HYDRA: A Multi-Head Encoder-only Architecture for Hierarchical Text Classification

Fabian Karl
Universität Ulm, Germany
fabian.karl@uni-ulm.de

Ansgar Scherp
Universität Ulm, Germany
ansgar.scherp@uni-ulm.de

Abstract

We introduce HYDRA, a simple yet effective multi-head encoder-only architecture for hierarchical text classification that treats each level in the hierarchy as a separate classification task with its own label space. State-of-the-art approaches rely on complex components like graph encoders, label semantics, and autoregressive decoders. We demonstrate that such complexity is often unnecessary. Through parameter sharing and level-specific parameterization, HYDRA enables flat models to incorporate hierarchical awareness without architectural complexity. Experiments on four benchmarks (NYT, RCV1-V2, BGC, and WOS) demonstrate that HYDRA always increases the performance over flat models and matches or exceeds the performance of complex state-of-the-art methods. The source code is available at https://github.com/FKarl/HYDRA

1 Introduction

Hierarchical text classification (HTC) is at the core of many real-world applications, such as news categorization, book genre classification, and scientific indexing, which depend on assigning documents to categories organized within taxonomies (Zangari et al., 2024). Recent state-of-the-art approaches increasingly rely on sophisticated architectural components. These include graph encoders to capture hierarchical structure (Wang et al., 2022), semantic label embeddings to model label interdependencies (Zhou et al., 2025), and autoregressive decoders to generate label sequences (Younes et al., 2024). While these HTC-specific designs achieve impressive classification performance, they have high architectural complexity. This reduces practical applicability and reproducibility. We question whether such sophisticated components are truly necessary for effective hierarchical classification.

In this work, we introduce **HYDRA** (HierarchY Divided classifieR Architecture), a simple yet effective method for HTC. HYDRA combines a shared text encoder architecture with distinct classification heads for each hierarchical level, enabling encoder-only models like BERT (Devlin et al., 2019) to integrate structural awareness without adding architectural complexity. By treating hierarchical classification as a multi-task problem, HYDRA combines parameter efficiency through shared representations with level-specific parameterization. In contrast to state-of-the-art HTC models, HYDRA does not require a graph encoder, label semantics, or autoregressive decoder. As shown in Figure 1, our approach retains the simplicity of encoder-only models while incorporating hierarchical information through level-wise training signals. Experiments demonstrate that HYDRA increases the performance over a range of strong encoder-only models and even matches or exceeds that of more complex state-of-the-art methods on standard HTC benchmarks (NYT, RCV1-V2, BGC, and WOS). Our key contributions are:

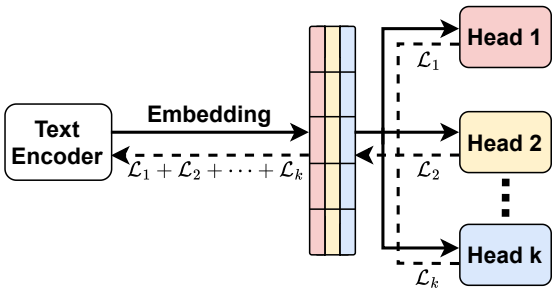

Figure 1: Our HYDRA architecture uses multiple classification heads with a shared embedding to make flat encoder-only models hierarchically aware without complex modeling. It is trained with k training signals, each guided by a different level of the hierarchy. Treating HTC as a multi-task problem effectively incorporates hierarchical information into flat encoder-only models.

- A simple but effective multi-head architecture HYDRA that enables flat models to effectively leverage hierarchical label information.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 9292–9303
November 4-9, 2025

- HYDRA matches or exceeds state-of-the-art performance on multiple benchmarks without relying on graph encoders, label semantics, or autoregressive decoders.
- We propose three different setups of HYDRA: Local Heads Only, Local Heads + Global Head, and Local Heads + Nested Head.

Below, we summarize the related work. Section 3 provides a problem statement and introduces our method. The experimental apparatus is described in Section 4. An overview of the results achieved is reported in Section 5. Section 6 discusses the results, before we conclude.

2 Related Work

2.1 Hierarchical Text Classification

Hierarchical text classification (HTC) addresses the challenge of assigning documents to labels organized in taxonomies with multiple levels of granularity. Existing approaches for HTC can be categorized into *flat*, *local*, and *global* approaches (Zangari et al., 2024).

The flat classification approach treats the hierarchical task as a flat multi-label classification by discarding the hierarchical information entirely (Younes et al., 2024). For example, in news categorization, a flat approach would treat all categories like "Politics", "Sports", and "Sports/Volleyball" as independent labels, disregarding their hierarchical relationships. Despite considerable advances in HTC, the field has largely overlooked flat models as standalone approaches. Most existing works only use BERT as a baseline without investigating newer and more powerful encoder-only models (Zhou et al., 2025; Liu et al., 2025; Wang et al., 2022). While HTC approaches incorporate encoder-only models within their architectures, they typically use them as a component in a complex system. This can be the use of an encoder-only model as underlying encoders for the graph structure (Jiang et al., 2022), as a text encoder with a separate graph encoder (Wang et al., 2022), or paired with an autoregressive decoder (Younes et al., 2024). In contrast, we rely on a single encoder-only model with a loss per hierarchy level (see Figure 1).

Local methods train independent classifiers per hierarchy level (Wehrmann et al., 2018; Shimura et al., 2018) or node (Banerjee et al., 2019). A local approach might train separate classifiers to distinguish between "Politics" and "Sports" and to distinguish between "Volleyball" and "Football".

Research focus has generally shifted toward global models, with practically all state-of-the-art models (U et al., 2023; Zhou et al., 2025; Jiang et al., 2022) adopting global approaches to incorporate hierarchical information. The global approach extends the flat classification paradigm by enriching it with hierarchical information, such as that "Volleyball" is a subcategory of "Sports". This can be done in multiple ways.

For example, HiAGM (Zhou et al., 2020) formulates the label hierarchy as a directed graph and uses structure encoders to capture label relations. Building on this idea, HGCLR (Wang et al., 2022) uses hierarchy-guided contrastive learning to inject hierarchical structure into the text encoder, allowing it to produce hierarchy-aware embeddings. HALB (Zhang et al., 2024) extends HGCLR by adding multi-label negative supervision and replacing the classification loss with asymmetric loss to achieve a hierarchy-aware and label-balanced model. Addressing label conflicts, DFG (Liu et al., 2025) introduces a two-stage approach: disentangling label features to eliminate unintended correlations, then selectively reconstructing hierarchical connections through a GNN-encoded graph. Other approaches focus on semantic alignment between texts and labels. HiMatch (Chen et al., 2021) treats hierarchical classification as a semantic matching problem, projecting texts and labels into a shared embedding space and optimizing their alignment across hierarchy levels. Similarly, HBGL (Jiang et al., 2022) combines global and local hierarchical structures by jointly modeling a static global hierarchy and text-specific local hierarchies using pre-trained language models.

Contrastive learning techniques are further explored by HILL (Zhu et al., 2024), which preserves hierarchical information through an information lossless contrastive framework that fuses the hierarchy with textual features without degrading original information. HGBL (Zhang et al., 2025) further enhances label-text interaction by guiding the contrastive learning based on the global features extracted by a BiLSTM (Graves and Schmidhuber, 2005). HJCL (U et al., 2023) uses an instance-based contrastive loss in combination with label-based contrastive learning to incorporate the hierarchical structure. Another contrastive learning approach is HiSR (Zhou et al., 2025), which generates

various negative samples to improve the model's ability to distinguish between fine-grained labels.

Previous work also explores the use of an autoregressive decoder for HTC. Architectures like Seq2Tree (Yu et al., 2022) extend T5 (Raffel et al., 2020) by capturing the hierarchical information using Depth-First Search (Tarjan, 1972) over the hierarchy. RADAr (Younes et al., 2024) takes a different approach. It uses RoBERTa as an encoder and a custom autoregressive decoder. Unlike other models, RADAr does not explicitly encode the label hierarchy, demonstrating that explicit hierarchy encoding is not always necessary.

2.2 Multi-Task Learning

Multi-Task Learning (MTL) (Caruana, 1997) is a paradigm in which a model is trained to perform multiple related tasks simultaneously. In neural networks, this is typically achieved by sharing parameters in the lower layers while maintaining task-specific output layers (Zhang et al., 2023). The main advantages of MTL include improved data efficiency, reduced risk of overfitting, and the ability to take advantage of auxiliary information from related tasks (Crawshaw, 2020).

A key distinction in MTL approaches is between hard parameter sharing, where multiple tasks share the same parameters for certain layers, and soft parameter sharing, where each task has its own parameters, but similarity between them is encouraged via regularization (Chen et al., 2024). Hard sharing acts as a strong regularizer, while soft sharing offers more flexibility for tasks that are less closely related. Effective multi-task learning generally requires sufficient relatedness between the component tasks and their data distributions (Zhang et al., 2023).

3 HYDRA: A Hierarchical Multi-Head Architecture

Current state-of-the-art approaches for hierarchical text classification do not use local classifiers and often rely on complex global models that incorporate graph encoders, label semantics, or autoregressive decoders (see Table 1). However, we question whether such complexity is truly necessary for effective HTC. To answer this, we propose HYDRA (HierarchY Divided classifieR Architecture), a simple yet effective local model. The key idea behind HYDRA is to train specialized classifier heads for each hierarchical level while utilizing hard parameter sharing for the encoder. By treating HTC as a multi-task problem, HYDRA leverages parameter sharing to incorporate hierarchical information into flat encoder-only models.

Model	Local-Classifier?	No Label Semantics?	No Graph Encoder?	No Autoregressive Decoder?
HiAGM	✗	✓	✗	✗
HiMatch	✗	✗	✗	✓
HGCLR	✗	✗	✗	✓
HILL	✗	✗	✗	✓
HBGL	✗	✗	✗	✓
HALB	✗	✗	✗	✓
HGBL	✗	✗	✗	✓
DFG	✗	✗	✗	✓
HJCL	✗	✗	✗	✓
HiSR	✗	✗	✗	✓
Seq2Tree	✗	✗	✓	✗
RADAr	✗	✓	✓	✗
HYDRA (Ours)	✓	✓	✓	✓

Table 1: Model characteristics across key architectural components. HYDRA distinguishes itself by avoiding complex components while using local classifiers.

3.1 Task Formulation

In hierarchical text classification, the label space is naturally organized into a hierarchy with multiple levels. We approach this problem by treating each level in the hierarchy as a separate classification sub-task that provides a complementary perspective on the full classification task. This multi-task formulation allows us to incorporate hierarchical awareness into flat models without complex architectural components. Formally, a sub-task is a tuple $s_j = (\mathcal{L}_j, L_j)$ consisting of a loss function $\mathcal{L}_j$ and a label subset L_j drawn from the full label space L at the j^{th} level of the hierarchy. The training objective of HYDRA is then defined as a set $S = \{s_1, \ldots, s_k\}$ with k being the number of hierarchy levels, forming the sub-tasks that provide complementary supervision during training. Unlike traditional multi-task learning, where tasks are truly separate, our hierarchies represent different views of the same underlying classification task, organized at different levels of granularity.

3.2 Architecture Design

HYDRA consists of a pre-trained text encoder that encodes an input text sequence into a fixed-dimensional latent representation. We then apply a learnable linear embedding projection that expands this embedding by a factor equal to the number of hierarchy levels. Finally, this is then forwarded to multiple classification heads.

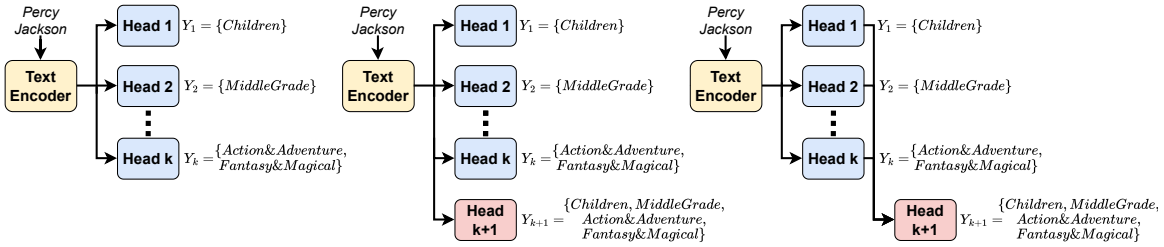

Figure 2: Our three proposed architectural setups for HYDRA at the example of classes assigned to a Percy Jackson fantasy book. All three setups extend on a shared encoder that generates a unified text embedding. The difference is in how the classification heads are applied: (a) Local Heads Only: Multiple local classification heads, each for one hierarchy level. (b) Local Heads + Global Head: Adds a global classification head over the full label set to align representations across levels. (c) Local Heads + Nested Head: The global head takes the aggregated outputs of the local heads as input rather than the embedding directly.

We propose three different setups for HYDRA, as illustrated in Figure 2. First, the Local Heads Only setup (Figure 2a), where each hierarchical level has its own classifier head. For example, with a book from the Percy Jackson series, the first level might classify it as "Children's Literature", the second level as "Middle Grade", and the third level as both "Action&Adventure", and "Fantasy&Magic".

Second, we propose a Local Heads + Global Head setup (Figure 2b) that, in addition to local heads, trains a global head over the embedding with the complete label set. This aligns the different local representations by providing an additional training signal across all hierarchy levels. The global head works on the union of all label sets (e. g., {Children, MiddleGrade, Action&Adventure, Fantasy&Magic}).

Third, we introduce a Local Heads + Nested Head setup (Figure 2c) where we train a head over the full label set that takes as input the outputs of the local heads rather than directly from the embedding. This allows for higher-level integration of the predictions from individual hierarchical levels.

3.3 Training Objective

The foundation of our approach is a deep neural network encoder $F(\cdot;\theta_F) : \mathcal{X} \to \mathbb{R}^d$ that maps input data to a d-dimensional embedding space. For any input $\boldsymbol{x}$, the encoder produces an embedding $\boldsymbol{z} = F(\boldsymbol{x};\theta_F)$. We consider a dataset $\mathcal{D} = \{(\boldsymbol{x}_i, Y_{i,1}, \ldots, Y_{i,k}) \mid i \in [n]\}$ of size n, where each input $\boldsymbol{x}_i \in \mathcal{X}$ has labels $Y_{i,j} \subseteq L_j \subseteq L$ across k hierarchical levels. Here, L represents the complete label space, L_j is the subset of labels for the j-th hierarchical level, and $Y_{i,j}$ are the specific labels assigned to input $\boldsymbol{x}_i$ at level j.

For the **(a) Local Heads Only** setup, each hierarchical level is treated as a distinct classification task with its own label space. Each hierarchy level has its own classifier head MLP_j implemented as a two-layer MLP. This MLP takes the shared embedding as input and maps it to the predictions for the corresponding label set L_j.

The training objective minimizes a weighted sum of losses across all hierarchical levels:

$$\min_{\{\theta_j\}_{j\in[k]},\theta_F} \frac{1}{n}\sum_{i\in[n]} \mathrm{Local}_i\,,$$

$$\mathrm{Local}_i = \sum_{j\in[k]} c_j \cdot \mathcal{L}_j(\mathrm{MLP}_j(\boldsymbol{z}_i); Y_{i,j})$$

where θ_j represents the parameters for the j^{th} level classifier, and the scalar $c_j \geq 0$ represents the level weight.

For the **(b) Local Heads + Global Head** setup, we add an additional term to the objective function:

$$\min_{\{\theta_j\}_{j\in[k]},\theta_g,\theta_F} \frac{1}{n}\sum_{i\in[n]} (\mathrm{Local}_i + \mathrm{Global}_i)\,,$$

$$\mathrm{Global}_i = c_g \cdot \mathcal{L}_g(\mathrm{MLP}_g(\boldsymbol{z}_i); Y_i)$$

where MLP_g represents the global classification head with parameters θ_g, $\mathcal{L}_g$ is the loss function of the global classification task, $Y_i = \bigcup_{j\in[k]} Y_{i,j}$ is the complete set of labels for input $\boldsymbol{x}_i$ across all

Dataset	Depth	#C	Labels/doc.	#Train	#Val	#Test
NYT	8	166	$7.59_{5.61}$	23,345	5,834	7,292
RCV1-V2	4	103	$3.24_{1.40}$	20,834	2,315	781,265
BGC	4	146	$3.01_{1.41}$	58,715	14,785	18,394
WOS-46985	2	141	$2.00_{0.00}$	30,070	7,518	9,397

Table 2: Characteristics of the HTC datasets. #C refers to the number of classes. We also report the average number of labels per document with standard deviation.

hierarchies, and $c_g \geq 0$ is the importance weight for the global task.

For the **(c) Local Heads + Nested Head** setup, the objective function becomes:

$$\min_{\{\theta_j\}_{j\in[k]},\theta_n,\theta_F} \frac{1}{n} \sum_{i\in[n]} (\text{Local}_i + \text{Nested}_i) \,,$$

$$\text{Nested}_i = c_n \cdot \mathcal{L}_n(\text{MLP}_n([o_{i,1}, \ldots, o_{i,k}]); Y_i) \,.$$

Here $o_{i,j} = \text{MLP}_j(\boldsymbol{z}_i)$ represents the output of the j^{th} local head for input $\boldsymbol{x}_i$, MLP_n is the nested head with parameters θ_n that takes as input the concatenated outputs $o_{i,j}$ of all local heads, and $c_n \geq 0$ is the importance weight for the nested classification task.

4 Experimental Apparatus

4.1 Datasets

For hierarchical text classification (HTC), we employ four widely used benchmark datasets as presented in Table 2. The New York Times Annotated Corpus (NYT) (Sandhaus, 2008) features news articles with an 8-level hierarchy. RCV1-V2 (Lewis et al., 2004) consists of Reuters news articles with a 4-level hierarchy of topic categories. The Blurb Genre Collection (BGC) (Aly et al., 2019) consists of English book blurbs annotated with multi-label genres organized in a 4-level hierarchy. WOS-46985 (WOS) (Kowsari et al., 2017) contains scientific abstracts from the Web of Science with a 2-level hierarchy of research areas and sub-areas.

4.2 Procedure

We train and evaluate a wide range of different flat classifier models. These include BERT-base (Devlin et al., 2019), BERT-large (Devlin et al., 2019), RoBERTa-base (Liu et al., 2019), DeBERTa-base (He et al., 2021), DeBERTaV3-base (He et al., 2023), ModernBERT-base (Warner et al., 2024), and unLLama-7B (Li et al., 2023). Each model is trained as a flat classifier over the entire label set to establish strong baselines.

For HYDRA, we select RoBERTa-base as the encoder, as it demonstrated the strongest performance among the flat models in preliminary experiments. We use the same encoder in all our three setups: (a) Local Heads Only, (b) Local Heads + Global Head, and (c) Local Heads + Nested Head. In addition, for setups with a global head, we separately evaluate a variant using only the local heads versus using only the global head.

All experiments are run five times with fixed seeds to ensure reproducibility, and we report the average performance along with the standard deviation. Due to computational constraints and long training times, unLLama was only trained once per dataset. All experiments were performed on a single NVIDIA H100 GPU with 80GB of memory.

4.3 Hyperparameter Optimization

We conduct a hyperparameter optimization using grid search for the flat baseline models, exploring learning rates of $\{2 \cdot 10^{-5}, 3.5 \cdot 10^{-5}, 5 \cdot 10^{-5}, 8 \cdot 10^{-5}\}$ and the threshold λ of $\{0.3, 0.5\}$ as decision boundary which labels to include as HTC is a multi-label classification task. Based on validation performance, we select a learning rate of $3.5 \cdot 10^{-5}$ and $\lambda = 0.5$ for all models except ModernBERT, which performed best with a learning rate of $8 \cdot 10^{-5}$. Models are trained for 50 epochs, with early stopping to prevent overfitting. For unLLaMA, we adopt the hyperparameters reported by Li et al. (2023), specifically a learning rate of $8 \cdot 10^{-5}$, $\lambda = 0.5$, LoRA rank of $r = 12$, and max pooling. For HYDRA, we use the same hyperparameters as for RoBERTa-base and assign each loss component equal importance, i. e., $c_j = 1$ for $j \in [k]$. We use Binary Cross-Entropy (BCE) as the loss function for all heads.

4.4 Metrics

Following previous work, we report micro- and macro-F1 scores, which are commonly used in multi-label and hierarchical classification settings. The micro F1 score aggregates contributions of all classes to compute the average performance, reflecting overall classification effectiveness. In contrast, the macro F1 score computes the metric independently for each class and then averages the results, giving equal weight to all classes.

Model	NYT		RCV1-V2		BGC		WOS		Provenance
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	
Flat Baselines									
BERT-base	$79.07_{0.22}$	$67.63_{0.42}$	$86.38_{0.23}$	$67.89_{1.42}$	$81.44_{0.37}$	$64.60_{0.74}$	$86.19_{0.11}$	$80.23_{0.20}$	Own experiment
BERT-large	$80.69_{0.08}$	$70.27_{0.39}$	$87.40_{0.11}$	$70.15_{0.39}$	$\underline{81.94_{0.35}}$	$65.39_{0.25}$	$86.66_{0.31}$	$80.80_{0.38}$	Own experiment
RoBERTa-base	$\underline{81.47_{0.15}}$	$\underline{71.45_{0.45}}$	$87.30_{0.10}$	$68.60_{1.40}$	$81.52_{0.26}$	$64.13_{0.55}$	$86.35_{0.15}$	$80.16_{0.13}$	Own experiment
DeBERTa-base	$81.21_{0.17}$	$71.33_{0.26}$	$87.24_{0.18}$	$69.66_{0.52}$	$81.28_{0.18}$	$63.85_{0.24}$	$86.82_{0.11}$	$80.85_{0.48}$	Own experiment
DeBERTaV3-base	$79.96_{0.20}$	$67.30_{0.23}$	$86.77_{0.18}$	$67.49_{1.87}$	$81.52_{0.16}$	$63.41_{1.13}$	$86.58_{0.24}$	$80.41_{0.69}$	Own experiment
ModernBERT-base	$79.83_{0.53}$	$68.99_{0.54}$	$86.20_{0.42}$	$67.43_{0.66}$	$81.29_{0.23}$	$63.00_{0.53}$	$86.15_{0.33}$	$80.34_{0.22}$	Own experiment
unLLama-7B (LoRa)	77.20	61.68	84.45	61.42	76.29	45.43	84.13	74.32	Own experiment
Hierarchical Classification Methods (SOTA)									
BERT+HiAGM	78.64	66.76	85.58	67.93	78.62	62.98	86.04	80.19	Wang et al. (2022)[1]
BERT+HiMatch	—	—	86.33	68.66	79.23	62.62	86.70	81.06	Liu et al. (2025)
HGCLR	78.86	67.96	86.49	68.31	79.36	63.64	87.11	81.20	Wang et al. (2022)[1]
HILL	80.47	69.96	87.31	70.12	—	—	87.28	81.77	Zhu et al. (2024)
HBGL	80.47	70.19	87.23	$\underline{71.07}$	—	—	87.36	82.00	Jiang et al. (2022)
HBGL+RoBERTa	79.89	70.57	87.52	70.52	—	—	$\underline{\mathbf{87.66}}$	81.96	Younes et al. (2024)
HALB	79.56	69.28	86.94	69.32	—	—	87.45	82.04	Zhang et al. (2024)
DFG	—	—	87.44	70.37	81.17	66.13	87.42	82.27	Liu et al. (2025)
HJCL	$80.52_{0.28}$	$70.02_{0.31}$	$87.04_{0.24}$	$70.49_{0.32}$	$81.30_{0.29}$	$\underline{\mathbf{66.77_{0.37}}}$	—	—	U et al. (2023)
HiSR	80.32	70.11	$\underline{87.59}$	70.72	—	—	87.52	82.04	Zhou et al. (2025)
Seq2Tree	—	—	86.88	70.01	79.72	63.96	87.20	$\underline{\mathbf{82.50}}$	Yu et al. (2022)
RADAr	$79.84_{0.07}$	$68.64_{0.28}$	$87.23_{0.05}$	$69.64_{0.12}$	—	—	$87.17_{0.04}$	$81.84_{0.08}$	Younes et al. (2024)
HYDRA (Ours)									
(a) Local Heads Only									
RoBERTa-base	$81.69_{0.22}$	$72.11_{0.34}$	$87.30_{0.20}$	$69.36_{1.12}$	$81.65_{0.40}$	$64.23_{1.09}$	$86.90_{0.22}$	$81.18_{0.33}$	Own experiment
(b) Local Heads + Global Head									
RoBERTa-base (Local Heads)	$81.91_{0.07}$	$\mathbf{72.74_{0.23}}$	$87.67_{0.09}$	$70.58_{0.44}$	$81.94_{0.53}$	$65.47_{0.89}$	$86.83_{0.16}$	$81.16_{0.30}$	Own experiment
RoBERTa-base (Global Head)	$\mathbf{81.96_{0.06}}$	$72.35_{0.22}$	$\mathbf{87.79_{0.11}}$	$70.08_{0.76}$	$82.14_{0.44}$	$65.40_{0.76}$	$86.91_{0.18}$	$81.22_{0.40}$	Own experiment
(c) Local Heads + Nested Head									
RoBERTa-base (Local Heads)	$81.87_{0.31}$	$72.43_{0.41}$	$87.73_{0.19}$	$\mathbf{71.12_{0.24}}$	$\mathbf{82.18_{0.36}}$	$66.01_{0.66}$	$86.90_{0.12}$	$81.14_{0.24}$	Own experiment
RoBERTa-base (Nested Head)	$81.82_{0.29}$	$72.10_{0.63}$	$87.72_{0.19}$	$70.57_{0.37}$	$82.17_{0.28}$	$66.17_{0.64}$	$86.83_{0.11}$	$81.08_{0.27}$	Own experiment
Performance Gains									
HYDRA vs. RoBERTa-base	+0.49	+1.29	+0.49	+2.52	+0.66	+2.04	+0.56	+1.06	
HYDRA vs. Best Baseline	+0.49	+1.29	+0.20	+0.05	+0.24	-0.60	-0.75	-1.28	

[1] The BGC results for these models are as reported by Liu et al. (2025).

Table 3: Results on four common HTC datasets. For our experiments, we report the mean and standard deviation over five runs. **Bold** entries denote the overall best result and underlined entries denote the best baseline.

5 Results

Our results are presented in Table 3, which shows the effectiveness of HYDRA in various datasets. The benefits of our multi-head architecture are demonstrated by the performance gains, which are especially noticeable compared to flat models and more complex hierarchical methods.

Flat Models Our experiments reveal strong results from flat encoder-only models in most datasets. RoBERTa-base outperforms many specialized hierarchical models on the NYT dataset, achieving the highest scores with a micro-F1 of 81.47 and macro-F1 of 71.45 among the baseline models. Similarly, BERT-large delivers the best baseline micro-F1 performance of 81.94 on the BGC dataset. This indicates that even without explicit architectural elements for hierarchy modeling, pre-trained language models already provide good results. However, their performance on the WOS dataset is less competitive. In particular, unLLama-7B consistently performs worse than smaller models on all datasets, even though it is the largest model evaluated.

HYDRA with Only Local Heads The simplest HYDRA setup using only level-specific classification heads has already demonstrated consistent improvements over the flat model baselines. The gains over the strong flat RoBERTa-base are particularly notable for macro-F1 scores, with improvements of +0.66 on NYT, +0.76 on RCV1-V2, +0.10 on BGC, and +1.02 on WOS.

HYDRA with Local and Global Heads Adding a global head further enhances the performance across all datasets. When evaluated on the local heads, the Global Head setup achieves micro/macro-F1 improvements of +0.22/+0.63 on NYT, +0.37/+1.22 on RCV1-V2, +0.29/+1.24 on BGC, and -0.07/-0.02 on WOS compared to the Local Heads Only setup. When using the global head for inference, the performance is comparable to using only the local heads. The Nested Head setup further improves results on RCV1-V2 and BGC while maintaining comparable performance on NYT and WOS.

Summary HYDRA consistently improves on the base RoBERTa model across all datasets, with

micro/macro-F1 gains of +0.49/+1.29 on NYT, +0.49/+2.52 on RCV1-V2, +0.66/+2.04 on BGC, and +0.56/+1.06 on WOS. Compared to the best baseline/SOTA models, HYDRA achieves improvements of +0.49 micro-F1 and +1.29 macro-F1 on NYT and small improvements of +0.20 micro-F1 and +0.05 macro-F1 on RCV1-V2 and +0.24 micro-F1 on BGC. HYDRA performs below the best baseline on WOS by -0.75 micro-F1 and -1.28 macro-F1 and BGC macro-F1 by -0.60.

Our results demonstrate that our simple multi-head approach HYDRA is competitive with much more complex hierarchical classification methods.

Model	WOS	
	Micro-F1	Macro-F1
Local Heads Only		
RoBERTa-base	$86.37_{0.18}$	$85.79_{0.15}$
Local Heads + Global Head		
RoBERTa-base (Local Heads)	$\mathbf{87.11_{0.07}}$	$\mathbf{86.53_{0.10}}$
RoBERTa-base (Global Head)	$84.35_{0.18}$	$77.72_{0.39}$
Local Heads + Nested Head		
RoBERTa-base (Local Heads)	$86.90_{0.20}$	$86.28_{0.18}$
RoBERTa-base (Nested Head)	$85.35_{0.47}$	$79.13_{0.48}$
vs. Multi-label HYDRA	+0.20	+5.31
vs. Best Baseline	-0.55	+4.03

Table 4: Performance of HYDRA on the WOS dataset using single-label classifiers. By adapting the classification heads HYDRA achieves far better Macro-F1 scores. **Bold** entries denote the best result.

6 Discussion

The experimental results demonstrate that HYDRA provides a simple but effective approach to hierarchical text classification. Our findings challenge the idea that sophisticated architectural elements are required to achieve state-of-the-art performance on hierarchical classification tasks. In this section, we discuss our results, present ablation studies, and suggest directions for future work.

6.1 Key Scientific Insights

Flat Models Our experiments reveal that flat encoder-only models are surprisingly strong baselines for hierarchical text classification, often outperforming specialized hierarchical methods. This strength may be partially explained by the rigorous hyperparameter optimization performed versus relying on numbers reported in the literature. Existing HTC methods compare themselves only against a BERT-base model, rather than using stronger encoder-only models. The performance of models like RoBERTa and BERT-large suggests that pre-trained language models are already strong HTC classifiers. The poor performance of unLLama-7B, despite its much larger parameter count, is particularly interesting. This may be attributed to several factors, including differences in pre-training objectives, the challenges of fine-tuning large models with limited data, and the potential mismatch between the generative capabilities of LLMs and the discriminative nature of classification tasks.

HYDRA with Only Local Heads The consistent improvements achieved by HYDRA with only local heads confirm our hypothesis. Treating hierarchical levels as distinct but related tasks can enhance performance without additional architectural complexity. The strong improvements in macro-F1 scores suggest that HYDRA's multi-task formulation is especially beneficial for less frequent classes, which often correspond to more specific labels in lower hierarchy levels. By providing explicit supervision at each level, HYDRA learns more robust representations for these challenging instances.

HYDRA with Local and Global Heads Adding a global head to HYDRA further enhances performance across most datasets. This improvement shows that the global head encourages the shared representation to capture beneficial information at all hierarchical levels by regularizing it. The level-specific signals from the local heads are effectively aided by an extra training signal from the global head. The nested head setup demonstrates that the use of the output of local heads for global prediction can be effective for certain datasets (i. e., RCV1-V2 and BGC). While training with additional heads introduces a small computational overhead, the cost of these lightweight MLPs is minimal compared to the performance gains. For details on model size and runtime, see Appendix A. Additionally, for inference, it is possible to use either only the local heads or only the global head, which reduces HYDRA to a flat classifier. Using only the global head still outperforms all flat baselines. Thus, HYDRA benefits from the better representations learned during training and yields better inference efficiency.

HYDRA's Performance on WOS While HYDRA improves the flat RoBERTa model on WOS it is not as strong as other baseline models. This performance difference comes from a unique characteristic of the WOS dataset, which contains ex-

Model	NYT		RCV1-V2		BGC		WOS	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
HYDRA w/ Nested Head	$81.87_{0.31}$	$72.43_{0.41}$	$\mathbf{87.73_{0.19}}$	$\mathbf{71.12_{0.24}}$	$\mathbf{82.18_{0.36}}$	$\mathbf{66.01_{0.66}}$	$\mathbf{86.90_{0.12}}$	$81.14_{0.24}$
HYDRA w/ Global Head	$\mathbf{81.91_{0.07}}$	$\mathbf{72.74_{0.23}}$	$87.67_{0.09}$	$70.58_{0.44}$	$81.94_{0.53}$	$65.47_{0.89}$	$86.83_{0.16}$	$81.16_{0.30}$
HYDRA w/ local heads only	$81.69_{0.22}$	$72.11_{0.34}$	$87.30_{0.20}$	$69.36_{1.12}$	$81.65_{0.40}$	$64.23_{1.09}$	$\mathbf{86.90_{0.22}}$	$\mathbf{81.18_{0.33}}$
w/o shared encoder	$79.87_{0.87}$	$69.08_{1.77}$	$86.47_{0.70}$	$67.50_{1.10}$	$74.84_{5.75}$	$41.90_{18.01}$	$86.11_{0.35}$	$80.46_{0.53}$
w/o embedding projection	$81.62_{0.04}$	$72.05_{0.12}$	$87.29_{0.22}$	$69.21_{1.09}$	$81.19_{0.35}$	$64.00_{0.71}$	$86.58_{0.36}$	$80.85_{0.50}$
w/o multiple heads (RoBERTa-base)	$81.47_{0.15}$	$71.45_{0.45}$	$87.30_{0.10}$	$68.60_{1.40}$	$81.52_{0.26}$	$64.13_{0.55}$	$86.35_{0.15}$	$80.16_{0.13}$

Table 5: Ablation study of HYDRA's components. Every component contributes to HYDRA's performance, and incorporating a global head during training yields further improvements. **Bold** entries denote the best result.

actly one label per hierarchy level, i. e., two labels per document. This makes multi-label classifiers suboptimal for this dataset.

We conducted additional experiments to further investigate this observation. We modified HYDRA by replacing its local multi-label classifier with a single-label classifier. This adaptation ensures strict single-label predictions at each hierarchy level and highlights the modularity of HYDRA. Its classification heads can be trivially swapped to match dataset-specific requirements. As shown in Table 4, this modification improves Macro-F1 from 81.22 to 86.53 (+5.31), surpassing Seq2Tree, the best state-of-the-art HTC model by +4.03 Macro-F1. Interestingly, in this experiment, a global head improves the local heads but does not yield good results on its own. This is due to the fact that the global head is still a flat multi-label classifier.

6.2 Ablation study

Our ablation study of HYDRA's components is presented in Table 5. The shared encoder is the most critical component, with performance dropping drastically when using separate encoders for each hierarchy level (e. g., -1.82 micro-F1 and -3.03 macro-F1 on NYT). This demonstrates the importance of parameter sharing, which allows the model to learn common representations that benefit all hierarchy levels. The embedding projection layer also contributes to performance, though to a lesser extent. Removing this component results in a moderate performance decrease across all datasets (e. g., -0.46 micro-F1 and -0.23 macro-F1 on BGC), suggesting that additional shared parameters after the embedding help to capture level-specific features. Among the different HYDRA variants, adding a global or nested head generally provides further performance improvements. The nested head setup proves especially effective for RCV1-V2 and BGC, while the global head setup performs slightly better on NYT. These performance gains come with minimal additional computational cost during training, but for inference, the model can rely solely on local heads.

6.3 Future Work and Impact

Future research on HYDRA may investigate adaptive weighting schemes for hierarchical levels during training, potentially improving performance on imbalanced hierarchies. Rather than assigning equal importance to each level, weights could be adjusted based on validation performance or uncertainty estimates, optimizing the balance between level-specific and overall classification accuracy. These applications would test the scalability of our approach to very deep and broad hierarchies not covered by current benchmarks. Another promising direction is to explore the combination of local and global heads for inference. This could be done by applying ensemble learning techniques, such as averaging the logits of both heads, to potentially further improve classification performance.

By demonstrating that simple architectures can achieve state-of-the-art results, this work encourages a shift toward less sophisticated and more interpretable approaches in hierarchical text classification. Architectural simplicity makes hierarchical classification more accessible to researchers and practitioners who do not have expertise in complex architectures.

7 Conclusion

We introduced HYDRA, a simple yet effective multi-head encoder-only architecture for hierarchical text classification that treats each level in the hierarchy as a separate classification task. Our experiments on four standard benchmarks demonstrate that this approach matches or exceeds the performance of more complex state-of-the-art methods without requiring graph encoders, label semantics, or autoregressive decoders. By showing that hierarchical awareness can be effectively incorporated

into flat encoder-only models through a multi-task formulation, our work opens new possibilities for accessible hierarchical classification.

Limitations

While HYDRA offers a simpler and more efficient approach to hierarchical text classification, several limitations must be recognized. Our method assumes a fixed pre-defined hierarchy, making it less suitable for dynamic taxonomies that evolve over time without retraining. Although HYDRA matches or exceeds state-of-the-art performance on standard benchmarks, there may be domains or extremely complex hierarchies where more sophisticated architectural components provide benefits that our multi-head approach cannot capture. Furthermore, the current implementation treats each hierarchical level independently during inference, potentially leading to inconsistent predictions across levels. However, this also applies to most other models and can be solved with post-processing using the hierarchy. Finally, while we demonstrate HYDRA's effectiveness across multiple benchmarks, its performance on extremely large-scale hierarchies with thousands of classes remains to be explored in future work.

Ethical Considerations

While text classification has a broad spectrum of application domains ranging from recommender systems and information retrieval to tasks such as sentiment analysis, e-commerce, advertising, and news classification (Zangari et al., 2024), we believe that no specific societal consequences require immediate emphasis in the context of this work.

Acknowledgments

The authors acknowledge support by the state of Baden-Württemberg through bwHPC. This research is co-funded by the SmartER project (No. 515537520) of the DFG, German Research Foundation.

References

Rami Aly, Steffen Remus, and Chris Biemann. 2019. Hierarchical multi-label classification of text with capsule networks. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28 - August 2, 2019, Volume 2: Student Research Workshop*, pages 323–330. Association for Computational Linguistics.

Siddhartha Banerjee, Cem Akkaya, Francisco Perez-Sorrosal, and Kostas Tsioutsiouliklis. 2019. Hierarchical transfer learning for multi-label text classification. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 6295–6300. Association for Computational Linguistics.

Rich Caruana. 1997. Multitask learning. *Mach. Learn.*, 28(1):41–75.

Haibin Chen, Qianli Ma, Zhenxi Lin, and Jiangyue Yan. 2021. Hierarchy-aware label semantics matching network for hierarchical text classification. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4370–4379. Association for Computational Linguistics.

Shijie Chen, Yu Zhang, and Qiang Yang. 2024. Multi-task learning in natural language processing: An overview. *ACM Comput. Surv.*, 56(12):295:1–295:32.

Michael Crawshaw. 2020. Multi-task learning with deep neural networks: A survey. *CoRR*, abs/2009.09796.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. DeBERTaV3: Improving DeBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DeBERTa: decoding-enhanced bert with disentangled attention. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Wei Huang, Chen Liu, Bo Xiao, Yihua Zhao, Zhaoming Pan, Zhimin Zhang, Xinyun Yang, and Guiquan Liu. 2022. Exploring label hierarchy in a generative way for hierarchical text classification. In *Proceedings of*

the 29th International Conference on Computational Linguistics, COLING 2022, Gyeongju, Republic of Korea, October 12-17, 2022, pages 1116–1127. International Committee on Computational Linguistics.

Ting Jiang, Deqing Wang, Leilei Sun, Zhongzhi Chen, Fuzhen Zhuang, and Qinghong Yang. 2022. Exploiting global and local hierarchies for hierarchical text classification. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 4030–4039. Association for Computational Linguistics.

Kamran Kowsari, Donald E. Brown, Mojtaba Heidarysafa, Kiana Jafari Meimandi, Matthew S. Gerber, and Laura E. Barnes. 2017. HDLTex: Hierarchical deep learning for text classification. In *16th IEEE International Conference on Machine Learning and Applications, ICMLA 2017, Cancun, Mexico, December 18-21, 2017*, pages 364–371. IEEE.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. RCV1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5:361–397.

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

Renyuan Liu, Xuejie Zhang, Jin Wang, and Xiaobing Zhou. 2025. Disentangled feature graph for hierarchical text classification. *Information Processing & Management*, 62(3):104065.

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

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

Evan Sandhaus. 2008. The New York Times Annotated Corpus. Linguistic Data Consortium, Philadelphia, 6(12):e26752.

Kazuya Shimura, Jiyi Li, and Fumiyo Fukumoto. 2018. HFT-CNN: learning hierarchical category structure for multi-label short text categorization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 811–816. Association for Computational Linguistics.

Robert Endre Tarjan. 1972. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160.

Simon Chi Lok U, Jie He, Víctor Gutiérrez-Basulto, and Jeff Z. Pan. 2023. Instances and labels: Hierarchy-aware joint supervised contrastive learning for hierarchical multi-label text classification. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 8858–8875. Association for Computational Linguistics.

Zihan Wang, Peiyi Wang, Lianzhe Huang, Xin Sun, and Houfeng Wang. 2022. Incorporating hierarchy into text encoder: a contrastive learning approach for hierarchical text classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 7109–7119. Association for Computational Linguistics.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, and 1 others. 2024. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. *arXiv preprint arXiv:2412.13663*.

Jonatas Wehrmann, Ricardo Cerri, and Rodrigo C. Barros. 2018. Hierarchical multi-label classification networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 5225–5234. PMLR.

Yousef Younes, Lukas Galke, and Ansgar Scherp. 2024. RADAr: A transformer-based autoregressive decoder architecture for hierarchical text classification. In *ECAI 2024 - 27th European Conference on Artificial Intelligence, 19-24 October 2024, Santiago de Compostela, Spain - Including 13th Conference on Prestigious Applications of Intelligent Systems (PAIS 2024)*, volume 392 of *Frontiers in Artificial Intelligence and Applications*, pages 1559–1566. IOS Press.

Chao Yu, Yi Shen, and Yue Mao. 2022. Constrained sequence-to-tree generation for hierarchical text classification. In *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, pages 1865–1869. ACM.

Alessandro Zangari, Matteo Marcuzzo, Matteo Rizzo, Lorenzo Giudice, Andrea Albarelli, and Andrea Gasparetto. 2024. Hierarchical text classification and its foundations: A review of current research. *Electronics*, 13(7):1199.

Chaoqun Zhang, Linlin Dai, Chengxing Liu, and Longhao Zhang. 2025. HGBL: A fine granular hierarchical multi-label text classification model. *Neural Process. Lett.*, 57(1):1.

Jun Zhang, Yubin Li, Fanfan Shen, Chenxi Xia, Hai Tan, and Yanxiang He. 2024. Hierarchy-aware and label balanced model for hierarchical text classification. *Knowl. Based Syst.*, 300:112153.

Zhihan Zhang, Wenhao Yu, Mengxia Yu, Zhichun Guo, and Meng Jiang. 2023. A survey of multi-task learning in natural language processing: Regarding task relatedness and training methods. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2023, Dubrovnik, Croatia, May 2-6, 2023*, pages 943–956. Association for Computational Linguistics.

Jie Zhou, Chunping Ma, Dingkun Long, Guangwei Xu, Ning Ding, Haoyu Zhang, Pengjun Xie, and Gongshen Liu. 2020. Hierarchy-aware global model for hierarchical text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 1106–1117. Association for Computational Linguistics.

Juncheng Zhou, Lijuan Zhang, Yachen He, Rongli Fan, Lei Zhang, and Jian Wan. 2025. A novel negative sample generation method for contrastive learning in hierarchical text classification. In *Proceedings of the 31st International Conference on Computational Linguistics, COLING 2025, Abu Dhabi, UAE, January 19-24, 2025*, pages 5645–5655. Association for Computational Linguistics.

He Zhu, Junran Wu, Ruomei Liu, Yue Hou, Ze Yuan, Shangzhe Li, Yicheng Pan, and Ke Xu. 2024. HILL: hierarchy-aware information lossless contrastive learning for hierarchical text classification. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 4731–4745. Association for Computational Linguistics.

Supplementary Material

A Efficiency of HYDRA

This section discusses the number of parameters for both HYDRA and compared HTC models, as well as the training time per epoch for HYDRA across all datasets. Table 6 lists the parameter counts of the HTC models included in our comparison. If the number of parameters is available in the literature, we report those values. In cases where this information is not available, we measure the parameter counts by running the respective model source code on the NYT dataset. For models lacking both published parameter counts and accessible code, we indicate that it is unavailable. Table 7 shows the parameter counts for the flat RoBERTa model and the three HYDRA setups. The parameter count varies based on dataset characteristics, particularly the number of hierarchical levels and classes per level. Even in the most complex case (NYT with 8 hierarchical levels), HYDRA with Local Heads + Global Head adds only about 10 M parameters over the flat baseline. The Nested Head setup consistently requires fewer parameters than the Global Head setup since the number of classes in our datasets is always less than the embedding size, making the nested head smaller than the global head.

Model	Parameters	Provenance
RoBERTa	126M	Measured
BERT + HiAGM	143M	(Huang et al., 2022)
BERT + HiMatch	153M	(Huang et al., 2022)
HGCLR	129M	(Zhu et al., 2024)
HILL	117M	(Zhu et al., 2024)
HBGL	110M	Measured
HALB	120M	Measured
HGBL	—	Unavailable
DFG	131M	Measured
HJCL	241M	Measured
HiSR	116M	Measured
Seq2Tree	—	Unavailable

Table 6: Parameter counts for models compared in this work. "Measured" indicates that the values are obtained by running the model source code on the NYT dataset.

Table 8 compares the training time per epoch across all models and datasets. The additional computational cost of HYDRA is negligible, with an increase of only 1-6 seconds per epoch compared to the flat baseline. This minimal overhead comes from the lightweight nature of the classification heads, which are simple two-layer MLPs that require only minimal computational resources. These efficiency results demonstrate that HYDRA achieves superior hierarchical classification performance with minimal additional computational cost.

B DeBERTa as an Alternative Encoder

To test the generalizability of HYDRA's architecture across different pre-trained language models, we conducted additional experiments using DeBERTa-base as the encoder. Table 9 presents the performance of HYDRA using DeBERTa-base compared to RoBERTa-base. The results demonstrate that HYDRA's multi-head approach effectively improves hierarchical text classification regardless of whether RoBERTa or DeBERTa is used as the encoder. Consistent with our main findings, all HYDRA setups outperform the respective flat baseline models. This further supports our finding that treating hierarchical levels as distinct but related tasks enhances performance without requiring complex architectural components. Notably, while RoBERTa-base generally outperforms DeBERTa-base, DeBERTa shows competitive performance.

Model	NYT	RCV1-V2	BGC	WOS
Flat RoBERTa	126M	126M	126M	126M
HYDRA w/ Local Heads Only	134M	130M	130M	127M
HYDRA w/ Local Heads + Global Head	136M	131M	131M	129M
HYDRA w/ Local Heads + Nested Head	135M	130M	130M	128M

Table 7: Parameter counts in millions for RoBERTa-base and HYDRA across all datasets, demonstrating the minimal parameter overhead introduced by HYDRA.

Model	NYT	RCV1-V2	BGC	WOS
Flat RoBERTa	$324.35_{4.42}$	$285.94_{1.96}$	$804.11_{1.42}$	$413.61_{0.56}$
HYDRA w/ Local Heads Only	$330.60_{1.36}$	$288.86_{2.24}$	$809.52_{1.01}$	$414.77_{0.65}$
HYDRA w/ Local Heads + Global Head	$330.39_{0.97}$	$288.59_{2.70}$	$810.36_{1.07}$	$416.04_{1.00}$
HYDRA w/ Local Heads + Nested Head	$330.07_{0.97}$	$288.78_{1.73}$	$809.74_{0.91}$	$415.91_{0.80}$

Table 8: Training time per epoch in seconds for RoBERTa-base and HYDRA across 5 runs, reporting mean and standard deviation. HYDRA shows only marginal computational overhead compared to a flat RoBERTa.

Model	NYT		RCV1-V2		BGC		WOS	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
Local Heads Only								
RoBERTa-base	$81.69_{0.22}$	$72.11_{0.34}$	$87.30_{0.20}$	$69.36_{1.12}$	$81.65_{0.40}$	$64.23_{1.09}$	$86.90_{0.22}$	$81.18_{0.33}$
DeBERTa-base	$81.28_{0.24}$	$71.53_{0.37}$	$87.25_{0.12}$	$69.82_{0.25}$	$80.91_{0.27}$	$63.37_{0.51}$	$86.82_{0.30}$	$80.88_{0.11}$
Local Heads + Global Head								
RoBERTa-base (Local Heads)	$81.91_{0.07}$	$\mathbf{72.74_{0.23}}$	$87.67_{0.09}$	$70.58_{0.44}$	$81.94_{0.53}$	$65.47_{0.89}$	$86.83_{0.16}$	$81.16_{0.30}$
RoBERTa-base (Global Head)	$\mathbf{81.96_{0.06}}$	$72.35_{0.22}$	$\mathbf{87.79_{0.11}}$	$70.08_{0.76}$	$82.14_{0.44}$	$65.40_{0.76}$	$\mathbf{86.91_{0.18}}$	$81.22_{0.40}$
DeBERTa-base (Local Heads)	$81.26_{0.27}$	$71.57_{0.41}$	$87.07_{0.25}$	$69.33_{0.59}$	$81.13_{0.25}$	$63.89_{0.52}$	$86.69_{0.19}$	$81.09_{0.22}$
DeBERTa-base (Global Head)	$81.45_{0.23}$	$71.34_{0.44}$	$87.23_{0.21}$	$68.18_{2.01}$	$81.26_{0.23}$	$63.53_{0.72}$	$86.75_{0.15}$	$81.17_{0.17}$
Local Heads + Nested Head								
RoBERTa-base (Local Heads)	$81.87_{0.31}$	$72.43_{0.41}$	$87.73_{0.19}$	$\mathbf{71.12_{0.24}}$	$\mathbf{82.18_{0.36}}$	$66.01_{0.66}$	$86.90_{0.12}$	$81.14_{0.24}$
RoBERTa-base (Nested Head)	$81.82_{0.29}$	$72.10_{0.63}$	$87.72_{0.19}$	$70.57_{0.37}$	$82.17_{0.28}$	$\mathbf{66.17_{0.64}}$	$86.83_{0.11}$	$81.08_{0.27}$
DeBERTa-base (Local Heads)	$81.58_{0.24}$	$72.07_{0.28}$	$87.18_{0.13}$	$69.60_{0.47}$	$81.26_{0.18}$	$64.13_{0.53}$	$86.74_{0.12}$	$\mathbf{81.35_{0.13}}$
DeBERTa-base (Nested Head)	$81.53_{0.25}$	$71.64_{0.28}$	$87.15_{0.14}$	$68.85_{0.76}$	$81.20_{0.20}$	$63.73_{0.37}$	$86.67_{0.14}$	$81.21_{0.16}$

Table 9: Comparison of RoBERTa-base and DeBERTa-base as encoders for HYDRA. All scores are presented as mean and standard deviation over five runs. **Bold** entries indicate the best performance for each metric across all setups.