks*

To further test the robustness of our conclusions in more challenging settings, we evaluate all eight combinatorial strategies on two additional benchmarks: HoC, a domain-specific, highly imbalanced multi-label classification task, and HPN, a long-document news classification task (Section 4.1.1). Table 7 summarizes results for encoder-only (BERT, mGTE) and decoder-only (Qwen-0.5B) backbones.

Table 7: Cross-dataset validation on HoC (multi-label, imbalanced, domain-specific) and HPN (long-document) using all eight combinatorial strategies.

Dataset	Model	Split	ProjSp InParad LblCtx	Lab Dir Excl	Lab Dir Incl	Lab Inst Excl	Lab Inst Incl	Voc Dir Excl	Voc Dir Incl	Voc Inst Excl	Voc Inst Incl
HoC	BERT	Dev	M-F1%	**72.8**	14.7	**71.0**	15.5	**71.1**	11.5	**72.5**	9.7
			μ-F1%	**84.4**	73.0	**83.9**	73.0	**83.9**	73.0	**83.9**	73.0
		Test	M-F1%	**72.2**	14.4	**72.1**	15.1	**72.3**	11.0	**72.4**	9.9
			μ-F1%	**85.1**	73.2	**84.8**	73.2	**84.8**	73.2	**84.5**	73.2
	Qwen-0.5B	Dev	M-F1%	**70.8**	10.9	**70.1**	11.2	**70.9**	**70.2**	**71.5**	**70.8**
			μ-F1%	**84.2**	73.0	**84.9**	73.0	**84.0**	**84.4**	**85.3**	**84.9**
		Test	M-F1%	**70.9**	12.0	**70.8**	11.4	**71.1**	**70.2**	**72.5**	**72.5**
			μ-F1%	**85.0**	73.2	**85.1**	73.2	**84.8**	**85.0**	**85.9**	**85.5**
HPN	mGTE	Dev	M-F1%	**90.8**	81.3	**91.8**	80.9	**91.8**	59.6	**90.5**	65.2
			Acc.%	**91.7**	82.7	**92.3**	82.6	**92.3**	64.6	**92.2**	70.8
		Test	M-F1%	**90.1**	80.9	**90.1**	80.4	**91.5**	46.2	**90.1**	61.4
			Acc.%	**90.8**	82.7	**90.8**	82.6	**92.3**	64.6	**90.8**	66.2
	Qwen-0.5B	Dev	M-F1%	**92.8**	54.0	**93.3**	59.3	**93.7**	**93.5**	**94.0**	**93.3**
			Acc.%	**93.3**	64.6	**93.9**	67.7	**94.3**	**93.9**	**94.4**	**93.9**
		Test	M-F1%	**92.9**	48.0	**93.3**	51.9	**93.0**	**93.3**	**93.3**	**93.9**
			Acc.%	**93.7**	63.1	**93.9**	63.1	**93.4**	**93.9**	**93.9**	**94.4**

Note: HoC is evaluated with macro-F1 (M-F1) and micro-F1 (μ-F1); HPN is evaluated with macro-F1 (M-F1) and accuracy (Acc.). See Table 4 for metric definitions and annotation conventions.

On the multi-label HoC benchmark, both BERT and Qwen-0.5B exhibit similar macro-F1 and micro-F1 under the best label-space and vocabulary-space configurations, confirming that the projection-space equivalence generalizes to multi-label, imbalanced biomedical classification. Instructional input again provides no consistent advantage over direct input: for both models, Inst/Excl configurations are at best on par with Dir/Excl. Notably, label inclusion leads to substantial degradation for encoder-only BERT across all projections, while decoder-only Qwen-0.5B remains competitive under Voc/·/Incl but collapses under Lab/·/Incl—replicating the architecture-dependent sensitivity observed in single-label settings.

For the long-document HPN benchmark, mGTE and Qwen-0.5B achieve comparable macro-F1 and accuracy for their best configurations, with label-space and vocabulary-space variants again performing comparably within each backbone. Consistent with previous findings, instructional input does not systematically improve performance, and label inclusion is consistently harmful for encoder-only mGTE, particularly under vocabulary-space projection where performance drops dramatically. Decoder-only Qwen-0.5B, by contrast, attains its strongest results under vocabulary-space projections, with both label-excluded and label-included variants among the top-performing strategies, while Lab/·/Incl remains unstable.

Collectively, these results demonstrate that our core findings—projection-space equivalence, limited utility of explicit instructions in supervised fine-tuning, and architecture-sensitive effects of label inclusion—extend robustly to multi-label classification with severe class imbalance and long-document understanding tasks, strengthening the external validity of our conclusions.

4.5. Instruction analysis

4.5.1. Effect of instruction template style

One possible concern is that the limited impact of instructional input observed in Sections 4.2–4.4 might be due to the specific template we use, rather than to instructions per se. To address this, we compare three instruction styles on AG's News using both BERT and Qwen-0.5B under the instructional, label-excluded setting (Lab/Inst/Excl and Voc/Inst/Excl).

Beyond our original label-style template (see Figure 4), which explicitly lists the numeric label set and asks the model to output a label ID, we evaluate two widely used alternatives. The first is a *natural-language QA-style* prompt that poses the task as a question–answer format ("Which topic is this news article about?"), following recent work on natural question prompts for classification [40]. The second is a *few-shot* template with four in-context labeled examples, adapted from standard few-shot prompt designs for text classification [41]. In all cases, only the instruction text is changed; the underlying model architectures, training schedules, and evaluation protocols are kept identical to those in Section 4.1.3.

Table 8: Effect of instruction template style on AG's News test performance under instructional, label-excluded configurations.

Model	ProjSp	Metric	Prompt style		
			Label-style (ours)	QA-style	Few-shot
BERT	Lab	M-F1%	<u>**94.2**</u>	**94.0**	**93.7**
		Acc.%	<u>**94.2**</u>	**94.0**	**93.7**
	Voc	M-F1%	<u>**94.2**</u>	**93.9**	**94.0**
		Acc.%	<u>**94.2**</u>	**93.8**	**94.0**
Qwen-0.5B	Lab	M-F1%	**94.6**	**94.6**	<u>**94.7**</u>
		Acc.%	**94.6**	**94.6**	<u>**94.7**</u>
	Voc	M-F1%	**94.3**	**94.3**	<u>**94.5**</u>
		Acc.%	<u>**94.3**</u>	<u>**94.3**</u>	**94.2**

Note: Lab and Voc denote Lab/Inst/Excl and Voc/Inst/Excl configurations, respectively. Metric conventions follow Table 4.

Table 8 reports macro-F1 and accuracy on the AG's News test set for BERT and Qwen-0.5B under instructional, label-excluded configurations. Across both models and both projection spaces, the three instruction styles yield very similar performance: differences are within about one percentage point in macro-F1 and accuracy, and no template consistently dominates the others. These results indicate that, within the supervised fine-tuning regime considered here, reasonable variations in instruction wording and the inclusion of a small number of in-context examples do not materially change the effectiveness of instructional input.

4.5.2. Effect of training data scale on instructional efficacy

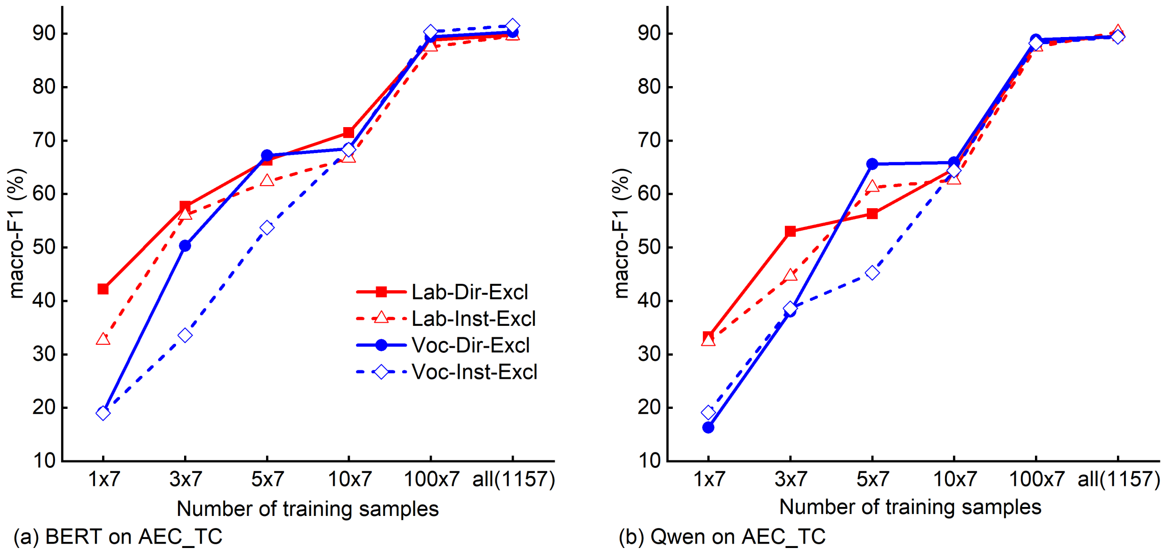


Figure 9: Macro-F1 versus training data scale for BERT and Qwen across input paradigms (Dir/Inst) and projection spaces (Lab/Voc). All configurations use label exclusion (Excl). Data scales: 7 (1×7), 21 (3×7), 35 (5×7), 70 (10×7), 700 (100×7), and 1,157 (Full) samples. Projection space (Lab: Label, Voc: Vocabulary); Input paradigm (Dir: Direct, Inst: Instructional); Label context (Excl: Exclusion, Incl: Inclusion).

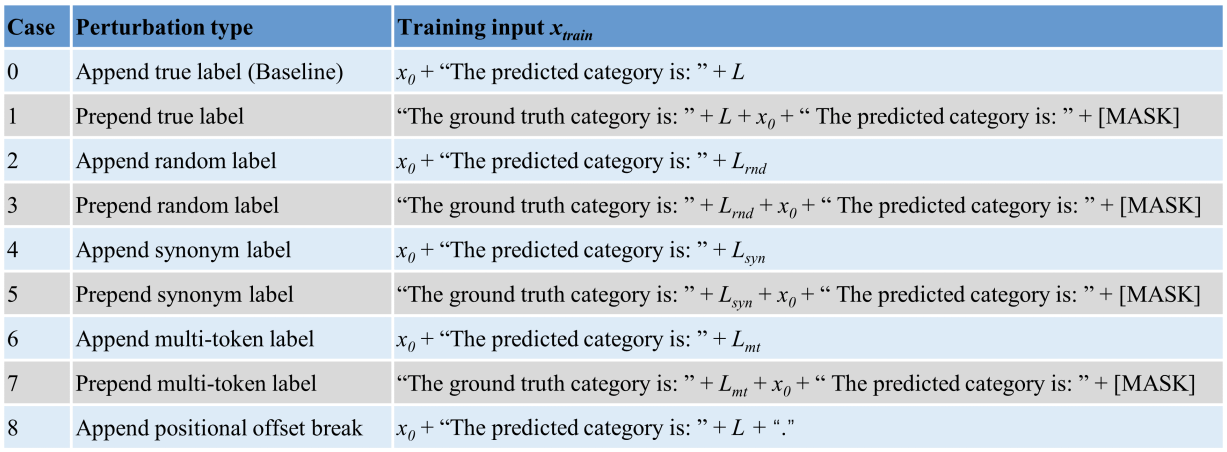
To investigate whether instructional input confers advantages under limited supervision, we run controlled experiments on the AEC_TC dataset across multiple training-set sizes. Specifically, six data regimes are constructed by uniformly sampling per class: 7 (1×7), 21 (3×7), 35 (5×7), 70 (10×7), 700 (100×7), and the full training set (1,157 samples). We evaluate two input paradigms—direct input (Dir) and instructional input (Inst)—under fixed label exclusion (Excl) with both label-space (Lab) and vocabulary-space (Voc) projections. Performance (macro-F1) of BERT and Qwen is measured across all data scales (Figure 9).

Results reveal a consistent trend: instructional input (Inst) yields no statistically significant improvement over direct input (Dir) at any data scale, from extreme few-shot (7 samples) to full-data regimes. For both BERT and Qwen, the Dir and Inst curves remain closely aligned within each projection space, and small fluctuations do not exhibit any systematic Inst>Dir advantage under our significance criterion (Section 4.1.3). This absence of gains reinforces the view that supervised fine-tuning (SFT) and in-context learning (ICL) are governed by fundamentally distinct mechanisms. While ICL critically relies on task instructions for generalization in low-resource settings, SFT demonstrates no such reliance, even under severely limited supervision.

Combined with the prompt-style experiments in Section 4.5.1, these results indicate that the limited impact of task instructions in our experiments is robust both to reasonable variations in instruction wording and to substantial changes in training data scale. This convergence of evidence supports the interpretation that the weak influence of instructions is a structural property of supervised fine-tuning on a fixed text-classification task, rather than an artifact of a particular template choice or data regime.

We tentatively interpret this pattern as a consequence of gradient-based fine-tuning allowing the model to develop task-specific representations that are only weakly tied to the precise instructional framing. One plausible explanation is that, during fine-tuning, the model learns to extract task-relevant features primarily from the labeled input–output pairs, while treating the instruction span as a largely invariant auxiliary context; the loss is applied at the prediction positions, so paraphrasing the instruction or adding a few in-context examples induces only marginal adjustments in the gradient signal. This view is consistent with prior work showing that prompt formatting can have a substantial impact in pure prompting and few-shot settings, where model parameters are frozen and behavior is driven entirely by the prompt [42], whereas in instruction-tuned or domain-adapted scenarios, performance is largely governed by the adapted parameters and further surface-level prompt tweaks often bring only limited additional gains [43]. However, we emphasize that the precise mechanisms underlying this robustness to instruction in supervised fine-tuning remain an open question and merit more targeted theoretical and empirical analysis.

4.6. Mechanistic dissection of label-appended supervision



Case	**Perturbation type**	**Training input x_{train}**
0	Append true label (Baseline)	x_0 + "The predicted category is: " + L
1	Prepend true label	"The ground truth category is: " + L + x_0 + " The predicted category is: " + [MASK]
2	Append random label	x_0 + "The predicted category is: " + L_{rnd}
3	Prepend random label	"The ground truth category is: " + L_{rnd} + x_0 + " The predicted category is: " + [MASK]
4	Append synonym label	x_0 + "The predicted category is: " + L_{syn}
5	Prepend synonym label	"The ground truth category is: " + L_{syn} + x_0 + " The predicted category is: " + [MASK]
6	Append multi-token label	x_0 + "The predicted category is: " + L_{mt}
7	Prepend multi-token label	"The ground truth category is: " + L_{mt} + x_0 + " The predicted category is: " + [MASK]
8	Append positional offset break	x_0 + "The predicted category is: " + L + "."

Figure 10: Controlled perturbations of label tokens in label-included, direct-input configurations (LblCtx = Incl, InParad = Dir). x_0 denotes the raw text, L the gold label token, L_{rnd} a randomly sampled label from the set of class labels, L_{syn} a synonym of L, and L_{mt} a multi-token paraphrase of L. During evaluation, all configurations share the same template x_0 + "The predicted category is: " + `[MASK]`; only the training inputs x_{train} differ as shown. In Case 8 (positional offset break), an additional neutral token "." is appended after the label. For decoder-only models, an analogous dedicated prediction token is used in place of `[MASK]`.

To determine whether models rely on label semantics or positional heuristics, we conduct a controlled perturbation study on AEC_TC using nine label-included, direct-input variants (Cases 0–8 in Figure 10). Case 0 reproduces our original setting with the true label appended after the text; Cases 1–3 vary label identity and position (random vs. gold label, appended vs. prepended); Cases 4–7 replace the gold label with a synonym or a multi-token paraphrase, again in appended vs. prepended form; and Case 8 appends a neutral token after the label to disrupt positional alignment while keeping the label itself unchanged. Performance for encoder-only and decoder-only models under these perturbations is reported in Table 9, with baseline comparisons drawn from Table 4.

Table 9: Performance on AEC_TC under nine label-included perturbation cases for encoder-only (BERT) and decoder-only (Qwen-0.5B) models.

Model	ProjSp	Split	Metric	Case								
				0	1	2	3	4	5	6	7	8
BERT	Lab	Dev	M-F1%	51.5	52.2	**91.0**	**90.1**	51.6	54.5	30.1	39.0	53.2
			Acc.%	57.2	55.9	**91.0**	**90.2**	56.6	57.5	38.6	44.1	56.4
		Test	M-F1%	48.2	50.2	**91.6**	**90.6**	52.9	51.4	30.0	35.6	53.6
			Acc.%	55.4	56.1	**91.6**	**90.5**	56.5	53.5	37.8	43.2	57.2
	Voc	Dev	M-F1%	13.7	11.9	**90.1**	**89.6**	10.6	13.4	42.5	45.4	10.5
			Acc.%	17.2	17.2	**90.2**	**89.7**	16.6	19.3	48.5	52.2	17.9
		Test	M-F1%	10.7	10.7	**90.4**	**91.1**	11.5	14.2	43.3	48.2	10.7
			Acc.%	12.5	14.9	**90.4**	**91.2**	18.2	21.0	50.3	53.1	18.2
Qwen-0.5B	Lab	Dev	M-F1%	16.6	60.5	86.1	**90.3**	12.4	28.9	18.9	27.5	30.0
			Acc.%	24.1	62.8	86.2	**90.3**	20.7	38.6	27.6	31.7	35.2
		Test	M-F1%	17.7	63.9	84.7	**90.6**	13.4	28.2	17.7	28.9	23.1
			Acc.%	27.0	64.9	84.5	**90.5**	23.0	40.5	28.4	33.8	33.8
	Voc	Dev	M-F1%	**90.5**	14.5	**90.2**	**91.0**	**89.8**	12.1	11.5	30.2	8.7
			Acc.%	**90.7**	19.3	**90.3**	**91.0**	**89.8**	15.2	20.7	36.6	16.5
		Test	M-F1%	**91.6**	14.3	**90.3**	**90.6**	**90.2**	10.6	14.3	34.2	7.7
			Acc.%	**91.6**	18.2	**90.4**	**90.5**	**90.2**	16.2	19.6	39.9	16.9

Note: Columns 0–8 correspond to the nine label-included perturbation cases defined in Figure 10. All configurations use direct input with label inclusion (Dir/Incl). Metric conventions follow Table 4.

Encoder-only (BERT) results. Across all nine label-included cases, encoder-only BERT is robust only when the label tokens do not convey usable information about the true class. In both label- and vocabulary-space projection, the random-label conditions (Cases 2 and 3) are the only settings whose dev/test macro-F1 values remain statistically comparable to the label-exclusion baseline in Table 4. By contrast, all configurations that expose the gold label or its semantic variants (Cases 0, 1, 4–8) yield substantially lower scores in both macro-F1 and accuracy (Table 9). This holds for both projection heads: injecting the true label, a synonym, or a lengthened (multi-token) label—whether appended or prepended—creates a shortcut that the encoder can exploit, leading to degraded generalization relative to standard label-excluded fine-tuning.

Decoder-only (Qwen) results. Qwen-0.5B is robust to label inclusion primarily under a narrow set of configurations. In the label-space setting, the prepend–random-label condition (Case 3) is the only one that remains statistically indistinguishable from the label-exclusion baseline in Table 4. The append–random-label condition (Case 2) mitigates the most severe degradation but still performs noticeably below both Case 3 and the baseline, while all configurations that expose the true label or its semantic variants (Cases 0, 1, 4–8) show clear drops in macro-F1 and accuracy. In the vocabulary-space setting, competitive performance is observed only for the random-label conditions (Cases 2 and 3) and for cases where the label is realized as a single token appended at the end of the input (Cases 0 and 4). Once this pattern is minimally perturbed—for example, by using multi-token labels, moving the label away from the final position, or appending an additional neutral token after the label (Case 8)—performance drops sharply relative to the label-exclusion baseline.

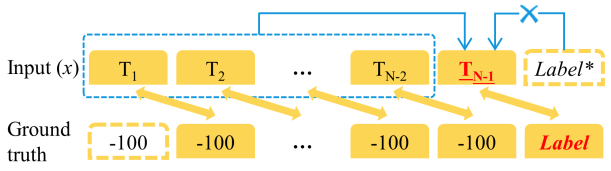
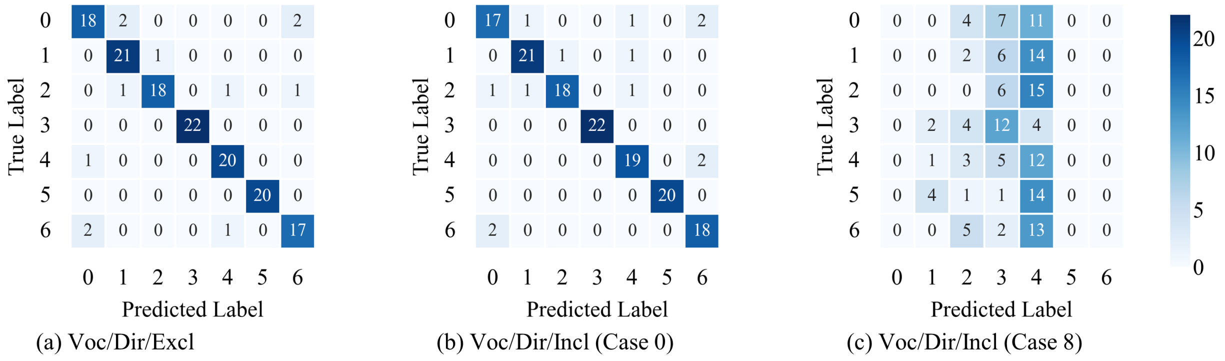


Figure 11: Offset-aligned supervision in decoder-only vocabulary-space projection with appended labels. For input sequence $[T_1, T_2, \ldots, T_{N-1}, \text{Label}^*]$, causal masking restricts attention to preceding tokens (blue arrow). The standard shifted loss aligns logits at positions $1, \ldots, N-1$ with targets where only the final position uses the gold label (others set to -100). Logits at the Label* position are discarded (blue cross), preventing the label from being both context and target—eliminating trivial copy shortcuts.

Mechanistic interpretation. The observed patterns reflect fundamental differences in supervision alignment between architectures. For encoder-only models, bidirectional attention enables exploitation of label-based shortcuts regardless of projection space. In both label-space and vocabulary-space settings, any context token reliably encoding the gold label—whether original, synonym, or paraphrase—induces the model to minimize loss through direct attention to that token rather than processing the task text. This explains the performance collapse across Cases 0, 1, 4–8, where only random labels (Cases 2–3) break the correlation and restore baseline performance.

For decoder-only models, label-space projection faces two distinct challenges: first, the inclusion of label text alters the final-token representation used for pooling, creating an input–inference mismatch; second, the model learns to exploit label tokens as positional shortcuts instead of building robust text representations. Only prepended random labels (Case 3) circumvent both issues by preserving the pooling position within task text while eliminating exploitable semantic cues. In vocabulary-space projection, decoder robustness stems from the interplay of causal masking and next-token prediction: when single-token labels are appended terminally (Cases 0, 4), the architectural coupling shown in Figure 11 prevents trivial copying—causal masking restricts attention to preceding tokens, and the offset alignment ensures label tokens are never both context and prediction targets. This forces reliance on textual context rather than enabling shortcut solutions.



(a) Voc/Dir/Excl

	0	1	2	3	4	5	6
0	18	2	0	0	0	0	2
1	0	21	1	0	0	0	0
2	0	1	18	0	1	0	1
3	0	0	0	22	0	0	0
4	1	0	0	0	20	0	0
5	0	0	0	0	0	20	0
6	2	0	0	0	1	0	17

(b) Voc/Dir/Incl (Case 0)

	0	1	2	3	4	5	6
0	17	1	0	0	1	0	2
1	0	21	1	0	1	0	0
2	1	1	18	0	1	0	0
3	0	0	0	22	0	0	0
4	0	0	0	0	19	0	2
5	0	0	0	0	0	20	0
6	2	0	0	0	0	0	18

(c) Voc/Dir/Incl (Case 8)

	0	1	2	3	4	5	6
0	0	0	4	7	11	0	0
1	0	0	2	6	14	0	0
2	0	0	0	6	15	0	0
3	0	2	4	12	4	0	0
4	0	1	3	5	12	0	0
5	0	4	1	1	14	0	0
6	0	0	5	2	13	0	0

Figure 12: Confusion matrices for Qwen-0.5B on AEC_TC under decoder-only vocabulary-space configurations. (a) Baseline (Voc/Dir/Excl); (b) Appended true label (Voc/Dir/Incl, Case 0); (c) Appended true label with trailing neutral token (Voc/Dir/Incl, Case 8).

Figure 12 and Table 10 together provide qualitative validation of our mechanistic interpretation. The confusion matrices show that the baseline (a) and appended-label Case 0 (b) maintain near-diagonal structure, while Case 8 (c) collapses predictions into few columns (no-

Table 10: Qualitative predictions of Qwen-0.5B on AEC_TC under label-excluded and label-included configurations.

Example	Text	Gold	Predicted Label		
		Label	Excl	Case 0	Case 8
E1	Dance classrooms should preferably be furnished with wooden flooring.	0	0	0	4
E2	The site shall have satisfactory engineering geological conditions.	4	4	4	4
E3	Point-type detectors should be installed horizontally.	6	6	6	4

Note: Excl = Voc/Dir/Excl (baseline); Case 0 = Voc/Dir/Incl with appended true label; Case 8 = Voc/Dir/Incl with appended label and trailing neutral token.

tably class 4). Correspondingly, the qualitative examples demonstrate that Cases 0 and Excl produce correct predictions across diverse inputs, whereas Case 8 consistently maps different texts to the same incorrect labels (e.g., class 4). Since only the positional structure varies across these configurations, this evidence confirms that decoder-only models' robustness to label-appended inputs relies on architectural alignment rather than semantic reasoning.

Core insight. Our mechanistic analysis reveals that decoder-only models' success with appended labels is architecturally determined—contingent on their causal attention and offset alignment mechanisms rather than semantic reasoning or in-context learning capabilities. This explains both the severe degradation in encoders (where full-context attention creates pervasive shortcut vulnerability) and the position-dependent performance in decoders (where success strongly depends on precise architectural alignment between label placement and prediction offset), thereby clarifying fundamental distinctions between supervised fine-tuning and in-context learning paradigms.

4.7. Discussion

Our systematic analysis yields both practical guidelines for text classification deployment and fundamental insights into PLM fine-tuning mechanisms. From an engineering perspective, we recommend label-space heads as the default choice across both encoder and decoder architectures, regardless of data availability (from low-resource to high-resource settings), due to their parameter efficiency and equivalent performance to vocabulary-space projections. Practitioners should opt for vocabulary-space projections only in multi-task systems that inherently require vocabulary-space modeling, where a vocabulary head is already present and actively used for other tasks. Instructional templates should be omitted entirely, as they incur computational costs without providing consistent accuracy gains across data regimes—from extreme few-shot (7 samples) to full-data settings—or template variations. Label exclusion remains the optimal strategy, as label inclusion introduces architecture-specific vulnerabilities: encoders suffer from pervasive shortcut exploitation due to bidirectional attention, while decoders' conditional robustness depends critically on the coupling between causal masking and next-token prediction objectives.

These findings clarify fundamental distinctions in PLM behavior. The apparent in-context learning capability observed in decoders with label-appended training is a mechanistic consequence of offset supervision in causal language modeling, not emergent semantic reasoning.

This understanding resolves the operational contrast between supervised fine-tuning and in-context learning: SFT updates parameters via gradient descent on explicit targets, making exposed labels induce harmful shortcuts, whereas ICL constructs task representations dynamically from exemplars. Consequently, label insertion during SFT creates exploitable heuristics without enhancing generalization.

5. Conclusion

This work presents a systematic study of three orthogonal design choices—projection space (label vs. vocabulary), input paradigm (direct vs. instructional), and label context (exclusion vs. inclusion)—in supervised fine-tuning for text classification. Through rigorous experimentation across six datasets and diverse architectures, we establish three core principles:

First, projection equivalence. Label-space and vocabulary-space projections deliver statistically indistinguishable performance across the encoder-only and decoder-only models and datasets considered in this study. The primary distinction is efficiency: vocabulary-space heads incur higher memory cost due to full-vocabulary logits, whereas label-space heads are more suitable for compact, dedicated classifiers.

Second, instructional irrelevance. Explicit task instructions, despite their utility in few-shot in-context learning, yield no consistent gains across all data regimes of supervised fine-tuning.

Third, label inclusion triggers positional shortcuts that catastrophically degrade encoder performance. For decoder-only models, robustness arises mainly when labels are positioned at the end of the sentence, due to the interaction between causal attention masks and next-token offset alignment, rather than from genuine semantic reasoning over label text. This contrast shows that the observed robustness is an architectural artifact rather than genuine semantic reasoning.

Taken together, these findings support simple design recommendations for practitioners. For standalone classifiers, label-space heads should be the default choice, with vocabulary-space projections reserved mainly for settings that already reuse vocabulary heads in multi-task or text-to-text systems. Instructional templates can be omitted from SFT pipelines, as they add computational overhead without providing robust gains. Likewise, label tokens are best kept out of the input, making label exclusion a stable, architecture-agnostic option for supervised text classification.

Limitations and future work. Despite covering six datasets across two languages and multiple regimes (single- vs. multi-label, short vs. long documents, general vs. domain-specific text), our study still operates on a relatively narrow slice of the possible design space.

First, dataset diversity remains limited: we do not consider highly specialized or safety-critical domains such as legal, financial, or clinical decision support, nor low-resource or morphologically rich languages, where supervision design choices may interact more strongly with domain shift and annotation scarcity.

Second, while our models span 85M to 14B parameters, they stop short of frontier-scale LLMs and mixture-of-experts architectures; future work should test whether projection equivalence and the limited role of instructions persist in models with stronger emergent reasoning abilities and more complex training pipelines.

Third, our experiments focus on standard LoRA-based supervised fine-tuning for closed-set classification; extending the analysis to other adaptation mechanisms (e.g., prefix tuning, full-parameter fine-tuning, retrieval-augmented or semi-supervised setups) and to more interactive task formats (e.g., rationales, chain-of-thought supervision) would help clarify how robust our conclusions are beyond the text classification setting. Addressing these limitations will further sharpen the methodological role of SFT as a baseline for adapting PLMs in diverse real-world applications.

6. Acknowledgments

This work was supported by National Natural Science Foundation of China (No. 62172042) and Major Scientific and Technological Innovation Projects of Shandong Province (2020CXGC010116).

References

[1] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, J. Gao, Deep learning–based text classification: a comprehensive review, ACM Computing Surveys 54 (3) (2021) 1–40. doi:10.1145/3439726.

[2] S. Kasmaiee, S. Kasmaiee, M. Homayounpour, Correcting spelling mistakes in Persian texts with rules and deep learning methods, Scientific Reports 13 (1) (2023) 19945. doi:10.1038/s41598-023-47295-2.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, NIPS'17, Curran Associates Inc., Red Hook, NY, USA, 2017, pp. 6000–6010. doi:10.5555/3295222.3295349.

[4] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 4171–4186. doi:10.18653/v1/N19-1423.

[5] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, Improving language understanding by generative pre-training, Tech. rep., OpenAI (2018).

[6] Y. Wu, J. Wan, A survey of text classification based on pre-trained language model, Neurocomputing 616 (2025) 128921. doi:10.1016/j.neucom.2024.128921.

[7] Z. Liu, W. Lin, Y. Shi, J. Zhao, A robustly optimized bert pre-training approach with post-training, in: China national conference on Chinese computational linguistics, Springer, 2021, pp. 471–484. doi:10.1007/978-3-030-88480-2_35.

[8] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, et al., Exploring the limits of transfer learning with a unified text-to-text transformer, Journal of machine learning research 21 (140) (2020) 1–67.

[9] G. Cui, S. Hu, N. Ding, L. Huang, Z. Liu, Prototypical verbalizer for prompt-based few-shot tuning, in: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Dublin, Ireland, 2022, pp. 7014–7024. doi:10.18653/v1/2022.acl-long.483.

[10] Z. Li, X. Li, Y. Liu, H. Xie, J. Li, F.-l. Wang, Q. Li, X. Zhong, Label supervised llama finetuning, arXiv preprint arXiv:2310.01208 (2023). doi:10.48550/arXiv.2310.01208.

[11] J. Wei, M. Bosma, V. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, Q. V. Le, Finetuned language models are zero-shot learners, in: International Conference on Learning Representations (ICLR), 2022.

[12] Q. Dong, L. Li, D. Dai, C. Zheng, J. Ma, R. Li, H. Xia, J. Xu, Z. Wu, B. Chang, X. Sun, L. Li, Z. Sui, A survey on in-context learning, in: Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Miami, Florida, USA, 2024, pp. 1107–1128. doi:10.18653/v1/2024.emnlp-main.64.

[13] Y. Wang, W. Wang, Q. Chen, K. Huang, A. Nguyen, S. De, Zero-shot text classification with knowledge resources under label-fully-unseen setting, Neurocomputing 610 (2024) 128580. doi:10.1016/j.neucom.2024.128580.

[14] M. Chen, H. Fu, C. Liu, X. J. Wang, Z. Li, J. Sun, Build a good human-free prompt tuning: Jointly pre-trained template and verbalizer for few-shot classification, IEEE Transactions on Knowledge and Data Engineering (2025). doi:10.1109/TKDE.2025.3543422.

[15] Y. Wang, W. Wang, Q. Chen, K. Huang, A. Nguyen, S. De, Prompt-based zero-shot text classification with conceptual knowledge, in: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop), Association for Computational Linguistics, Toronto, Canada, 2023, pp. 30–38. doi:10.18653/v1/2023.acl-srw.4.

[16] S. Zhang, L. Dong, X. Li, S. Zhang, X. Sun, S. Wang, J. Li, R. Hu, T. Zhang, F. Wu, G. Wang, Instruction tuning for large language models: A survey, arXiv preprint arXiv:2308.10792 (2025). doi:10.48550/arXiv.2308.10792.

[17] R. Lou, K. Zhang, W. Yin, Large Language Model Instruction Following: A Survey of Progresses and Challenges, Computational Linguistics 50 (3) (2024) 1053–1095. doi:10.1162/coli_a_00523.

[18] M. J. J. Bucher, M. Martini, Fine-tuned 'small' llms (still) significantly outperform zero-shot generative ai models in text classification, arXiv preprint arXiv:2406.08660 (2024). doi:10.48550/arXiv.2406.08660.

[19] A. Edwards, J. Camacho-Collados, Language models for text classification: Is in-context learning enough?, in: Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024), ELRA and ICCL, Torino, Italia, 2024, pp. 10058–10072.

[20] M. Mosbach, T. Pimentel, S. Ravfogel, D. Klakow, Y. Elazar, Few-shot fine-tuning vs. in-context learning: A fair comparison and evaluation, in: Findings of the Association for Computational Linguistics: ACL 2023, Association for Computational Linguistics, Toronto, Canada, 2023, pp. 12284–12314. doi:10.18653/v1/2023.findings-acl.779.

[21] X. Sun, X. Li, J. Li, F. Wu, S. Guo, T. Zhang, G. Wang, Text classification via large language models, in: Findings of the Association for Computational Linguistics: EMNLP 2023, Association for Computational Linguistics, Singapore, 2023, pp. 8990–9005. doi:10.18653/v1/2023.findings-emnlp.603.

[22] S. Min, X. Lyu, A. Holtzman, M. Artetxe, M. Lewis, H. Hajishirzi, L. Zettlemoyer, Rethinking the role of demonstrations: What makes in-context learning work?, in: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 2022, pp. 11048–11064. doi:10.18653/v1/2022.emnlp-main.759.

[23] J. Liu, D. Shen, Y. Zhang, B. Dolan, L. Carin, W. Chen, What makes good in-context examples for GPT-3?, in: Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures, Association for Computational Linguistics, Dublin, Ireland and Online, 2022, pp. 100–114. doi:10.18653/v1/2022.deelio-1.10.

[24] C. W. F. Mayer, S. Ludwig, S. Brandt, Prompt text classifications with transformer models! an exemplary introduction to prompt-based learning with large language models, Journal of Research on Technology in Education 55 (2022) 125 – 141. doi:10.1080/15391523.2022.2142872.

[25] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, D. Krishnan, Supervised contrastive learning, in: Advances in Neural Information Processing Systems, Vol. 33, Curran Associates, Inc., 2020, pp. 18661–18673.

[26] Q. Le, T. Mikolov, Distributed representations of sentences and documents, in: Proceedings of the 31st International Conference on Machine Learning, Vol. 32, PMLR, Beijing, China, 2014, pp. 1188–1196.

[27] X. Zhang, J. Zhao, Y. LeCun, Character-level convolutional networks for text classification, in: Proceedings of the 29th International Conference on Neural Information Processing Systems, NIPS'15, MIT Press, Montreal, Canada, 2015, pp. 649–657. doi:10.5555/2969239.2969312.

[28] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, C. Potts, Recursive deep models for semantic compositionality over a sentiment treebank, in: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Seattle, Washington, USA, 2013, pp. 1631–1642. doi:10.18653/v1/D13-1170.

[29] L. Xu, H. Hu, X. Zhang, L. Li, C. Cao, Y. Li, et al., Clue: A chinese language understanding evaluation benchmark, in: Proceedings of the 28th International Conference

on Computational Linguistics, International Committee on Computational Linguistics, Barcelona, Spain (Online), 2020, pp. 4762–4772. doi:10.18653/v1/2020.coling-main.419.

[30] Z. Zheng, Y.-C. Zhou, K.-Y. Chen, X.-Z. Lu, Z.-T. She, J.-R. Lin, A text classification-based approach for evaluating and enhancing the machine interpretability of building codes, Engineering Applications of Artificial Intelligence 127 (2024) 107207. doi:10.1016/j.engappai.2024.107207.

[31] S. Baker, I. Silins, Y. Guo, I. Ali, J. Högberg, U. Stenius, A. Korhonen, Automatic semantic classification of scientific literature according to the hallmarks of cancer, Bioinformatics 32 (3) (2016) 432–440. doi:10.1093/bioinformatics/btv585.

[32] J. A. Fries, L. Weber, N. Seelam, G. Altay, D. Datta, R. Su, et al., BIGBIO: a framework for data-centric biomedical natural language processing, in: Proceedings of the 36th International Conference on Neural Information Processing Systems (NeurIPS 2022), Curran Associates Inc., Red Hook, NY, USA, 2022, pp. 1870:1–1870:15. doi:10.5555/3600270.3602140.

[33] J. Kiesel, M. Mestre, R. Shukla, E. Vincent, P. Adineh, D. Corney, B. Stein, M. Potthast, SemEval-2019 task 4: Hyperpartisan news detection, in: Proceedings of the 13th International Workshop on Semantic Evaluation, Association for Computational Linguistics, Minneapolis, Minnesota, USA, 2019, pp. 829–839. doi:10.18653/v1/S19-2145.

[34] QwenTeam, Qwen2.5: A party of foundation models, `https://qwenlm.github.io/blog/qwen2.5/`, accessed: 2025-7-7 (September 2024).

[35] X. Zhang, Y. Zhang, D. Long, W. Xie, Z. Dai, J. Tang, H. Lin, B. Yang, P. Xie, F. Huang, M. Zhang, W. Li, M. Zhang, mGTE: Generalized long-context text representation and reranking models for multilingual text retrieval, in: Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track, Association for Computational Linguistics, Miami, Florida, US, 2024, pp. 1393–1412. doi:10.18653/v1/2024.emnlp-industry.103.

[36] S. Hu, Y. Tu, X. Han, et al., Minicpm: Unveiling the potential of end-side large language models, `https://openbmb.vercel.app/minicpm-en`, accessed: 2025-7-7 (2024).

[37] Hugging Face, Transformers: State-of-the-art natural language processing, `https://github.com/huggingface/transformers`, accessed: 2025-7-7 (2025).

[38] T. Wolf, L. Debut, V. Sanh, et al., Transformers: State-of-the-art natural language processing, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computational Linguistics, Online, 2020, pp. 38–45. doi:10.18653/v1/2020.emnlp-demos.6.

[39] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, LoRA: Low-rank adaptation of large language models, in: Proceedings of the 10th International Conference on Learning Representations (ICLR), 2022.

[40] D. Ramati, D. Gottesman, M. Geva, Eliciting textual descriptions from representations of continuous prompts, in: Findings of the Association for Computational Linguistics: ACL 2025, Association for Computational Linguistics, Vienna, Austria, 2025, pp. 16545–16562. doi:10.18653/v1/2025.findings-acl.849.

[41] T. Gao, A. Fisch, D. Chen, Making pre-trained language models better few-shot learners, in: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Association for Computational Linguistics, Online, 2021, pp. 3816–3830. doi:10.18653/v1/2021.acl-long.295.

[42] M. Sclar, Y. Choi, Y. Tsvetkov, A. Suhr, Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting, in: Proceedings of the 12th International Conference on Learning Representations (ICLR), 2024.

[43] O. Rohanian, M. Nouriborji, S. Kouchaki, F. Nooralahzadeh, L. Clifton, D. A. Clifton, Exploring the effectiveness of instruction tuning in biomedical language processing, Artificial Intelligence in Medicine 158 (2024) 103007. doi:10.1016/j.artmed.2024.103007.