XAutoLLM: Efficient Fine-Tuning of Language Models via Meta-Learning and AutoML

Anonymous ACL submission

Abstract

Experts in machine learning distinguish themselves from amateurs by leveraging domain knowledge to effectively navigate the myriad decisions involved in model selection, hyperparameter optimisation, and resource allocation. This distinction is especially critical for *Language Models* (LMs), whose repeated fine-tuning trials incur substantial computational overhead and raise environmental concerns. Yet, no single AutoML framework simultaneously addresses both model selection and hyperparameter optimisation for resource-efficient LM fine-tuning.

We propose **XAutoLLM**, a novel AutoML framework that integrates *meta-learning* to warm start the search space. By drawing on task- and system-level meta-features, XAutoLLM reuses insights from previously tuned LMs on related tasks. Through extensive experimentation on four text classification datasets, our framework discovers solutions in up to half the search time, reduces search errors by as much as sevenfold, and produces over 40% more pipelines that achieve improved performance–time trade-offs than a robust baseline. By systematically retaining and learning from prior experiments, XAutoLLM enables resource-friendly, *Green AI* fine-tuning, thereby fostering sustainable NLP pipelines that balance state-of-the-art outcomes with minimised computational overhead.

1 Introduction

Experts in Machine Learning (ML) distinguish themselves not merely through superior tools or abundant data but by utilising *domain knowledge* to focus on promising model and hyperparameter configurations. Rather than mindlessly testing all combinations, they use their experience to identify a small subset of promising options, ensuring their approach is strategic and informed.

Automated Machine Learning (AutoML) seeks to replicate part of this expert-driven process by automating model selection, pipeline construction, and Hyperparameter Optimisation (HPO) (Hutter et al., 2019). However, AutoML systems often lack the rich background knowledge required to effectively prune less promising solutions. Meta-learning addresses this gap by learning from past AutoML runs, providing insights that guide future executions (Vanschoren, 2019). This approach avoids exhaustive searches and shortens optimisation cycles.

Although AutoML techniques have matured in areas such as tabular and vision tasks (Hutter et al., 2019), applying them to language modelling remains underdeveloped (Tornede et al., 2023). LMs are extremely resource-intensive to train and fine-tune (Wang et al., 2023b), even for smaller architectures like BERT (Devlin et al., 2018) or T5 (Raffel et al., 2020). Repeated model evaluations can thus be prohibitively costly, motivating **Green AI** efforts to curb the environmental footprint of large-scale NLP. Yet, most existing AutoML systems for LMs focus on HPO in isolation (Mallik et al., 2024), overlooking the need for a holistic solution that jointly optimises both model selection and HPO, especially under tight resource constraints.

We present XAutoLLM, a meta-learning-enhanced framework that biases the AutoML search space via **warm-start strategies** to address these shortcomings. In particular, we integrate expert heuristics and prior experiences (encoded as meta-features of tasks and hardware configurations) into the initial sampling distribution. By jumpstarting the search with knowledge about plausible or implausible configurations, XAutoLLM systematically filters out resource-heavy, low-yield pipelines. This open-source tool aims to unify model selection *and* HPO for LLMs, enabling practitioners to fine-tune models more efficiently and sustainably. The proposed framework is an open-source tool available for the language modelling research community (Appendix A).

We summarise the main contributions of our paper as follows:

- We propose **XAutoLLM**, the first AutoML framework that simultaneously tackles model selection and hyperparameter optimisation for LM fine-tuning.
- We introduce an extensible *meta-learning mechanism* that leverages task and system meta-features to guide the search, balancing performance and efficiency constraints.
- We demonstrate the effectiveness of our approach through extensive experimentation on four text classification tasks, discovering solutions in up to half the time of a robust baseline, reducing error rates by as much as sevenfold, and achieving competitive or superior performance.

2 Related Work

AutoML strategies in language modelling can be divided into two (not necessarily disjoint) subsets: AutoML for LLMs and LLMs for AutoML (Tornede et al., 2023). The former comprises AutoML techniques to produce optimal LM pipelines tailored for specific scenarios, akin to traditional AutoML. The latter employs language models to enhance the AutoML process, for example, by providing linguistic interfaces to configure the optimisation process or leveraging them to guide the search (e.g., using LMs to generate code for optimal ML pipelines).

AutoML for LLMs in particular poses significant challenges (Tornede et al., 2023). Namely, LMs are extremely resource-intensive (Bannour et al., 2021), even when only considering their later stages (e.g., fine-tuning, inference). Table 1 compares AutoML approaches that leverage LLMs according to relevant features characterising their responses to the field's challenges.

We observe that there are more **LLMs for AutoML** systems than vice versa, likely due to the proliferation of prompt engineering and increased access to open-source language models. For instance, Zhou et al. (2022) developed the Automatic Prompt Engineer (APE) system, which achieved performance competitive with human-generated instructions. In contrast, systems such as GL-Agent (Wei et al., 2023), AutoM3L (Luo et al., 2024) and GizaML (Sayed et al., 2024) integrate language models into their optimisation strategies to produce graph learning pipelines, highly capable multimodal ML pipelines, and time-series forecasting pipelines, respectively.

Systems \ Features	AutoML for LLMs	LLMs for AutoML	Inference	Fine-tuning	HPO	Model Selection	Meta-learning
APE		✓	✓				
GPT-NAS	✓	✓			✓	✓	
GL-Agent		✓					
AutoGen	✓	✓	✓				
EcoOptiGen	✓		✓		✓		
AutoML-GPT	✓	✓			≈		
HuggingGPT	≈	✓	✓			✓	
AutoM3L		✓			✓	✓	≈
PriorBand	✓			✓	✓		✓
GizaML		✓			✓	✓	✓
GE	✓	✓	✓		✓		≈
AutoGOAL	✓		✓		✓	✓	
Introduced in this paper							
XAutoLLM	✓		✓	✓	✓	✓	✓

Table 1: Comparison of systems for AutoML with LLMs

Systems like AutoGen (Wu et al., 2023), GPT-NAS (Yu et al., 2024), GE (Morris et al., 2024), AutoML-GPT (Zhang et al., 2023), and HuggingGPT (Shen et al., 2024) are hybrids that span both categories; they leverage LMs to produce LM-based solutions. However, the last two differ from traditional AutoML (and NAS) systems: AutoML-GPT does not evaluate solution candidates (only simulates their training), and HuggingGPT produces responses to prompts without outputting the pipelines capable of handling them.

Often, the choice of model is as, if not more, critical than the hyperparameter configuration used to produce responses. We found that AutoGOAL (Estevanell-Valladares et al., 2024) optimises pipelines by balancing efficiency and performance metrics, taking into account both model selection and HPO, but only supports LMs for inference. All other AutoML for LLMs systems we surveyed, such as EcoOptiGen (Wang et al., 2023a) and PriorBand (Mallik et al., 2024), focus solely on HPO.

Nonetheless, we find no single framework that simultaneously addresses model selection and hy-

perparameter optimisation for LM fine-tuning, primarily when resource limitations exist. This motivates our proposal of XAutoLLM, a meta-learning-based AutoML system designed to fill these gaps in efficiency and performance via multi-objective optimisation.

3 Proposal

We propose XAutoLLM, the first AutoML system to combine model selection and hyperparameter optimisation for fine-tuning LMs. We leverage an explicable and extensible meta-learning method that employs accumulated knowledge from previous evaluations to bias the initial search space towards solutions similar to those that have proven effective in related tasks. In AutoML terminology, this constitutes a sophisticated *warm-start* mechanism that accelerates convergence by exploiting task similarities identified through meta-features.

The core innovation of XAutoLLM lies in its ability to dynamically adjust the search space based on task-level similarities and multi-objective optimisation criteria. This capability enables resource-efficient fine-tuning under constrained computational budgets. Our framework utilises the optimisation strategy and pipeline abstraction from AutoGOAL[1] (Estevez-Velarde et al., 2020), alongside embedding and generative models for various NLP tasks available within AutoGOAL's algorithm pool. Table 2 summarises the language models available in the pool at the time of writing, along with their respective categories. While the original system supports these models solely for inference (Estevanell-Valladares et al., 2024), XAutoLLM enables fine-tuning via three distinct methods: Low-Rank Adaptation (Hu et al., 2021), traditional fine-tuning, and a partial fine-tuning approach that freezes all parameters except those of the final layers.

3.1 Process Overview

XAutoLLM integrates prior experience into an iterative optimisation pipeline. First, resource constraints are defined and relevant historical evaluations (*experiences*) are retrieved from a centralised repository (Section 3.2). Next, meta-features capturing both the current task properties (Section 3.2.1) and system capabilities (Section 3.2.2) are extracted to assess task complexity and available computational resources. The search space is then probabilistically adjusted using these insights, emphasising configurations relevant to the current scenario and historically associated with high performance while de-emphasising those linked to errors (Section 3.3). Finally, the fine-tuning routine commences, guided by these refined configurations, and the results of each evaluation are recorded in the experience store to inform future optimisation cycles.

[1]Open-source framework available at `https://github.com/autogoal/autogoal`. Its license allows for dealing with the software without restriction

Category	Language Model
Encoders	BERT (Devlin et al., 2018)
	DistilBERT (Sanh et al., 2020)
	RoBERTa (Liu et al., 2019)
	XLM-RoBERTa (Conneau et al., 2020)
	DeBERTa (He et al., 2021)
	DeBERTaV3 (He et al., 2023)
	MDeBERTaV3 (He et al., 2023)
	ALBERT-v1 (Lan et al., 2019)
	ELECTRA (Clark et al., 2020)
Generative	T5 (Raffel et al., 2020)
	FLAN-T5 (Chung et al., 2024)
	GPT-2 (Radford et al., 2019)
	PHI-3 (Abdin et al., 2024)

Table 2: LMs available in AutoGOAL's algorithm pool.

3.2 Experience Store

Central to XAutoLLM's functionality is its repository of past evaluations, referred to as *experiences*. These records capture key elements such as algorithm choices, parameter configurations, task meta-features, and system hardware details, enabling informed decision-making in subsequent optimisation cycles.

3.2.1 Task Meta-Features.

Task meta-features provide insights into dataset complexity, variability, and class distribution (see Table 3). For example, entropy and imbalance ratios reflect dataset skewness, while document length statistics capture textual complexity. These features are critical for tailoring fine-tuning strategies to specific task requirements.

3.2.2 System Meta-Features.

Hardware characteristics (e.g., number of CPU cores, RAM size) are equally crucial for ensuring that fine-tuning remains feasible within resource constraints (see Table 4). For instance, while large

Category	Feature	Rationale
Instance/Class	Nr Instances	Scalability
	Nr Classes	Complexity
	Entropy	Distribution
	Min Cls Prob	Imbalance
	Max Cls Prob	Imbalance
	Imbalance Ratio	Skewness
Documents	Avg. Length	Complexity
	Std. Length	Variability
	Coef. Var. Length	Consistency
Landmark	PCA + D.Tree Acc.	Baseline

Table 3: Rationale for selected dataset meta-features

Feature	Rationale
CPU physical cores CPU logical cores	Parallel Processing Capabilities
CPU max frequency	Processing Speed
RAM size VRAM size	Data Handling Capabilities

Table 4: Rationale for selected system meta-features

models like GPT-4 (OpenAI, 2023) may yield superior results compared to smaller alternatives, their feasibility depends heavily on available computational resources.

XAutoLLM constructs a holistic representation of each optimisation scenario by combining task-specific and system-level meta-features, enabling robust similarity assessments across diverse contexts.

3.3 Warm-Start Optimization

Building on our *experience store*, we propose a warm-start mechanism that biases the initial search towards promising fine-tuning configurations. Given a new task with meta-features t_T, we measure its similarity to each stored experience via a distance function $\mathrm{Dist}(t_T, t_i)$ (e.g. Euclidean or Cosine).

To modulate the influence of past experiences based on their similarity to T, we introduce an adaptive decay parameter β. Two formulations are proposed:

$$\textbf{Std-Only:} \quad \beta = \frac{\beta_{\text{scale}}}{\sigma_d + \epsilon}$$

$$\textbf{Std-Plus-Mean:} \quad \beta = \frac{\beta_{\text{scale}}}{\max(\sigma_d, \epsilon) + \mu_d}$$

where σ_d and μ_d denote the standard deviation and mean of the distances $\{d_i\}$ across all experiences, respectively, and $\epsilon > 0$ is a small constant to prevent division by zero.

Utility-Driven Prioritization. XAutoLLM assigns a utility score u_i to each positive experience (i.e., those with valid performance scores) to bias the search process toward promising configurations. We consider three approaches:

Weighted Sum. The utility score for experience e_i is computed as

$$u_i = w_{\text{F1}} \cdot \frac{F1_i}{F1_{\max}} + w_{\text{time}} \cdot \left(1 - \frac{et_i - et_{\min}}{et_{\max} - et_{\min}}\right)$$

where $F1_i$ represents the $F1_{\text{macro}}$ score, et_i is the evaluation time, and the weights $w_{\text{F1}}, w_{\text{time}} \geq 0$ satisfy $w_{\text{F1}} + w_{\text{time}} = 1$.

Linear Front. Using non-dominated sorting, experiences are assigned to N fronts. The utility score is given by

$$u_i = \frac{N - r_i}{N}$$

where r_i denotes the rank of experience e_i. This linear scaling ensures that the highest-ranked experience attains a score of 1, with a proportional decrement for lower-ranked experiences.

Logarithmic Front. To accentuate differences among top-ranked experiences, we propose a logarithmic ranking-based utility function:

$$u_i = \frac{\ln(N - r_i + 1)}{\ln(N + 1)}$$

This formulation compresses the utility differences among lower-ranked experiences while preserving a steeper gradient among the highest-ranked ones, which is particularly useful in scenarios where a slight improvement in rank corresponds to a substantial performance gain.

These utility scores modulate the learning rate adjustments. Specifically, the learning rate for a positive experience e_i is updated as

$$\alpha_i^+ = \alpha_{\max}^+ \, u_i \, e^{-\beta d_i}$$

whereas, conversely, negative experiences are de-emphasised using

$$\alpha_i^- = \alpha_{\max}^- \, e^{-\beta d_i}$$

Here, $\alpha^{+}_{\max}$ and $\alpha^{-}_{\max}$ are parameters that can be either fixed values or adaptively computed based on the amount of available positive and negative experiences, respectively.

Probabilistic Model Adjustment. The probabilistic model governing configuration parameters is updated iteratively:

$$P(c \mid \theta) \leftarrow (1 - \alpha_i^{+})\, P(c \mid \theta) + \alpha_i^{+}\, P_i(c \mid \theta),$$

$$P(c \mid \theta) \leftarrow (1 + \alpha_i^{-})\, P(c \mid \theta) - \alpha_i^{-}\, P_i(c \mid \theta)$$

Positive adjustments increase the probability of configurations similar to those associated with successful experiences, while negative adjustments decrease the probability of configurations linked to suboptimal outcomes. This warm-start mechanism enables XAutoLLM to efficiently explore promising regions of the search space while avoiding redundant evaluations of suboptimal configurations. See Appendix A for additional implementation details.

4 Experimentation

We evaluated our **warm-started XAutoLLM** proposal by comparing its performance and the quality of generated solutions against a version of **XAutoLLM without meta-learning**. This robust baseline corresponds to the proposal from Estevanell-Valladares et al. (2024), which has demonstrated strong performance in both single-objective (Estevez-Velarde et al., 2020) and multi-objective scenarios, albeit with an extended algorithm pool (including our fine-tuning implementations).

We designed two experimental setups to evaluate our proposal on diverse text classification scenarios. First, we conducted a pilot evaluation for **Single-Objective** optimisation scenarios to directly compare performance improvements (Section 4.1). Then, we assessed **Multi-Objective** optimisation capabilities to evaluate our proposal's ability to balance resource efficiency and performance (Section 4.2). See Appendix A for supplementary experimentation details.

Table 5 summarises the classification tasks selected for evaluation. We chose semantically diverse tasks with varying computational resource requirements and performance expectations. For instance, Reusens et al. (2024) reported that the maximum $F1_{macro}$ scores for LIAR and MELD were 0.23 and 0.40, respectively, compared to 0.89 for SST2 and 0.93 for AGNews. Similarly, AGNews comprises nearly ten times the number of samples as LIAR and MELD.

Class. Task	Dataset	Cls	Size
Fake news	LIAR (Wang, 2017)	6	12,836
Topic	AG News (Zhang et al., 2015)	4	127,600
Polarity	SST2 (Socher et al., 2013)	2	68,221
Emotion	MELD (Poria et al., 2018)	7	13,708

Table 5: Text Classification tasks selected for evaluation.

Experimental Setup

All experiments were executed on an i9-9900K CPU with 127 GB RAM and an RTX TITAN GPU (24 GB VRAM). We preemptively generated meta-learning experiences by running XAutoLLM without meta-learning for 48 hours on each task (Table 6). We used $F1_{macro}$ and evaluation time (ET) as optimisation objectives with a one-hour timeout per pipeline evaluation.

Dataset	Generated			Available		
	Pos	Neg	Total	Pos	Neg	Total
LIAR	100	236	336	116	480	596
SST2	33	122	155	183	594	777
MELD	68	190	258	148	526	674
AG News	15	168	183	216	548	764

Table 6: Disposition of experiences participating in the experiments.

4.1 Single-Objective Evaluation

The single-objective setup compared the baseline system against three warm-start configurations with varying levels of initial bias (low, moderate, high). These configurations were derived by generating fourteen parameter combinations, grouping them by Total Variation (TV), and selecting the top representatives from each group. We employed the Std-Only beta scale for regulating distances (See Section 3.3).

Figure 1 shows the resulting probabilities for fine-tuning methods after applying warm-start configurations on LIAR. Configurations were sorted by their variation over the baseline distribution. Level groups were demarcated iteratively based on approximately doubling variation differences.

Table 7 reports results from six runs per configuration on LIAR and SST2 over 24 hours. Across both datasets, while peak performance improvements were modest, significant gains were ob-

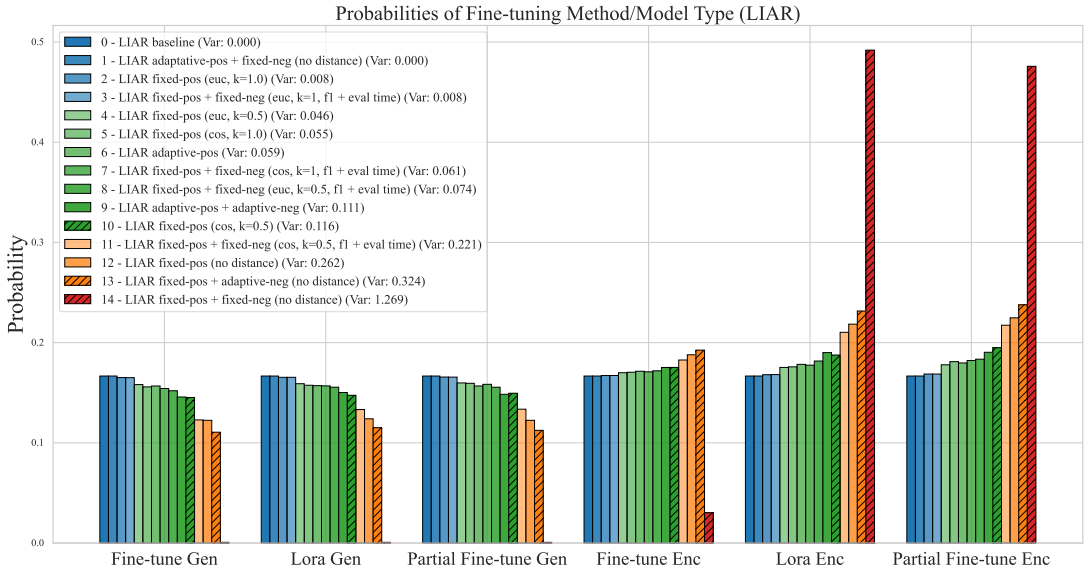

Figure 1: Fine-tuning method probabilities for LIAR under 14 meta-learning configurations, sorted by TV relative to the uniform baseline. Blue indicates near-baseline, while green, orange, and red denote low, moderate, and high bias levels. Patterned markers (10, 13, 14) represent our selected experiment configurations.

Dataset	Config.	Max $F1_m$	Mean $F1_m$	$TT50$ (h)	$TT75$ (h)	$TT90$ (h)	No. Eval	E. Ratio
LIAR	Baseline	0.248 ±0.018	0.09 ±0.004	2.00	6.38	8.15	**173**	0.69
	Low WS	**0.253 ±0.006**	**0.11 ±0.008**	**1.35**	**4.10**	9.05	166	0.61
	Mod WS	0.251 ±0.015	**0.11 ±0.008**	1.57	4.88	**6.43**	165	0.46
	High WS	0.247 ±0.006	0.10 ±0.009	1.37	5.42	10.74	156	**0.24**
SST2	Baseline	0.928 ±0.018	0.56 ±0.053	1.69	2.07	4.64	85	0.83
	Low WS	0.917 ±0.016	**0.59 ±0.063**	1.28	2.41	5.09	**98**	0.80
	Mod WS	**0.941 ±0.004**	0.56 ±0.064	0.70	3.88	5.21	55	0.69
	High WS	0.932 ±0.002	0.56 ±0.058	**0.41**	**0.41**	**2.23**	58	**0.58**

Table 7: Overview of XAutoLLM performance on optimising $F1_{macro}$ for LIAR and SST2. Results are averaged over six runs with different seeds. 'Max $F1_m$' and 'Mean $F1_m$' show the mean and standard deviation, respectively; 'TT50', 'TT75', and 'TT90' report the average time to reach 50%, 75%, and 90% $F1_m$; and 'No. Eval' and 'E. Ratio' indicates the average number of pipeline evaluations and the ratio of such evaluations that were errors.

served in the convergence speed against the baseline (reduced $TT50$, $TT75$ and $TT90$). On LIAR, none of the configurations outperformed the baseline on max $F1_{macro}$ with statistical significance, but mean $F1_{macro}$ exhibited an overall significant effect ($p < 0.01$). However, the pairwise comparisons were insignificant after correction, likely due to insufficient samples ($n = 6$).

Results on SST2 show a similar pattern of incremental yet not always statistically significant gains in maximum performance, with Mod WS reaching the highest max $F1_{macro}$. Although the ANOVA (McHugh, 2011) test reached significance for max $F1_{macro}$ ($p = 0.031$), the Friedman (Pereira et al., 2015) test yielded a slightly more conservative outcome, highlighting the variability in effect sizes across runs.

The efficiency advantages of warm-starting were more pronounced in both tasks. In LIAR, $TT50$ and $TT75$ decreased notably for Low WS, while Mod WS reached $TT90$ almost 1.7 hours faster than the baseline. In SST2, Mod WS, the best performing in max $F1_{macro}$, displayed a similar pattern as LIAR's best, improving $TT50$ and $TT75$ but with slightly worse $TT90$ compared to baseline. High WS displayed the best time-to-threshold values in SST2, where it at least improved 2 times against baseline. Additionally, we notice a direct proportionality between the initial bias level and the ratio of errors discovered in the search.

4.2 Multi-Objective Evaluation

Multi-objective evaluations were conducted across all four datasets (Table 5). For each task, we evalu-

ated the original LIAR candidates (Figure 1) alongside six newly selected candidates. These additional candidates were chosen by independently selecting warm-start configurations with median and maximum TV from each group, derived from over 180 combinations evaluated per task (see Appendix B for details on the selected candidate configurations).

	Config.	Max $F1_m$	Mean $F1_m$	Min ET	Mean ET	HV	PO R.	No. Eval	Error Ratio
LIAR	Baseline	**0.26**	**0.10**	12	540	**0.67**	**1**	**336**	0.70
	Low (LIAR)	**0.26**	**0.10**	16	480	0.15	2	197	0.70
	Low (Med)	0.25	0.09	31	380	0.54	2	220	0.69
	Low (Max)	0.25	0.09	21	410	0.12	3	190	0.66
	Mod (LIAR)	**0.26**	**0.10**	36	462	0.02	2	132	0.53
	Mod (Med)	0.24	**0.10**	13	469	0.07	2	146	0.61
	Mod (Max)	0.25	0.08	44	516	0.07	4	121	0.39
	High (LIAR)	0.25	**0.10**	**6**	**153**	0.30	**1**	302	**0.09**
	High (Med)	0.25	**0.10**	9	277	0.18	**1**	193	0.33
	High (Max)	**0.26**	0.09	12	252	0.13	**1**	208	0.25
SST2	Baseline	**0.94**	**0.64**	**61**	1065	0.09	**1**	**155**	0.78
	Low (LIAR)	0.90	0.48	373	1148	0.30	2	87	0.82
	Low (Med)	0.90	0.52	227	**840**	0.05	3	62	0.83
	Low (Max)	**0.94**	0.58	252	**784**	0.03	2	98	0.81
	Mod (LIAR)	0.93	0.56	245	996	**0.39**	**1**	59	0.64
	Mod (Med)	**0.94**	0.52	132	1030	0.09	2	34	**0.55**
	Mod (Max)	0.93	0.52	184	1170	0.13	2	58	0.51
	High (LIAR)	0.92	0.62	365	1160	0.04	3	42	0.61
	High (Med)	**0.94**	0.53	164	844	0.18	**1**	52	0.68
	High (Max)	**0.94**	0.61	320	857	0.31	**1**	53	0.79
MELD	Baseline	0.41	**0.14**	39	769	0.19	3	258	0.73
	Low (LIAR)	**0.46**	**0.14**	20	532	0.11	**1**	150	0.64
	Low (Med)	0.45	0.11	17	387	0.52	**1**	229	0.64
	Low (Max)	0.39	0.09	30	477	**0.62**	**1**	186	0.65
	Mod (LIAR)	0.40	0.11	26	514	0.00	3	106	0.39
	Mod (Med)	0.40	0.11	36	546	0.05	3	130	0.52
	Mod (Max)	0.38	0.09	24	590	0.14	3	110	0.52
	High (LIAR)	0.44	**0.14**	**7**	**179**	0.16	**1**	**260**	**0.10**
	High (Med)	0.43	0.13	21	466	0.47	2	124	0.45
	High (Max)	0.42	0.12	12	322	0.01	2	233	0.51
AG NEWS	Baseline	0.90	0.52	308	1091	0.04	2	**254**	0.91
	Low (LIAR)	**0.93**	**0.73**	349	1183	0.03	2	93	0.90
	Low (Med)	0.92	0.65	665	1589	**0.41**	**1**	83	0.89
	Low (Max)	**0.93**	0.60	560	1164	0.00	**1**	77	0.90
	Mod (LIAR)	0.92	0.46	404	1345	0.24	**1**	50	0.80
	Mod (Med)	**0.93**	0.59	484	1102	0.02	2	48	0.79
	Mod (Max)	0.92	0.56	**249**	1402	0.03	**1**	57	0.73
	High (LIAR)	**0.93**	0.46	318	1437	0.01	2	45	**0.71**
	High (Med)	**0.93**	0.51	253	**833**	0.19	**1**	58	0.86
	High (Max)	0.92	0.54	350	1576	0.02	3	46	0.73

Table 8: Overview of performance of XAutoLLM on optimizing $F1_{macro}$ and ET on LIAR, SST2, MELD and AG NEWS. Data for each candidate corresponds to one run with a shared random seed.

Table 8 presents the results of the experiment. In addition to standard performance metrics like maximum $F1_{macro}$ and minimum evaluation time (ET), we report Hypervolume (HV) and Pareto-Optimality Rank (PO R.). The ranking follows the methodology outlined by Ibrahim et al. (2024) but with HV, max $F1_{macro}$ and min ET as the performance indicators to capture the tradeoffs of the candidates.

Although LIAR's baseline remains highly competitive—having both the highest max $F1_{macro}$ and the top HV—it is not always optimal in other scenarios. For instance, High (LIAR) in LIAR achieves PO Rank 1 by drastically reducing mean evaluation time (153 vs 387) without sacrificing too much performance (max $F1_{macro}$=0.25). A similar issue arises in SST2: while the baseline again attains the highest max $F1_{macro}$, some WS approaches produce faster convergence and ultimately surpass it in HV (e.g., Low (LIAR) attains a much lower min ET of 12 and a similar performance).

In MELD and AG NEWS, the meta-learning approaches outperform the baselines on multi-objective metrics. In MELD, Low (Med) achieves the best HV = 0.52 (against the baseline's 0.19) and near-max performance (0.45). Its PO Rank 1 indicates a difference in Pareto front quality compared to the baseline. In AG NEWS, Low (Med) delivers an HV = 0.41—far exceeding the baseline's 0.04—while improving max $F1_{macro}$ from 0.90 to 0.92. Across both datasets, high initial bias more consistently reduces evaluation time and error rates while maintaining or slightly surpassing the baseline in peak $F1_{macro}$.

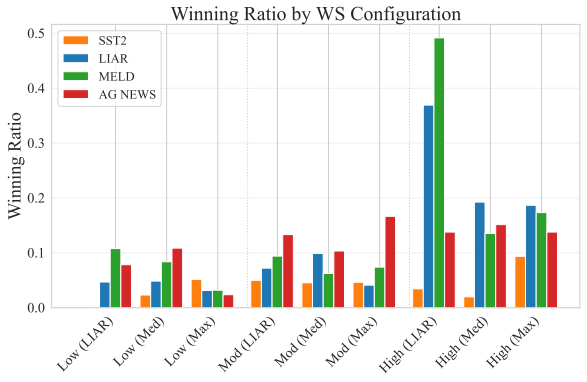

Figure 2: Ratio of fine-tuning pipelines outperforming the strong baseline per configuration and task.

From an exploration-efficiency perspective, WS candidates show clear advantages. Figure 2 shows the rate of discovered solutions by each warm-start configuration that improved over the robust baseline balancing efficiency and performance. High configurations produced the best winning ratios, with almost 50% of the pipelines discovered by High (LIAR) outperforming the baseline while diminishing up to seven times the ratio of discovered errors during the search.

Overall, these outcomes highlight that the warm-

start strategy simultaneously confers tangible benefits across multiple objectives, lowering error rates and search overhead while preserving or improving classification performance.

5 Discussion

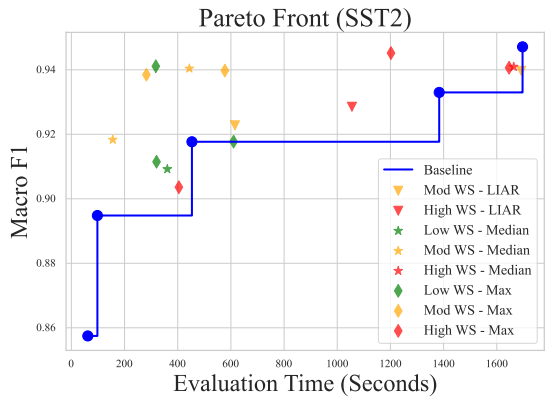

Figure 3: LM fine-tuning pipelines discovered by XAutoLLM on SST2.

Our results demonstrate the importance of leveraging *prior knowledge* in AutoML. The most significant difference with the baseline comes in tasks where relevant experience from similar tasks is available (see Figure 4 showing MELD and LIAR are most alike). Conversely, tasks with little or no relevant background knowledge benefit less from meta-learning. This highlights the need to balance exploitation and exploration adaptively, leaving room for standard AutoML searches when historical data is sparse. By discriminating less meaningful experiences, **XAutoLLM** effectively avoids over-biasing the search space. This was the case with SST2 and AG News, for which the system lacked a large relevant experience pool but still discovered pipelines outperforming a robust baseline (see Figure 3 and Appendix C for all Pareto Front visualizations).

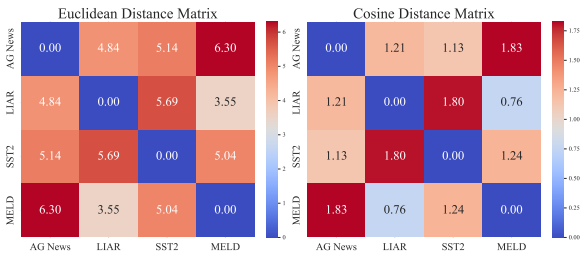

Figure 4: Distance between Tasks according to their meta-features (See Section 3.2.1).

Combining experience discrimination with adaptive probabilistic adjustments yields the best of both worlds: faster convergence to near-optimal solutions when prior knowledge is significant while remaining robust in problems with fewer references. For instance, on SST2, our approach discovered pipelines that reduced evaluation time by up to 6× while maintaining macro $F1$ performance within one percentage point of the global best. Similarly, experiments on LIAR and MELD show winning ratios of around 40% and 50%, respectively, meaning that a substantial fraction of the solutions discovered by **XAutoLLM** strike a superior balance between efficiency and performance. This supports our central hypothesis that *integrating prior knowledge is crucial for enhancing AutoML pipelines*.

A core motivation of our framework is to reduce the carbon footprint and environmental toll of repeated large-scale language model fine-tuning. By systematically reusing insights from past runs, **XAutoLLM** significantly reduces redundant evaluations and lowers the overall error rate during the search. Beyond simply lowering compute hours, this approach aligns with the growing *Green AI* ethos in NLP (Wang et al., 2023b), emphasising the importance of responsible resource usage. Our experiments illustrate that warm-start meta-learning not only improves performance but also streamlines the search process, yielding algorithms that better balance efficiency and performance.

6 Conclusions

We introduced **XAutoLLM**, the first AutoML framework that jointly addresses model selection and hyperparameter optimisation for LM fine-tuning. Using past experiences to warm-start the search process, our method achieves up to 2× faster convergence, sevenfold lower error ratios, and discovered pipelines with a 6× reduction in evaluation time while maintaining highly competitive macro $F1$ scores. With winning ratios of 40% and 50% on LIAR and MELD, respectively, our results demonstrate that **XAutoLLM** effectively balances performance and efficiency. These improvements support a more sustainable, *Green AI* approach by significantly reducing redundant computations and producing more resource-efficient fine-tuned LMs. Our open-source framework is available for the language modelling research community (Appendix A).

7 Limitations

We identify some limitations to our study that highlight avenues for further investigation:

Computational Overhead. Although our meta-learning warm-start mechanism reduces the number of redundant evaluations, language model fine-tuning remains computationally expensive. Users with limited GPU or CPU resources might face practical hurdles when repeatedly tuning larger transformer architectures. Nonetheless, our results show that even moderate hardware settings can benefit from XAutoLLM's efficiency gains.

Task and Data Scope. Our current experiments target text classification tasks of moderate scale. While these findings strongly support the effectiveness of XAutoLLM, further research is needed to confirm its utility in more diverse tasks such as sequence labelling or multi-modal pipelines, as well as in domains with highly specialised data (e.g., biomedical or legal corpora). Such explorations would help affirm the broad generalizability of our approach.

Risk of Negative Transfer. XAutoLLM adaptively weighs past experiences based on their relevance to a new task. However, negative transfer remains possible if the experience repository predominantly contains data from tasks that differ markedly or contain suboptimal configurations. Although our decay mechanisms mitigate this risk, future enhancements might include automated outlier detection or more selective filtering strategies to safeguard against inconsistent past knowledge.

Lack of Ablation Studies. We have demonstrated the overall value of XAutoLLM's meta-learning approach, but deeper insight could be gained through dedicated ablation studies. Specifically, isolating the impact of different distance metrics or rank-based scoring schemes would illuminate the contributions of each component. Our positive results indicate these elements collectively improve efficiency and performance, yet targeted experiments would offer more granular guidance.

Framework Coupling. Our approach builds on the AutoGOAL framework to leverage its pipeline abstraction, multi-objective optimisation capabilities and broad algorithm pool. While this integration streamlines experimentation, transferring XAutoLLM's warm-start mechanism to other AutoML platforms may require additional adaptation. The underlying concepts, however, remain framework-agnostic and can be extended with appropriate engineering.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

Nesrine Bannour, Sahar Ghannay, Aurélie Névéol, and Anne-Laure Ligozat. 2021. Evaluating the carbon footprint of nlp methods: a survey and analysis of existing tools. In *Proceedings of the second workshop on simple and efficient natural language processing*, pages 11–21.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. *Preprint*, arXiv:1911.02116.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ernesto L Estevanell-Valladares, Yoan Gutiérrez, Andrés Montoyo-Guijarro, Rafael Muñoz-Guillena, and Yudivián Almeida-Cruz. 2024. Balancing efficiency and performance in nlp: A cross-comparison of shallow machine learning and large language models via automl. *Procesamiento del Lenguaje Natural*, 73:221–233.

Suilan Estevez-Velarde, Yoan Gutiérrez, Andrés Montoyo, and Yudivián Almeida Cruz. 2020. Automatic discovery of heterogeneous machine learning pipelines: An application to natural language processing. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3558–3568.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing. *Preprint*, arXiv:2111.09543.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. 2019. *Automated Machine Learning*. Springer.

Amin Ibrahim, Azam Asilian Bidgoli, Shahryar Rahnamayan, and Kalyanmoy Deb. 2024. A novel pareto-optimal ranking method for comparing multi-objective optimization algorithms. *arXiv preprint arXiv:2411.17999*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. *CoRR*, abs/1909.11942.

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. *Preprint*, arXiv:1907.11692.

Daqin Luo, Chengjian Feng, Yuxuan Nong, and Yiqing Shen. 2024. Autom3l: An automated multimodal machine learning framework with large language models. In *Proceedings of the 32nd ACM International Conference on Multimedia*, MM '24, page 8586–8594, New York, NY, USA. Association for Computing Machinery.

Neeratyoy Mallik, Edward Bergman, Carl Hvarfner, Danny Stoll, Maciej Janowski, Marius Lindauer, Luigi Nardi, and Frank Hutter. 2024. Priorband: Practical hyperparameter optimization in the age of deep learning. *Advances in Neural Information Processing Systems*, 36.

Mary L McHugh. 2011. Multiple comparison analysis testing in anova. *Biochemia medica*, 21(3):203–209.

Clint Morris, Michael Jurado, and Jason Zutty. 2024. Llm guided evolution-the automation of models advancing models. *arXiv preprint arXiv:2403.11446*.

OpenAI. 2023. Gpt-4 technical report. Technical report, OpenAI. ArXiv:2303.08774.

Dulce G Pereira, Anabela Afonso, and Fátima Melo Medeiros. 2015. Overview of friedman's test and post-hoc analysis. *Communications in Statistics-Simulation and Computation*, 44(10):2636–2653.

Soujanya Poria, Devamanyu Hazarika, Navonil Majumder, Gautam Naik, Erik Cambria, and Rada Mihalcea. 2018. Meld: A multimodal multi-party dataset for emotion recognition in conversations. *arXiv preprint arXiv:1810.02508*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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. *Journal of Machine Learning Research*, 21(140):1–67.

Manon Reusens, Alexander Stevens, Jonathan Tonglet, Johannes De Smedt, Wouter Verbeke, Seppe vanden Broucke, and Bart Baesens. 2024. Evaluating text classification: A benchmark study. *Expert Systems with Applications*, 254:124302.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *Preprint*, arXiv:1910.01108.

Esraa Sayed, Mohamed Maher, Omar Sedeek, Ahmed Eldamaty, Amr Kamel, and Radwa El Shawi. 2024. Gizaml: A collaborative meta-learning based framework using llm for automated time-series forecasting. In *EDBT*, pages 830–833.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2024. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Alexander Tornede, Difan Deng, Theresa Eimer, Joseph Giovanelli, Aditya Mohan, Tim Ruhkopf, Sarah Segel, Daphne Theodorakopoulos, Tanja Tornede, Henning Wachsmuth, et al. 2023. Automl in the age of large language models: Current challenges, future opportunities and risks. *arXiv preprint arXiv:2306.08107*.

Joaquin Vanschoren. 2019. *Meta-Learning*, pages 35–61. Springer International Publishing, Cham.

Chi Wang, Susan Xueqing Liu, and Ahmed H. Awadallah. 2023a. Cost-effective hyperparameter optimization for large language model generation inference. *Preprint*, arXiv:2303.04673.

William Yang Wang. 2017. " liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*.

Xiaorong Wang, Clara Na, Emma Strubell, Sorelle Friedler, and Sasha Luccioni. 2023b. Energy and carbon considerations of fine-tuning bert. *arXiv preprint arXiv:2311.10267*.

Lanning Wei, Zhiqiang He, Huan Zhao, and Quanming Yao. 2023. Unleashing the power of graph learning through llm-based autonomous agents. *arXiv preprint arXiv:2309.04565*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*.

Caiyang Yu, Xianggen Liu, Yifan Wang, Yun Liu, Wentao Feng, Xiong Deng, Chenwei Tang, and Jiancheng Lv. 2024. Gpt-nas: Neural architecture search meets generative pre-trained transformer model. *Big Data Mining and Analytics*.

Shujian Zhang, Chengyue Gong, Lemeng Wu, Xingchao Liu, and Mingyuan Zhou. 2023. Automlgpt: Automatic machine learning with gpt. *arXiv preprint arXiv:2305.02499*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*.

A Additional Implementation Details and Experimental Configurations

In this section, we provide key implementation details to ensure that our work is fully reproducible. All configuration candidates used in our single-objective experiments are fully specified in the main text (see Figure 1). In contrast, candidates for multi-objective experiments are available in Appendix B due to the extremely high number of tested configurations. In our evaluations, candidate configurations were designed with two distinct learning rate schemes and distance discrimination strategies, as detailed below.

A.1 Learning Rate Configuration and Update Strategy

We adopt a dual-mode configuration for the learning rate updates applied to the probabilistic model. In experiments employing fixed learning rates, we set the parameters to

$$\alpha^{+}_{\max} = 0.05 \quad \text{and} \quad \alpha^{-}_{\max} = -0.02.$$

For configurations using adaptive learning rates, the values are computed as

$$\alpha^{+}_{\max} = \frac{1}{N_{\text{pos}}} \quad \text{and} \quad \alpha^{-}_{\max} = -\frac{1}{N_{\text{neg}}}$$

Where N_{pos} and N_{neg} denote the number of positive and negative experiences, respectively. Although these rates are expressed with positive and negative signs to indicate the direction of the update (reinforcing or de-emphasizing a configuration), all update steps are executed using the absolute values.

A.2 Normalization of Meta-Features

All meta-features used for computing distances are standardized using a standard scaler normalizer. This normalizer computes the mean and standard deviation of the feature vectors (with a small epsilon added to avoid division by zero) and returns the standardized data. This ensures that distance computations are robust and comparable across features.

A.3 Beta Scale and Utility Functions

For the decay parameter β, two formulations are employed: the *std-only* beta scale is used in single-objective experiments, whereas the *std-plus-mean* beta scale is applied in multi-objective settings.

All candidates for the single-objective experiments utilize a weighted sum approach with the $F1$ score weight set to 1 and the evaluation time weight set to 0. Detailed specifications of candidate configurations can be found in the visualizations provided in the respective sections (Section 4.1 for single-objective, and Appendix B for multi-objective).

A.4 Experimental Setup and Computational Resources

The main text fully discloses our experimental setup (See Section 4). All experiments were conducted on an Intel i9-9900K CPU with 127 GB RAM and an RTX TITAN GPU (24 GB VRAM). Each pipeline evaluation was granted a one-hour timeout.

A.5 Framework Overview and Dependencies

XAutoLLM is implemented on top of the AutoGOAL framework (Estevanell-Valladares et al.,

2024; Estevez-Velarde et al., 2020), leveraging its optimization strategy and abstractions. Our implementation is developed in Python and utilizes the HuggingFace Transformers library (Wolf et al., 2019) to access pre-trained language models. A complete list of dependencies, environment setup instructions, and detailed documentation on how to run the experiments (and statistical testing), reproduce the results, and navigate the codebase is provided in the repository.

The code and all associated materials can be accessed at the following anonymous GitHub repository: `https://anonymous.4open.science/r/XAutoLLM-A010` (currently private for blind review; will be made public upon completion of the review process).

B Multi-Objective Initial Probabilities

This section provides detailed illustrations of the *initial probability distributions* assigned to each fine-tuning method under varying meta-learning configurations. These figures supplement our multi-objective experiments from Section 4.2 by visualising how each configuration biases the AutoML search *before* any evaluations.

Recall that we generated up to 180 candidate configurations per dataset by systematically varying:

1. Inclusion/exclusion of *positive* (successful) and *negative* (error) past experiences,
2. Utility functions (e.g., weighted sum, linear front, logarithmic front),
3. Distance metrics (Euclidean, Cosine) and their scaling,
4. $\alpha^{+}_{\max}$ and $\alpha^{-}_{\max}$ values (fixed or adaptive) (See Section 3.3).

Each configuration yields a distinct initial probability vector for the available fine-tuning methods, with deviations from the baseline distribution measured via *Total Variation* (TV). Grouping configurations by TV allows us to categorise them into *low*, *moderate*, and *high* bias levels relative to the baseline's uniform initialisation.

LIAR. Figure 5 shows the initial probabilities of using each fine-tuning method for the LIAR dataset, sorted by their overall difference from the baseline. Blue bars indicate the baseline configuration, whereas green, orange, and red bars represent configurations increasingly diverging from the baseline. We marked selected *representative* configurations (patterned bars) for each bias level.

SST2. Figure 6 illustrates the same analysis on SST2. Although the dataset differs substantially from LIAR regarding meta-features (e.g., number of classes, data size, label distribution), we observe a similar pattern in how the bias level shifts probabilities among alternative fine-tuning methods. The High (Max) configuration notably shows more aggressiveness than LIAR's.

MELD. Figure 7 shows the MELD dataset's initial distributions. As discussed in Section 4, MELD shares some meta-feature similarities with LIAR (see Figure 4), causing some distributions to concentrate around methods found promising in LIAR's prior runs.

AG News. Lastly, Figure 8 displays the candidate configurations for AG NEWS, a large corpus with four news categories.

These visualisations underscore how our meta-learning strategy adapts the search space before optimisation begins. By systematically adjusting the initial probabilities, XAutoLLM avoids mindlessly searching all possibilities and exploits task similarities to emphasise configurations that are historically more successful or resource-feasible.

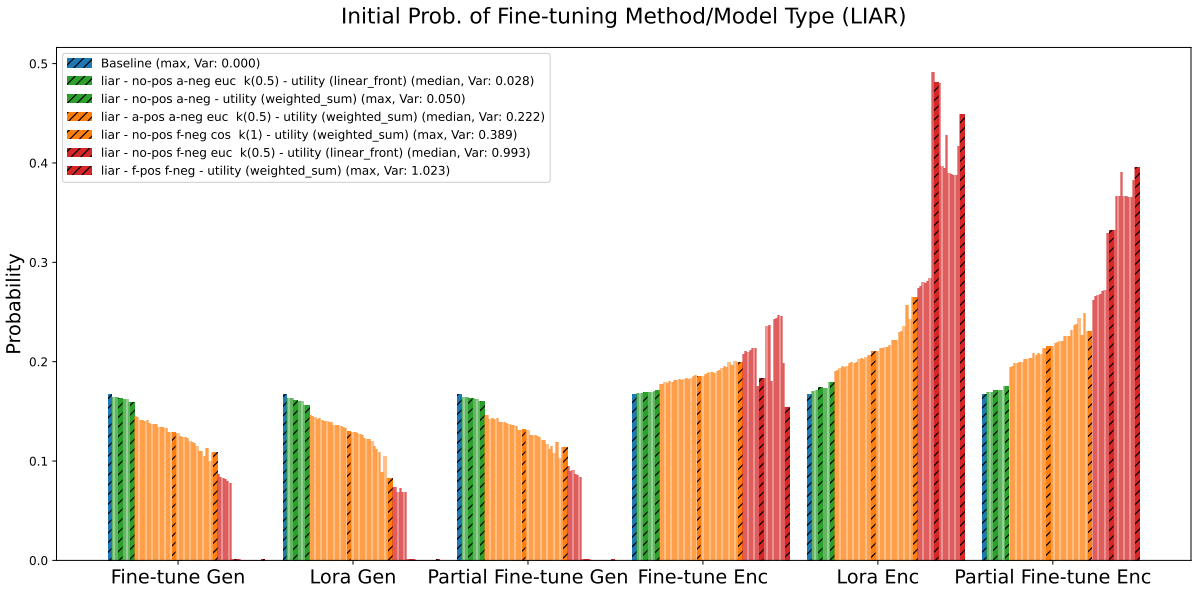

Figure 5: Initial probability distributions for fine-tuning methods on LIAR.

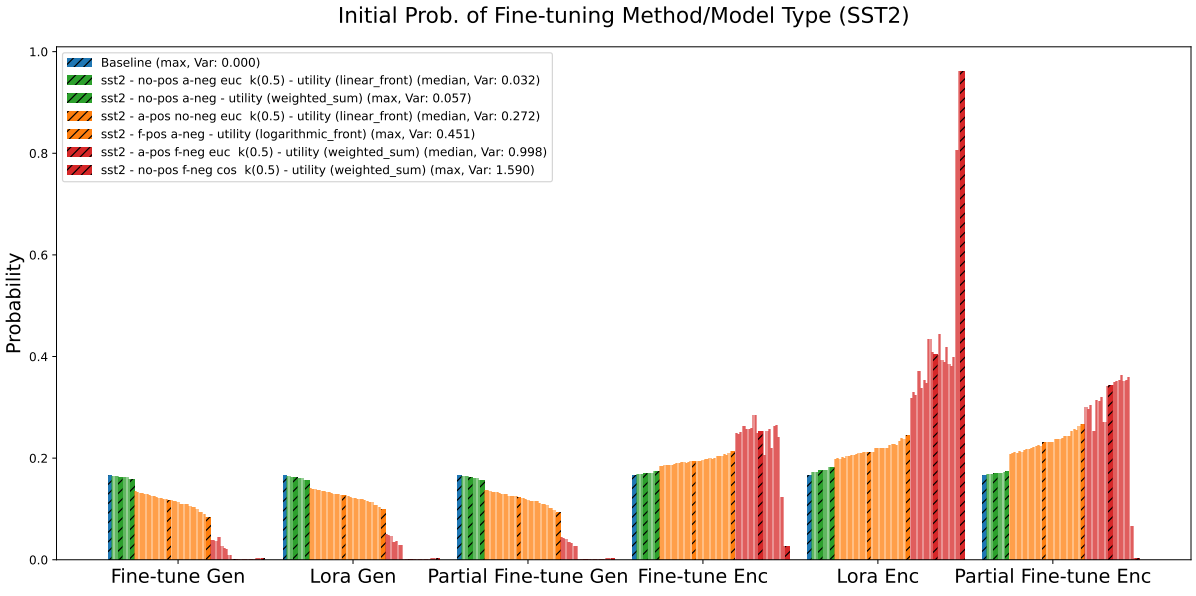

Figure 6: Initial probability distributions for fine-tuning methods on SST2

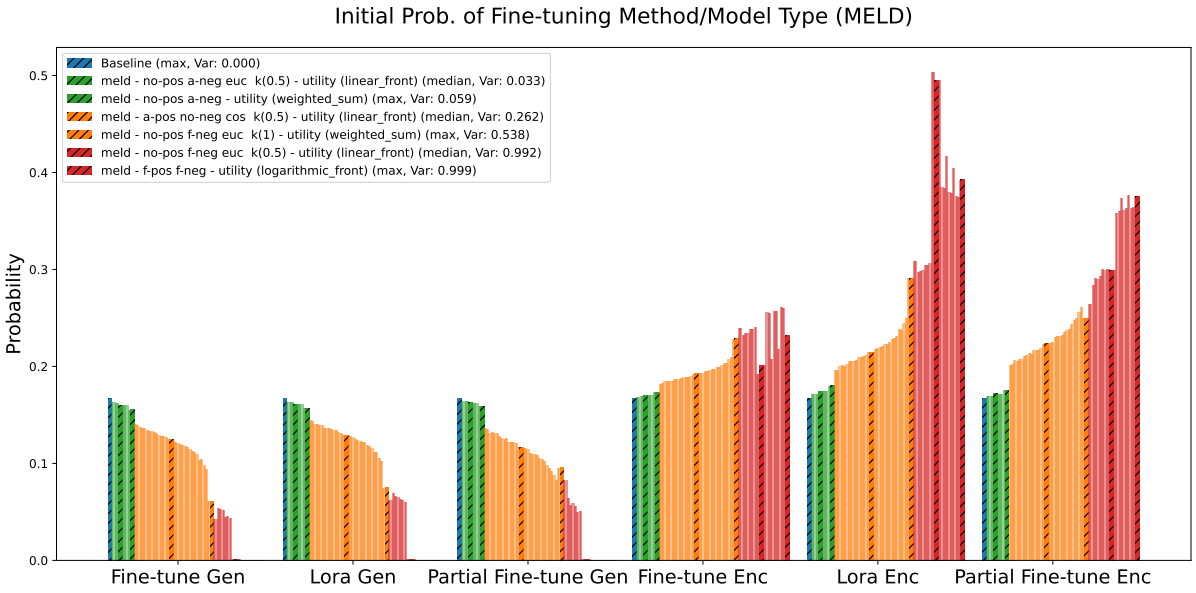

Figure 7: Initial probability distributions for fine-tuning methods on MELD

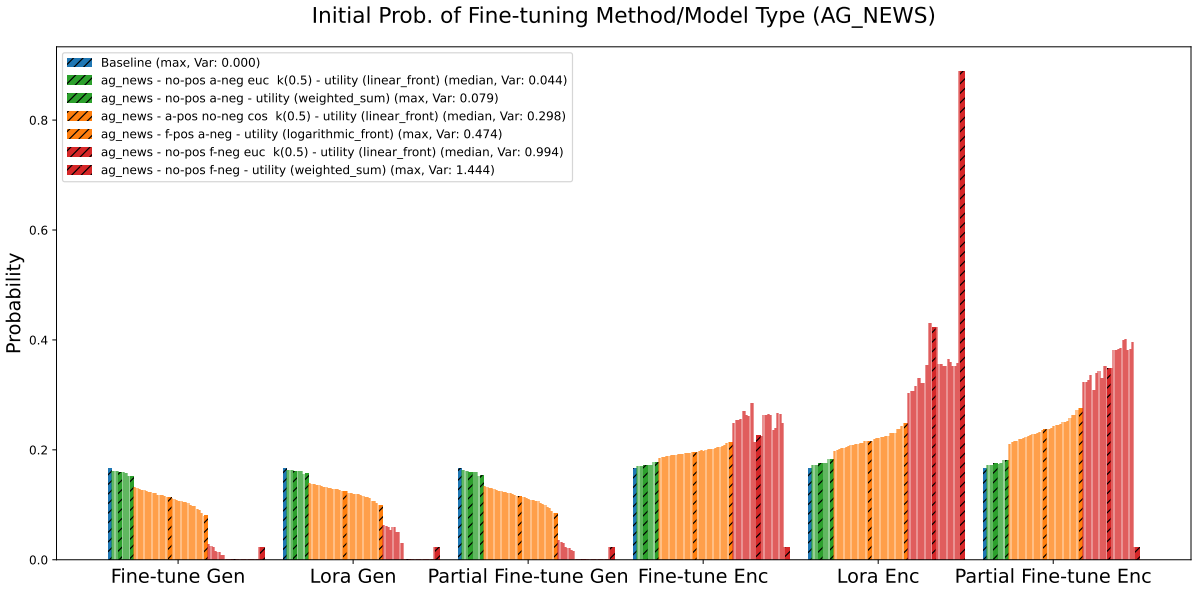

Figure 8: Initial probability distributions for fine-tuning methods on AG News

C Pareto Front Visualizations

Figure 9 illustrate the Pareto fronts discovered by XAUTOLLM on each dataset (LIAR, SST2, MELD, and AG NEWS). In each plot, **the blue line** represents the Pareto front of *baseline* solutions (i.e., the standard AutoML search without meta-learning). The coloured markers (triangles, diamonds, stars) represent pipelines discovered when applying our *warm-start* (WS) configurations. Each point is plotted in $(ET, F1_{\text{macro}})$ space, where:

- ET (*x-axis*) is the wall-clock time for a pipeline to be evaluated (lower is better).
- $F1_{\text{macro}}$ (*y-axis*) measures classification performance (higher is better).

Points that lie *to the left of* or *above* the baseline front dominate the baseline in at least one objective. In most cases, WS solutions (e.g., *High WS - Median*, *Mod WS - LIAR*) simultaneously improve upon the baseline's ET and $F1_{\text{macro}}$, indicating superior pipelines. Below, we discuss notable observations by dataset.

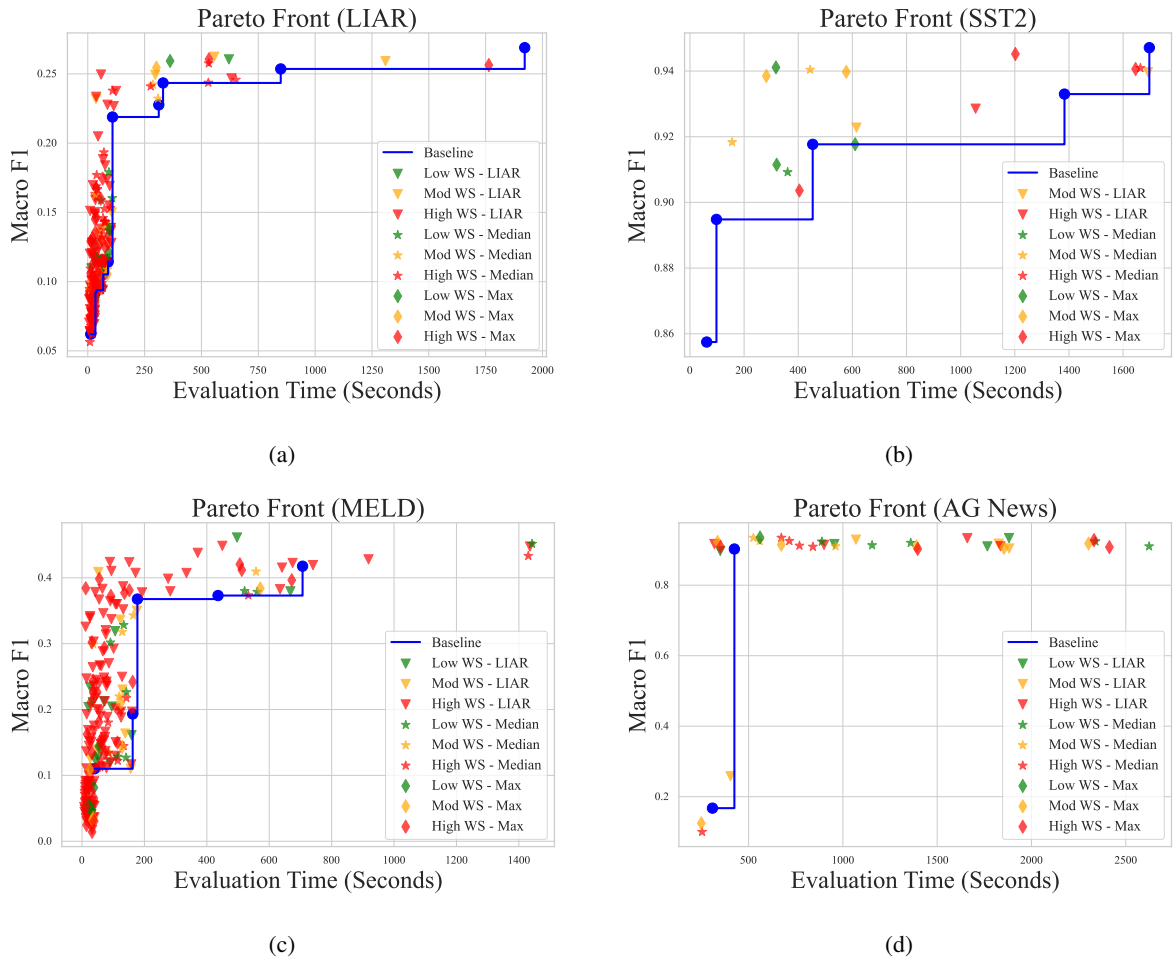

Figure 9: Pareto Fronts discovered by the different configurations of XAutoLLM on LIAR, SST2, MELD and AG News.

LIAR. Figure 9a shows how multiple WS configurations produce pipelines faster than or more accurate than any baseline point, effectively pushing the Pareto frontier upward and leftward. Consistent with the distance analysis (Figure 4 in the main paper), LIAR benefits enormously from relevant historical experience in its meta-learning pool. Evidently, HIGH WS - LIAR dominates the task, diminishing the error ratio by sevenfold and achieving around 40% winning ratio (Figure 2).

SST2. Figures 9b and 3 (also referenced in Section 5 of the main text) reveals a more *moderate* improvement, given fewer closely related prior tasks. Still, several WS pipelines *dominate* certain baseline solutions by achieving higher $F1_{\text{macro}}$ in less time. Notice that points above and to the left of the blue line reflect pipelines outperforming baseline results on both objectives.

MELD. Figure 9c demonstrates how MELD, like LIAR, sees *numerous* WS-discovered solutions outclassing the baseline. These configurations often exploit shared meta-features between MELD and LIAR (see Figure 4), culminating in faster convergence and higher accuracy, with fewer errors during the search. Mirroring LIAR, HIGH WS - LIAR dominates, diminishing the error ratio by sevenfold and almost getting 50% winning ratio (Figure 2).

AG News. Lastly, Figure 9d shows that while AG NEWS has only moderate overlap with other tasks, WS still yields solutions that meet or beat baseline performance in time-accuracy trade-offs. Notably, MOD and HIGH-bias configurations reduce error rates (see Table 8 in the main text), suggesting that historical knowledge, even if partially relevant, helps prune more obviously unproductive hyperparameter regions.

Overall, these Pareto front analyses confirm that XAUTOLLM leverages prior knowledge to reduce exploration overhead, often uncovering solutions that surpass a strong baseline in both $F1_{\text{macro}}$ and

evaluation time.