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

Ernesto L. Estevanell-Valladares[1,2], Suilan Estévez-Velarde[2], Yoan Gutiérrez[1], Andrés Montoyo[1], Ruslan Mitkov[3],
[1]University of Alicante, [2]University of Havana, [3]University of Lancaster,
ernesto.estevanell@ua.es

Abstract

Experts in machine learning leverage domain knowledge to navigate decisions in model selection, hyperparameter optimization, and resource allocation. This is particularly critical for fine-tuning language models (LMs), where repeated trials incur substantial computational overhead and environmental impact. However, no existing automated framework simultaneously tackles the entire model selection and hyperparameter optimization (HPO) task for resource-efficient LM fine-tuning. We introduce **XAutoLM**, a meta-learning-augmented AutoML framework that reuses past experiences to optimize discriminative and generative LM fine-tuning pipelines efficiently. XAutoLM learns from stored successes and failures by extracting task- and system-level meta-features to bias its sampling toward valuable configurations and away from costly dead ends. On four text classification and two question-answering benchmarks, XAutoLM surpasses zero-shot optimizer's peak $F1$ on five of six tasks, cuts mean evaluation time of pipelines by up to 4.5x, reduces search error ratios by up to sevenfold, and uncovers up to 50% more pipelines above the zero-shot Pareto front. In contrast, simpler memory-based baselines suffer negative transfer. We release XAutoLM and our experience store to catalyze resource-efficient, Green AI fine-tuning in the NLP community.

1 Introduction

Fine-tuning large language models (LLMs) has become indispensable across natural language processing (NLP) applications, yet even "small" models such as BERT (Devlin et al., 2018) or T5 (Raffel et al., 2020) incur substantial computational cost and carbon emissions (Wang et al., 2023b; Schwartz et al., 2020). Rather than exhaustively evaluating every model and hyperparameter combination, human experts draw on domain knowledge to focus on promising regions of this vast design space.

Automated Machine Learning (AutoML) seeks to mimic expert intuition by automating the two core stages of pipeline construction, model selection (MS) and hyperparameter optimization (HPO), into a unified search loop (Hutter et al., 2019). AutoML techniques have matured in areas such as tabular and vision tasks (Hutter et al., 2019), showing competitive performance against human experts (Estevez-Velarde et al., 2020). However, the joint MS+HPO pipeline for language models presents an ample, mixed discrete-continuous search space whose repeated evaluations are prohibitively costly (Wang et al., 2023b), thus posing a significant challenge for automation. While several recent efforts address HPO for LMs in isolation (Mallik et al., 2024), surveys highlight the underdevelopment of full-pipeline AutoML in NLP (Tornede et al., 2023), and no framework systematically unifies model selection and HPO under tight compute and Green AI constraints.

To address these shortcomings, we present **XAutoLM**, an AutoML framework that unifies model selection and hyperparameter optimization for LM fine-tuning via meta-learning. XAutoLM constructs an *experience-aware prior* from a repository of past pipeline evaluations annotated with task- and system-level meta-features which steers the search toward historically promising and away from infeasible configurations. Empirically, across four classification and two question-answering benchmarks, our method yields pipelines with stronger performance-time trade-offs than zero-shot or naive baselines under identical wall-clock budgets (Tables 5, 6). We release the code and the full experience store[1] to support sustainable, reproducible LM fine-tuning in the NLP community.

We summarize our main contributions as follows:

- A unified, meta-learning–augmented AutoML

[1]https://github.com/EEstevanell/XAutoLM

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 7863–7880
November 4-9, 2025

framework that integrates *both* model selection and hyperparameter optimisation for discriminative and generative LM fine-tuning.

- An extensible, task- and model-agnostic *experience-aware prior* that conditions the search on task *and* system meta-features and explicitly leverages negative traces to avoid costly dead ends.

- A comprehensive evaluation on six benchmarks showing consistent gains in F_1, mean pipeline evaluation time, and error ratio, and stronger Pareto fronts than zero-shot and naive memory baselines (see Section 4; Tables 5, 6).

We next review related work (Section 2), present XAutoLM (Section 3), and report the experimental setup and results (Section 4), followed by analysis (Section 5) and, finally, conclusions and limitations (Sections 6, 7).

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							
XAutoLM	✓		✓	✓	✓	✓	✓

Table 1: Comparison of systems for AutoML with LLMs

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 LMs. 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 optimization strategies to produce graph learning pipelines, highly capable multi-modal ML pipelines, and time-series forecasting pipelines, respectively.

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) optimizes 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 hyperparameter optimization for LM fine-tuning, particularly when resource limitations exist.

3 Proposal

We introduce **XAutoLM**, the first AutoML framework that unifies *model selection* and *hyperparameter optimisation* for both **discriminative** and **generative** language model fine-tuning. Our pipelines are composed of (i) a base LM from a curated pool of encoders and generators (Table 2), (ii) one of three fine-tuning strategies; full, partial, or LoRA (Hu et al., 2021), and (iii) a hyperparameter configuration. XAutoLM jointly explores this mixed search space by reusing past *experiences*, e.g., "LoRA-tuned DistilBERT achieved high macro-F1 on SST-2 under low VRAM", to steer the optimizer toward high-utility regions and away from error-prone configurations. This holistic reuse enables XAutoLM to discover strong fine-tuning pipelines under tight compute budgets.

Discriminative
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., 2024b)
New Additions
PHI-3.5 (Mini-Inst) (Abdin et al., 2024a)
PHI-4 (Mini-Inst, Reasoning) (Abdin et al., 2024a)
MIXTRAL (8x7B) (Mistral AI Team, 2023)
MISTRAL NEMO (Base-Inst) (Mistral AI Team, 2024)
Llama 3.1, 3.2 (1B - 70B) (Grattafiori et al., 2024)
DeepSeek R1[2] (DeepSeek-AI et al., 2025)

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

Background XAutoLM builds on AutoGOAL's[3] probabilistic optimizer (Estevez-Velarde et al., 2020). The optimizer represents every valid LM pipeline c as a point in a *mixed* search space that combines discrete choices (e.g. fine-tuning method, model, tokenizer) with continuous hyperparameters (e.g. learning rate, dropout). It maintains a probability distribution $P(c\,|\,\theta)$ over that space. It repeats a simple *sample–evaluate–update* loop: (1) *sample* a batch of pipelines from $P(c\,|\,\theta)$; (2) *evaluate* them on the target task; and (3) *update* $P(c\,|\,\theta)$ so that high-performing pipelines gain probability mass while under-performing and failures lose it. AutoGOAL always *initializes* this distribution **uniformly**, meaning every pipeline, adequate or not, is equally likely at the first generation.

3.1 Process Overview

XAutoLM replaces this uniform cold start with an *experience-aware prior* that follows a structured meta-learning process. Initially, the framework retrieves relevant historical evaluations (experiences) from a centralized repository (Section 3.2). Then, it computes detailed task and system meta-features (Section 3.2.1) to characterize the complexity and available resources for the present optimisation task. Leveraging this information, XAutoLM probabilistically adjusts the AutoML search space (Section 3.3), focusing on historically successful configurations and reducing exploration of previously unsuccessful paths. Once configured, the AutoML optimisation starts, fine-tuning pipelines are evaluated, and their outcomes, both successful and unsuccessful, are recorded back into the experience repository, to be used in future runs.

3.2 Experience Store

Our system learns from a growing repository of *experiences*; past pipeline evaluations that capture every factor influencing performance. Formally, an experience is a 4-tuple $e = \langle c,\ \mathbf{m},\ t,\ s\rangle$ where c is the complete pipeline configuration, $\mathbf{m}$ the vector of recorded metrics (e.g. F1, ROUGE, evaluation time), t a task meta-feature vector, and s straightforward system descriptors such as CPU cores, RAM, and GPU memory.

We label an experience **positive** if all fitness metrics are valid and **negative** otherwise, usually due to errors occurring during evaluation (out-of-memory, timeout, etc.). Both types are essential: positives pull the search toward valuable regions, and negatives push it away from costly dead-ends (Section 3.3).

3.2.1 Meta-Features

We design two complementary meta-feature templates according to the *nature of the output space* of a task. When the output is drawn from a **closed label set**, as in text classification or sequence labelling, dataset difficulty is dominated by class

[3]Open-source available at: https://github.com/autogoal/autogoal, licensed without restriction.

imbalance and document-length variation. Conversely, tasks whose output is an **open text sequence** (question answering, summarisation, translation) demand features that capture the relationship between the input prompt and the target text. Table 3 lists the core features for each template; the same templates can be reused for other label-based or free-form generation tasks with minimal adaptation.

Category/Feature
Dataset
Nr Samples
Nr Classes
Entropy
Min Cls Prob
Max Cls Prob
Imbalance Ratio
Documents
Avg. Length
Std. Length
Coef. Var. Length
Landmark
PCA + D.Tree Acc.

(a) **Label-based**

Category/Feature
Dataset
Nr Samples
Prompt
Avg.\Len (chars)
Std.\Len
Lexical Diversity (TTR)
Target
Avg.\Len (chars)
Std.\Len
Lexical Diversity (TTR)
Prompt–Target
Avg.\Len Ratio (T/P)
Vocabulary Novelty
Semantic Similarity
ROUGE-L F1
Semantic
Mean Prompt Embedding

(b) **Generation**

Table 3: Representative task meta-features.

Experiences record a minimal hardware profile in s (CPU cores, CPU frequency, total RAM, GPU VRAM) so similarity and feasibility reflect both task and system characteristics. For instance, while Llama 3.1 70B may yield superior results to smaller alternatives, systems with low VRAM cannot utilize its power.

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

3.3 Warm-Start optimization

XAutoLM maintains a probabilistic model $P(c \mid \theta)$ (Estevez-Velarde et al., 2020) over pipeline configurations c. When a new task T arrives, we retrieve a set of past *experiences* $\mathcal{E} = \{e_1, \ldots, e_n\}$ and update the model in two sweeps; one for positive experiences, one for negatives:

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

$$P(c \mid \theta) \leftarrow (1 + \alpha_i^-)\, P(c \mid \theta) - \alpha_i^-\, P_i(c \mid \theta) \quad (2)$$

where $P_i(c \mid \theta)$ is the empirical distribution induced by configuration c in experience e_i. Therefore *pull* the search toward successful regions and *push* it away from unsuccessful ones. The strength of each pull/push is governed by the *learning rates* α_i^+ and α_i^-.

We compute experience-specific learning rates considering their similarity to the current task and historical performance. Specifically, these rates are computed as follows:

$$\alpha_i^+ = \alpha_{\max}^+\, u_i\, e^{-\beta\, d_i}, \quad (3)$$

$$\alpha_i^- = \alpha_{\max}^-\, e^{-\beta\, d_i}. \quad (4)$$

Here $\alpha_{\max}^+$ and $\alpha_{\max}^-$ are predefined maximum learning rates, $u_i \in [0, 1]$ is a utility score (defined below) assigned *only* to positive experiences, and d_i is the distance between the current task and the one that generated experience e_i. The exponential kernel $e^{-\beta d_i}$ down-weights experiences that are less similar to the current task; $\beta > 0$ is an adaptive decay factor.

Task Similarity. Each task is described by a meta-feature vector t. Similarity is measured with a distance $d_i = \mathrm{Dist}(t_T, t_i)$ (e.g., Euclidean or Cosine). β is set automatically to compensate for scale:

$$\beta = \frac{\beta_{\text{scale}}}{\sigma_d + \varepsilon}, \quad \sigma_d = \mathrm{Std}\big(\{d_1, \ldots, d_n\}\big), \quad (5)$$

where $\varepsilon > 0$ prevents division by zero.

Utility Score. The utility function u_i quantifies the quality of each positive experience e_i relative to others from the same task. XAutoLM supports three distinct utility computation strategies: (*i*) Weighted Sum, (*ii*) Linear Front, and (*iii*) Logarithmic Front:

Weighted Sum. Let $\mathcal{M}$ denote the set of recorded performance metrics for each experience, such as F1, accuracy, evaluation time, or ROUGE-L. Each metric $m \in \mathcal{M}$ is associated with a known optimisation direction (maximize or minimize) and an importance weight w_m. For each positive experience e_i, we first normalize its metric value m_i:

$$m_i' = \begin{cases} \dfrac{m_i - m_{\min}}{m_{\max} - m_{\min}}, & \text{if maximized,} \\ 1 - \dfrac{m_i - m_{\min}}{m_{\max} - m_{\min}}, & \text{if minimized,} \end{cases} \quad (6)$$

where $m_{\min}$ and $m_{\max}$ denote the minimum and maximum values observed across all positive experiences for the metric m. If all metric values are identical, we default to a neutral utility score of 0.5 to avoid division by zero. The overall weighted utility score is computed as:

$$u_i = \frac{\sum_{m \in \mathcal{M}} w_m \cdot m'_i}{\sum_{m \in \mathcal{M}} w_m}, \tag{7}$$

Linear Front. In the Linear Front utility scheme, we first apply non-dominated sorting (NSGA-II style (Deb et al., 2002)) to all positive experiences, creating N Pareto fronts based on the recorded metrics in $\mathcal{M}$. Experiences in front 0 are non-dominated, followed by those in front 1, and so forth. Each positive experience e_i in front f_i is assigned a utility score inversely proportional to its front rank:

$$u_i = \frac{N - f_i}{N}, \tag{8}$$

Logarithmic Front. Using non-dominated sorting, the Logarithmic Front approach similarly ranks experiences into N Pareto fronts. However, to amplify the distinction among the highest-performing experiences (i.e., those in lower-numbered fronts), utilities decrease logarithmically with rank:

$$u_i = \frac{\ln(N - f_i + 1)}{\ln(N + 1)}, \tag{9}$$

These three utility functions provide complementary strategies for prioritizing past experiences. This flexibility allows XAutoLM to adapt effectively across diverse AutoML scenarios.

4 Experimentation

We report results from two *independent* transfer experiments designed to isolate knowledge reuse *within* a task family. The first study targets **text classification**. LIAR (Wang, 2017), SST-2 (Socher et al., 2013), MELD (Poria et al., 2018) and AG News (Zhang et al., 2015) present a deliberate gradient in sample size, label entropy, and average document length: LIAR (6 classes, 13k claims) and MELD (7 emotions, 14k utterances) are notoriously low-resource, whereas the polarity benchmark SST-2 (68k) and the large-scale news corpus AG (128k) approach the upper bound of single-GPU throughput. Previous work shows peak $F1_{\text{macro}}$ to vary from 0.23 (LIAR) to 0.93 (AG) (Reusens et al., 2024), offering a realistic range for efficiency–performance trade-offs.

The second experiment focuses on **question answering**. We select SQuAD 1.1 (Rajpurkar et al., 2016) and DROP (Dua et al., 2019) because they share the same input modality yet differ sharply in answer type, extractive spans versus multi-step numerical reasoning, making them a challenging test-bed for generative pipelines. For both studies, experiences are only exchanged among tasks of the same family; classification traces are invisible to QA runs and vice-versa. This constraint ensures that the reported gains stem from *task-relevant* meta-knowledge rather than accidental data leakage.

Hardware. All classification experiments run on an `i9-9900K` (16 threads, 35 GB RAM cap) paired with a single RTX TITAN (24 GB). QA experiments require larger context windows and execute on an `AMD EPYC 7742` (64 threads, identical RAM cap) with an A100 40 GB.

Baselines. Every run is compared against **Zero-Shot AutoGOAL**, the original optimizer with a uniform sampling distribution; in this setting, the update rules of equations (1)–(9) are never triggered.

In the text classification study, we include a naive **kNN-50** memory baseline for comparing against a naive experience retrieval method. For every target task, we assemble a query vector that concatenates (a) the task meta-features, (b) the current system profile, and (c) the best metric values observed across all stored traces; this encourages the search to drift toward high-performing regions. Distances to positive traces are computed on the full feature+metric space, whereas distances to *negative* traces ignore metrics (errors lack valid scores). The k nearest positives and k nearest negatives are selected; all receive the same fixed learning rate $\alpha_i^{\pm} = 1/k$. Setting $u_i = 1$ and $\beta = 0$ in equations (3)–(4) reduces our framework to this simple neighbour rule. For question answering the repository contains only between 5 and 10 positive traces per source task, making a neighbour count unreliable; therefore Zero-Shot remains the sole baseline in that study.

Warm-Start Priors. Throughout the paper, a *pipeline configuration* is a concrete tuple (LM, fine-tuning recipe, hyperparameters) that the AutoML engine executes and evaluates. A *warm-start prior* (WS prior) instead parameterizes the initial sampling distributions used by the meta-learner; it is defined by the distance type, utility scheme, decay factor β_{scale}, and pull limits $(k_{\text{pos}}, k_{\text{neg}})$.

For each task, we enumerate ≈ 180 WS-prior parameterizations. For a given candidate prior to a task, we apply it with the fixed experience store (leaving the experience for the current task out) to obtain the *induced* sampling distribution p over fine-tuning methods on that task. We then compute the total-variation (TV) distance between this induced marginal and the uniform distribution over the same method set. We rank candidates by TV and split them into three data-driven strata (*low* | *moderate* | *high* bias) at prominent TV gaps ($\approx 2\times$). In classification, we select per strata the median-TV and max-TV priors (six priors total). In QA, we select only the max-TV prior per strata (three priors) to respect the compute budget. Full probability plots of the induced method distributions and the selected prior identifiers are provided in Appendix B.

Execution protocol. For each task, we first ran the Zero-shot configuration for 48 hours to populate the experience store. Table 4 reports the positive/negative traces generated by this baseline run on each task. We then executed the kNN-50 baseline and all WS-prior variants for **24 hours of wall-clock time** each. The warm-start mechanism accesses only experiences originating from other tasks within the same study (clean cross-task transfer; see Table 4). For fairness in reporting, Zero-shot metrics are computed from the **first 24 hours** of their 48 hours runs, matching the wall-time allocated to WS-priors and kNN-50. This protocol isolates whether experience improves both effectiveness and efficiency under the same time budget.

In every AutoML run, each discovered LM pipeline has up to 1.5 GPU-hours in Text Classification and 2 GPU-hours in QA for evaluation. Objectives are $\langle F1_{\text{macro}}, ET\rangle$ for classification and $\langle F1, ET\rangle$ for QA, where ET is the wall-clock evaluation time of a pipeline (in seconds). All searches share a fixed random seed (42) and the same hardware; therefore, differences arise solely from the chosen warm-start prior.

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
SQUAD	5	124	129	10	160	170
DROP	10	160	170	5	124	129

Table 4: Disposition of experiences participating in the experiments.

4.1 Text Classification Results

Table 5 summarizes the effect of WS-priors on the four classification benchmarks. We report both performance and efficiency: max and mean $F1_{\text{macro}}$ reflect peak and average classification quality; mean evaluation time (ET) captures resource cost; the error ratio indicates the share of failed pipeline evaluations; and hypervolume (HV) measures Pareto-front coverage in objective space (Zitzler and Thiele, 1998). Mean ET is averaged over successfully completed pipeline evaluations only (i.e., runs that return valid fitness metrics); failed evaluations (e.g., out-of-memory, timeouts, runtime errors) are excluded from ET and are accounted for by the error ratio. All methods are run under the **same 24 hours single-GPU budget** (cf. Execution protocol), so ET differences reflect pipeline runtime rather than total search compute.

Across datasets, WS priors either *match* or *surpass* the best Zero-shot $F1_{\text{m}}$ while systematically improving efficiency. On LIAR, a HIGH prior lifts peak $F1_{\text{m}}$ from 0.24 to 0.26, cuts the mean ET by a factor of 3.5, and lowers the error ratio by sevenfold. A similar pattern emerges on MELD, where HIGH drives the error ratio from 0.77 to 0.10 and reduces mean ET 4.5×, while keeping $F1_{\text{m}}$ above the baseline. On SST-2, the Zero-shot baseline generated the highest $F1_{\text{m}}$ and lowest ET out of all variants.

Zero-shot runs exhibit high error ratios across all benchmarks (e.g., 0.73-0.92); the WS priors cut these failure rates dramatically, down to 0.09-0.90. Moreover, non-naive warm-started runs showed a sensible reduction in mean ET while maintaining peak $F1_{\text{m}}$. On AG News, all WS runs improve max $F1_{\text{m}}$ while several improve ET, HV and Error Ratio, showing that better performance–time trade-offs are discoverable even in large-scale settings.

The naive **kNN-50** baseline, although in SST-2 case attains large HV values, degrades performance on three datasets and notably obtains the worst

	WS Prior	Max $F1_m$	Mean $F1_m$	Min *ET*	Mean *ET*	HV	No. Eval	Error Ratio
LIAR	Zero-shot	0.24	**0.10**	12	537	0.06	202	0.73
	kNN (50)	0.24	**0.10**	28	451	0.11	**240**	0.44
	Low (LIAR)	**0.26**	**0.10**	16	480	0.10	197	0.70
	Low (Med)	0.25	0.09	31	380	**0.36**	220	0.69
	Low (Max)	0.25	0.09	21	410	0.08	190	0.66
	Mod (LIAR)	**0.26**	**0.10**	36	462	0.01	132	0.53
	Mod (Med)	0.24	**0.10**	13	469	0.04	146	0.61
	Mod (Max)	0.25	0.08	44	516	0.05	121	0.39
	High (LIAR)	0.25	**0.10**	**6**	**153**	0.20	302	**0.09**
	High (Med)	0.25	**0.10**	9	277	0.12	193	0.33
	High (Max)	**0.26**	0.09	12	252	0.09	208	0.25
SST2	Zero-shot	**0.94**	**0.69**	**97**	1297	0.02	76	0.77
	kNN (50)	0.93	0.59	326	1758	**0.54**	72	0.62
	Low (LIAR)	0.90	0.48	373	1148	0.15	87	0.82
	Low (Med)	0.90	0.52	227	**840**	0.02	62	0.83
	Low (Max)	**0.94**	0.58	252	**784**	0.01	98	0.81
	Mod (LIAR)	0.93	0.56	245	996	0.20	59	0.64
	Mod (Med)	**0.94**	0.52	132	1030	0.04	34	0.55
	Mod (Max)	0.93	0.52	184	1170	0.06	58	**0.51**
	High (LIAR)	0.92	0.62	365	1160	0.02	42	0.61
	High (Med)	**0.94**	0.53	164	844	0.09	52	0.68
	High (Max)	**0.94**	0.61	320	857	0.16	53	0.79
MELD	Zero-shot	0.41	**0.15**	39	808	0.11	161	0.77
	kNN (50)	0.37	0.11	52	768	0.00	59	0.54
	Low (LIAR)	**0.46**	0.14	20	532	0.06	150	0.64
	Low (Med)	0.45	0.11	17	387	0.30	229	0.64
	Low (Max)	0.39	0.09	30	477	**0.36**	186	0.65
	Mod (LIAR)	0.40	0.11	26	514	0.00	106	0.39
	Mod (Med)	0.40	0.11	36	546	0.03	130	0.52
	Mod (Max)	0.38	0.09	24	590	0.08	110	0.52
	High (LIAR)	0.44	0.14	**7**	**179**	0.09	**260**	**0.10**
	High (Med)	0.43	0.13	21	466	0.27	124	0.45
	High (Max)	0.42	0.12	12	322	0.01	233	0.51
AG NEWS	Zero-shot	0.90	0.62	424	1043	0.00	**108**	0.92
	kNN (50)	0.67	0.28	478	1881	0.09	22	0.77
	Low (LIAR)	**0.93**	**0.73**	349	1183	0.01	93	0.90
	Low (Med)	0.92	0.65	665	1589	**0.20**	83	0.89
	Low (Max)	**0.93**	0.60	560	1164	0.00	77	0.90
	Mod (LIAR)	0.92	0.46	404	1345	0.12	50	0.80
	Mod (Med)	**0.93**	0.59	484	1102	0.01	48	0.79
	Mod (Max)	0.92	0.56	**249**	1402	0.01	57	0.73
	High (LIAR)	**0.93**	0.46	318	1437	0.00	45	**0.71**
	High (Med)	**0.93**	0.51	253	**833**	0.09	58	0.86
	High (Max)	0.92	0.54	350	1576	0.01	46	0.73

Table 5: Results overview in text classification. Priors with "(LIAR)" suffix were calibrated during a single-objective pilot on LIAR. The same meta-parameters are then applied unchanged to every new target task. Full probability curves and all prior IDs are listed in Appendices B-C.

results out of all priors in AG NEWS (0.90 → 0.67 $F1_m$) and MELD (0.41 → 0.37 $F1_m$).

4.2 Question Answering Results

Table 6 reports results on the generative SQuAD 1.1 and DROP datasets. Knowledge reused from a single related task already yields substantial gains. For SQuAD, WS priors outperform the baseline in almost all metrics. The HIGH-MAX prior, in particular, raises $F1$ from 0.34 to 0.89 while shrinking

	WS Prior	Max $F1$	Mean $F1_m$	Min *ET*	Mean *ET*	HV	No. Eval	Error Ratio
SQUAD	Zero-shot	0.34	0.23	2189	4081	**0.25**	**71**	0.95
	Low (Max)	**0.89**	0.33	1435	3150	0.03	30	**0.76**
	Mod (Max)	0.86	0.41	1468	1953	0.01	32	0.90
	High (Max)	**0.89**	**0.87**	**1195**	**1337**	0.0	15	0.8
DROP	Zero-shot	0.39	0.18	2114	3556	0.11	**96**	0.94
	Low (Max)	0.18	0.11	4995	5929	0.05	32	0.90
	Mod (Max)	**0.40**	0.23	**775**	2259	**0.29**	66	0.86
	High (Max)	**0.40**	**0.28**	783	**1881**	0.13	34	**0.82**

Table 6: Results overview in Question Answering.

mean ET from 4081s to 1337s (-3×). Similarly to the text classification results, WS priors bring error ratios down from 0.94–0.95 (zero-shot) to 0.76–0.90.

On DROP, the LOW prior illustrates negative transfer, yet both MODERATE and HIGH priors outperform Zero-shot on *every* metric; peak $F1$ improves slightly (0.39→0.40) and mean ET falls by 47%. These outcomes confirm that cross-task meta-knowledge generalizes beyond classification and that the adaptive pull/push schedule mitigates catastrophic transfers.

5 Discussion

Warm-start priors consistently steer the search toward stronger performance–time trade-offs across all six benchmarks. Figure 1 reports the winning ratio: the share of evaluated LM pipelines that improve upon the zero-shot Pareto front.

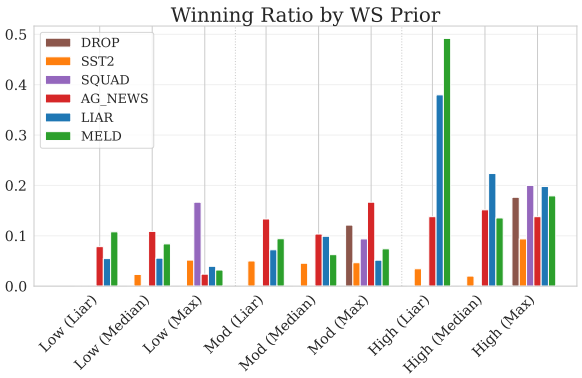
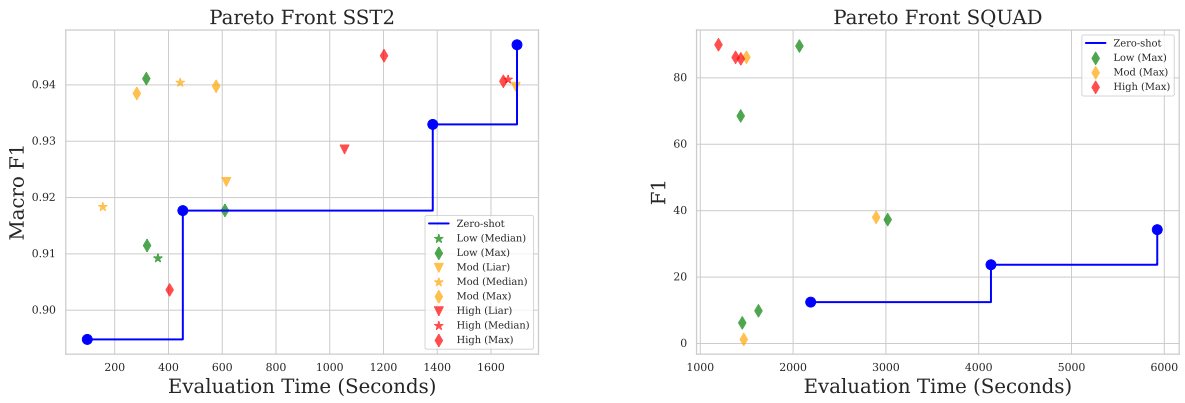


Figure 1: Ratio of discovered pipelines outperforming the Zero-shot baseline in Text Classification and QA.

The HIGH–MAX prior is the most stable, winning about 20% of pipelines on SQuAD, LIAR, MELD, and DROP, and 10–15% on SST-2 and AG News. On the LIAR and MELD pair, the HIGH–LIAR prior achieves winning ratios near 50% and 40%, respectively, while cutting the error rate by a factor of seven (Table 5). For clarity, all



(a) (b)

Figure 2: Pareto Fronts discovered by the different Priors on SST2 (a) and SQUAD (b).

ET values are computed only on successful evaluations, while failure rates are captured by the Error Ratio, with all methods allotted an identical 24 GPU-hour wall-clock budget per run.

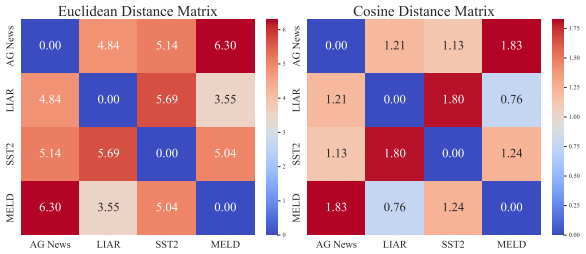


Figure 3: Distance between Text Classification Tasks according to their meta-features (Section 3.2.1).

These results show that combining experience discrimination with adaptive probability shifts yields the best of both worlds: rapid convergence when relevant meta-knowledge exists yet robustness when it does not. Whenever the experience store contained closely related traces, e.g., MELD–LIAR (Figure 3), the similarity-aware priors trimmed average evaluation time by up to 4.5x and increased peak $F1_m$ (Table 5). Even on sparsely related tasks such as SST-2 and AG News, softer pulls uncovered superior Pareto trade-offs by moderating exploration strength (Figure 2a).

The baseline performance of kNN highlights the significance of selective memory. While it has access to both positive and negative examples, it assigns equal weight to all neighbors, failing to demote weak configurations and causing accuracy to fall on three of four classification datasets. In contrast, XAutoLM's asymmetric pull–push update penalizes both past failures and underperforming successes. DROP, for example, illustrates the need to learn from failures: a low-bias prior that ignores negatives collapses to $F_1 = 0.18$, whereas reinstating the push restores $F_1 = 0.40$ and halves mean evaluation time.

Our findings further show that transfer using our method extends beyond classification. With barely a handful of relevant experience, a high-bias prior multiplies SQuAD $F1$ from ≈ 0.3 to ≈ 0.9 and compresses evaluation time by threefold, producing a dominant Pareto front (Figure 2b). On the other hand, DROP illustrates the importance of negative experiences: a low-bias prior that ignores negatives collapses to $F_1 = 0.18$, whereas reinstating the push restores $F_1 = 0.40$ and cuts mean evaluation time by 50 % (Table 6).

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, **XAutoLM** 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; Schwartz et al., 2020), emphasizing the importance of responsible resource usage. Our experiments demonstrate that our warm-start strategy enhances performance and streamlines the search process, resulting in algorithms that strike a better balance between efficiency and performance.

6 Conclusions

XAutoLM converts the costly trial–and–error of language model fine-tuning into a guided, resource-

aware search. By seeding the optimizer with a similarity-weighted prior built from past *successes & failures*, the framework consistently uncovers pipelines with superior performance–time trade-offs. Across four text-classification corpora and two generative QA benchmarks, it surpasses the best zero-shot F_1 on five tasks, matching it on SST-2, while cutting mean pipeline evaluation time by *up to* a factor of four and reducing error rates by *as much as* sevenfold. These gains hold across a refreshed model pool that ranges from lightweight discriminative to compact generative models. Because every recovered pipeline reuses information already paid for, XAutoLM advances the *Green AI* agenda (Schwartz et al., 2020), delivering competitive results in less search time, while avoiding redundant computation.

7 Limitations

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

Scaling to bigger LLMs

XAutoLM is *scale-agnostic*: the optimizer treats candidates as black-box fit/evaluate calls and does not rely on model internals. Our open-source implementation presently evaluates on a **single GPU**, which constrained the largest models tested; this is a property of the evaluator backend, not of the optimization method. The experience store logs a minimal hardware profile (Section 3.2), which helps steer the search away from infeasible pipelines under a given machine with a single GPU setup. Supporting larger models, therefore, amounts to adding multi-GPU meta-features and swapping in a larger-model evaluator (e.g., parameter-efficient (Hu et al., 2021)/quantized (Nagel et al., 2021; Dettmers et al., 2023) or distributed evaluators (Zhao et al., 2023)) in future releases; the search algorithm and experience-based priors remain unchanged. We leave such engineering backends to future work and keep our claims limited to the single-GPU setting evaluated here.

Multimodality

The current experience store and benchmarks are text-only; verifying that the warm-start prior transfers to dialogue, speech, or multimodal pipelines is an essential next step.

Statistical Tests

Statistical support is available only for the single-objective probes archived in Appendix C. Extending significance testing to the multi-objective fronts of Tables 5 and 6 would require many repeated runs and is left for future work, where bootstrap or fully Bayesian analyses are planned.

Efficiency Measures

Our energy discussion rests on the empirical link between execution time and power draw reported by prior work (Wang et al., 2023b; Estevanell-Valladares et al., 2024); we did not log wattage directly. The next release of XAutoLM will record real-time power and emit CO_2 estimates alongside performance metrics.

Acknowledgments

This research has been partially funded by the University of Alicante, the University of Havana, the Spanish Ministry of Science and Innovation, the Generalitat Valenciana, and the European Regional Development Fund (ERDF) through the following funding: At the regional level, and as the primary source of support, the Generalitat Valenciana (Conselleria d'Educacio, Investigacio, Cultura i Esport), FEDER granted funding for CIDEGENT (CIDEXG/2023/13); and NL4DISMIS (CIPROM/2021/21). At the national level, the following projects were granted: HEART-NLP (PID2024-156263OB-C22); COOLANG (PID2021-122263OB-C22); SOCIALTRUST (PDC2022-133146-C22); ILENIA (2022/TL22/00215334) and ALIA models (`https://alia.gob.es`) funded by MCIN/AEI/10.13039/501100011033 and, as appropriate, by ERDF A way of making Europe, by the European Union or by the European Union NextGenerationEU/PRTR; and by the State Subprogram for Training, Attraction, and Retention of Talent (PEICTI 2024) of the Spanish Ministry of Science and Innovation, grant PRX24/00272.

References

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

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, et al. 2024b. Phi-3 technical report: A highly capable language model locally on your phone. *Preprint*, arXiv:2404.14219.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

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.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115.

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*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*.

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.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

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.

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.

Mistral AI Team. 2023. Mixtral of experts. `https://mistral.ai/news/mixtral-of-experts`. Accessed: 2025-05-17.

Mistral AI Team. 2024. Mistral NeMo: our new best small model. https://mistral.ai/news/mistral-nemo. Accessed: 2025-05-17.

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

Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart Van Baalen, and Tijmen Blankevoort. 2021. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*.

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.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

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.

Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. 2020. Green ai. *Communications of the ACM*, 63(12):54–63.

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*.

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.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. 2023. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*.

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*.

Eckart Zitzler and Lothar Thiele. 1998. Multiobjective optimization using evolutionary algorithms—a comparative case study. In *International conference on parallel problem solving from nature*, pages 292–301. Springer.

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 multi-objective and single-objective experiments are available in Appendix B and Appendix C 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 (Appendix C) 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 (Appendix C for single-objective, and Appendix B for multi-objective).

A.4 Experimental Setup and Computational Resources

The main text fully discloses our experimental setup (Section 4).

A.5 Framework Overview and Dependencies

XAutoLM 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 GitHub repository: `https://github.com/EEstevanell/XAutoLM`.

B Multi-Objective Initial Probabilities

This appendix visualizes the initial probability distributions over fine-tuning methods induced by different meta-learning configurations (Prior) in our multi-objective experiments (see Section4). Each configuration is defined by:

1. Inclusion of positive and/or negative experiences,

2. Utility function (Weighted Sum, Linear Front, Logarithmic Front),

3. Distance metric (Euclidean, Cosine) with scaling, and

4. Pull/push limits k_{pos}, k_{neg} and learning-rate scheme (fixed/adaptive).

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. α^{+}_{max} and α^{-}_{max} values (fixed or adaptive) (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 categorize them into *low*, *moderate*, and *high* bias levels relative to the baseline's uniform initialisation.

B.1 Classification Tasks

For each classification dataset (LIAR, SST-2, MELD, AG News), Figures 4–7 plot the initial probabilities for representative configurations at each bias level. In each figure:

- **Blue:** Uniform baseline.
- **Green, Orange, Red:** Increasing TV distance (Low, Moderate, High).
- **Patterned Bars:** Selected *Max-TV* configuration within each bin.

LIAR. Figure 4 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 5 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 6 shows the MELD dataset's initial distributions. As discussed in Section 4, MELD shares some meta-feature similarities with LIAR (Figure 3), causing some distributions to concentrate around methods found promising in LIAR's prior runs.

AG News. Lastly, Figure 7 displays the candidate configurations for AG NEWS, a large corpus with four news categories.

B.2 QA Tasks

Figures 8a and 8b show the analogous distributions for DROP and SQuAD. Despite fewer experiences, meta-learning concentrates probability mass on the partial and traditional fine-tuning strategy while avoiding Lora.

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

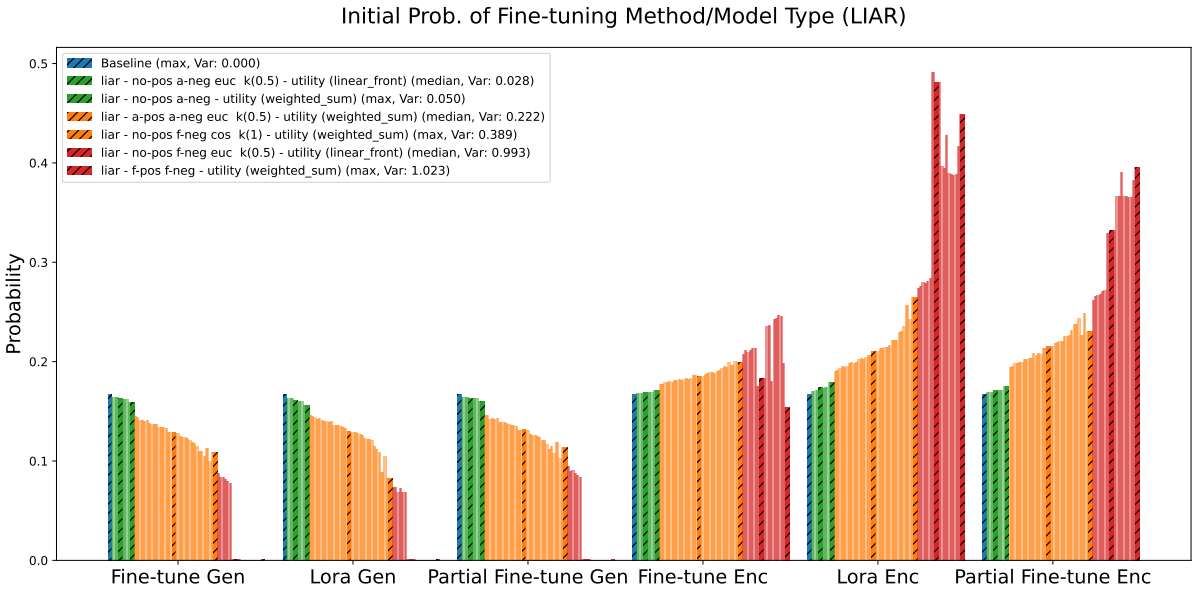


Figure 4: Initial probability distributions for fine-tuning methods on LIAR.

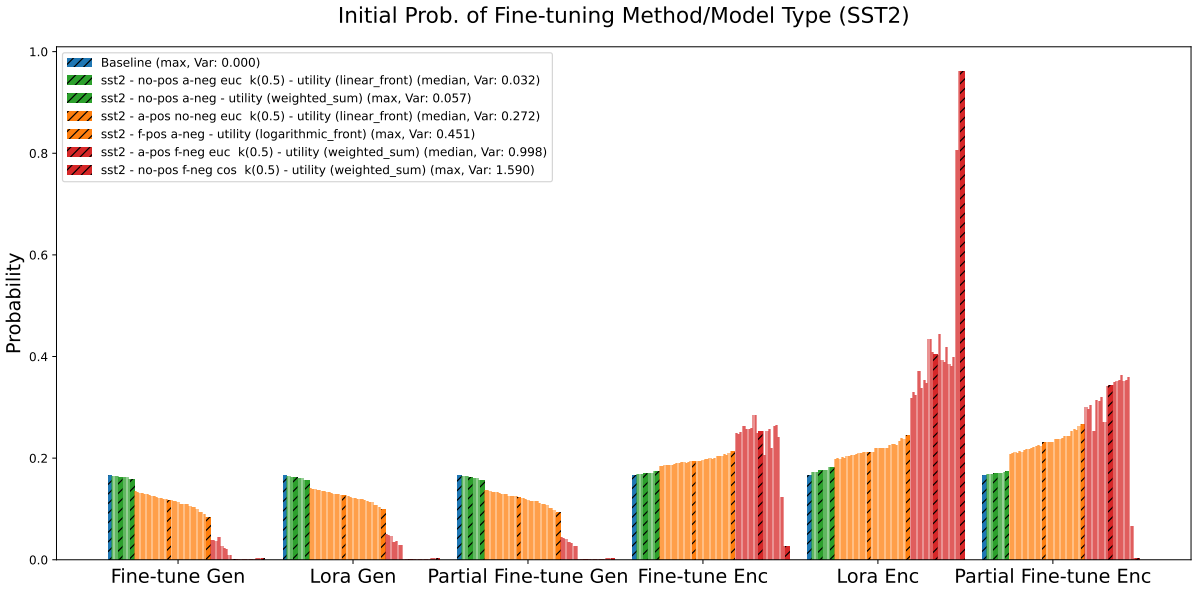


Figure 5: Initial probability distributions for fine-tuning methods on SST2

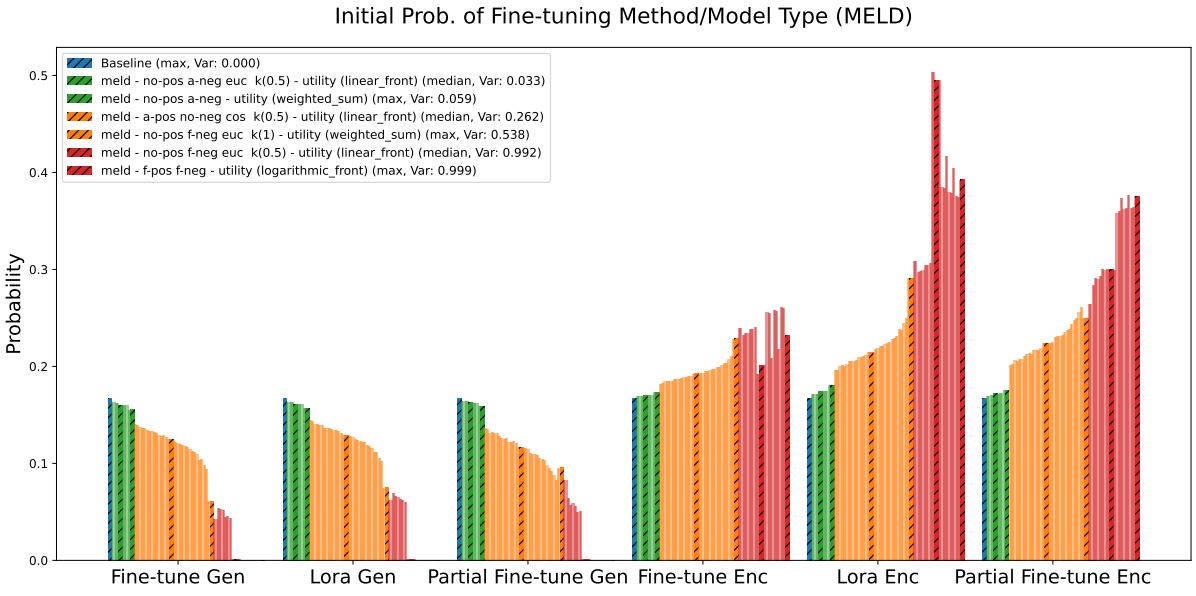


Figure 6: Initial probability distributions for fine-tuning methods on MELD

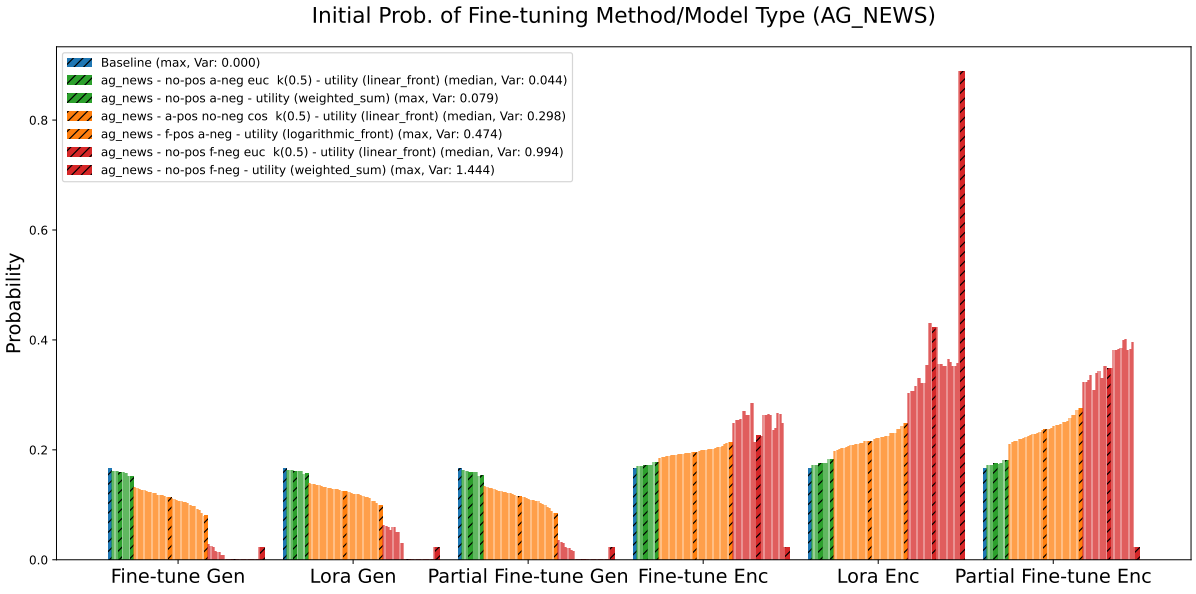


Figure 7: Initial probability distributions for fine-tuning methods on AG News

C Single-Objective Warm Start Evaluation

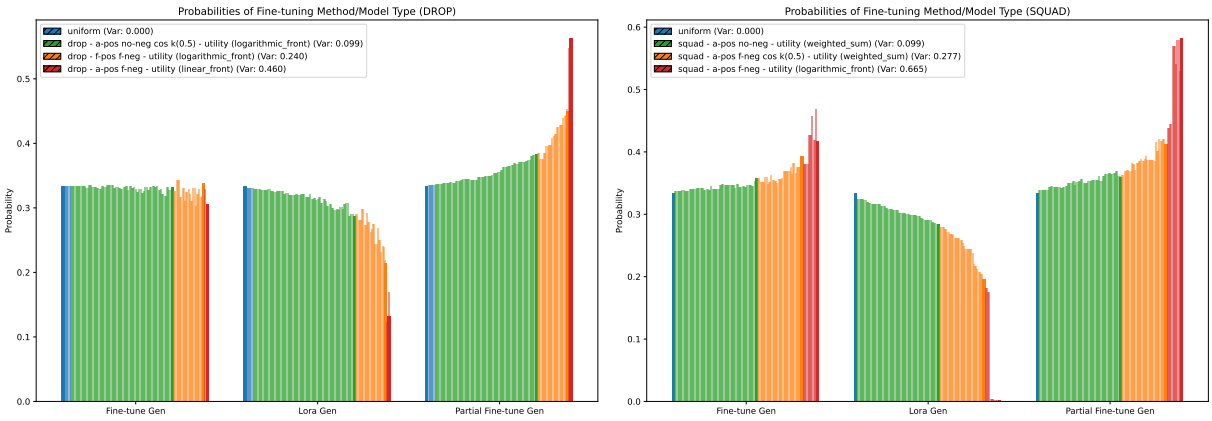
This appendix reports single-objective experiments optimizing the macro-$F1$ score alone. We compare the Zero-shot AutoGOAL baseline against three representative warm-start priors, Low, Moderate, and High bias, selected from fourteen candidate configurations grouped by total variation (TV) distance. All priors use the std-only β scale, Euclidean distance, and fixed learning rates ($\alpha^{+}_{\max} = 0.05$, $\alpha^{-}_{\max} = 0.02$).

C.1 Initial Probability Distributions

Figure 9 shows LIAR's initial fine-tuning method distributions under the fourteen meta-learning priors, sorted by TV relative to the uniform baseline. The solid blue bar indicates the baseline; patterned green, orange, and red bars mark the chosen Low, Moderate, and High priors.

C.2 Performance Results

Table 7 reports our results. We conducted a detailed statistical analysis across six independent runs per



(a) (b)

Figure 8: Initial probability distributions for fine-tuning methods on DROP (a) and SQUAD (b)

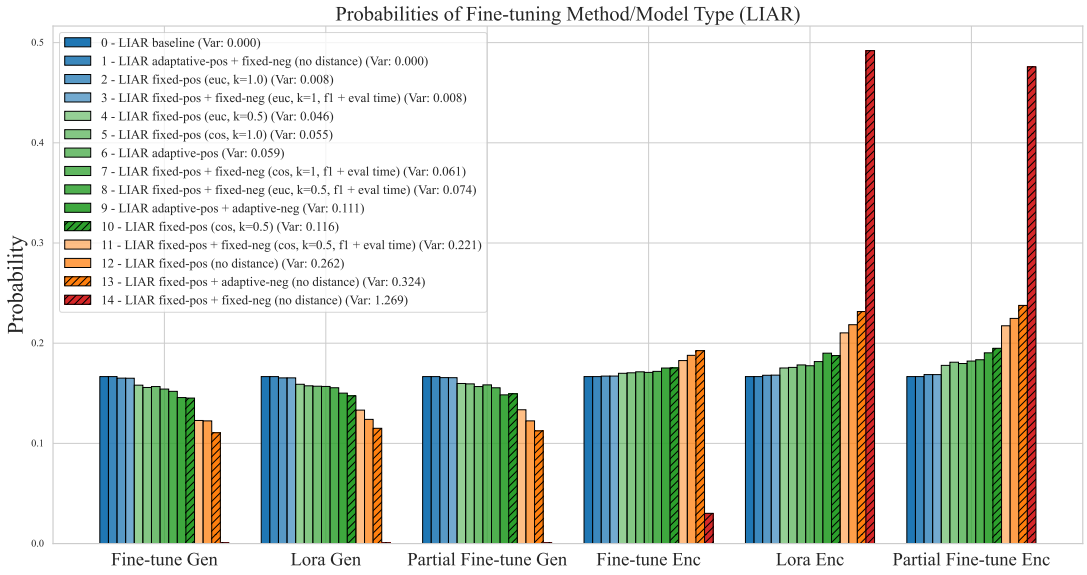


Figure 9: Initial fine-tuning probabilities for LIAR under fourteen priors, sorted by TV. Solid blue denotes the uniform baseline; patterned green, orange, and red denote the Low, Moderate, and High bias priors, respectively

configuration on LIAR and SST-2, evaluating performance, convergence time, and reliability. Normality was tested using Shapiro–Wilk, followed by ANOVA (McHugh, 2011) for normal metrics, and Friedman tests (Pereira et al., 2015) for non-parametric ones. We report Cohen's d and Cliff's δ as effect-size measures; power analyses accompany each test in the repository.

On **LIAR**, while none of the warm-start priors significantly outperformed the baseline in peak $F1_{\text{macro}}$ (ANOVA $p = 0.856$, Friedman $p = 0.94$), we observed a significant overall improvement in *mean* performance across groups (ANOVA $p = 0.005$, Friedman $p = 0.004$). Post-hoc comparisons, however, were not significant after correction, likely due to limited sample size. More notably, the *error ratio*, the share of failed evaluations, dropped dramatically from 0.69 (baseline) to 0.24 (High WS), a difference found to be statistically significant (Friedman $p = 0.031$) with a large effect size (Cohen's $d = 3.39$). Convergence time metrics (TT50, TT75, TT90) also trended lower, with moderate effect sizes, although these differences did not reach statistical significance.

On **SST-2**, the Mod WS prior achieved the highest max $F1_{\text{macro}}$ (0.941), and the ANOVA test confirmed a significant group effect ($p = 0.031$). The error ratio again showed a significant overall effect (Friedman $p = 0.038$), improving from 0.83 (baseline) to 0.58 (High WS). Convergence time reductions were most pronounced with the High WS prior, which reached 50% of peak $F1$ four times

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 XAutoLM 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.

faster than the baseline (0.41h vs. 1.69h). While these improvements showed large effect sizes (e.g., TT50 $d = 0.55$), they were not statistically significant in pairwise tests, most likely due to low sample power ($n = 6$).

In summary, warm-start priors consistently yielded practical convergence speed and robustness benefits. While not all improvements were statistically significant, expected under a small-sample regime, our analysis shows that key metrics such as error ratio and mean F1 on LIAR and max F1 on SST-2 do reach significance. Full results, post hoc comparisons, and power analyses are available in our open-source repository.

D Pareto Front Visualizations

Figure 10 presents the Pareto fronts obtained on each benchmark under the zero-shot baseline and three representative warm-start bias levels (Low, Moderate, High).

Across all datasets, warm-start priors shift the search toward regions that often dominate zero-shot pipelines in both evaluation time (ET) and task performance ($F1_{\text{macro}}$ or $F1$). Below we highlight key observations: 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.

LIAR. High-bias priors calibrated on LIAR produce up to 40% of pipelines that dominate the baseline, reducing error rates by roughly sevenfold (cf. Table 5). Due to the substantial meta-feature similarity between LIAR and MELD (Figure 3), both tasks see rapid convergence to high-$F1_{\text{macro}}$ regions.

SST2. With fewer closely related experiences, Moderate bias yields the best trade-offs, uncovering pipelines that match or slightly exceed baseline $F1_{\text{macro}}$ in less time, demonstrating robustness against negative transfer.

MELD. Figure 10c 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 3), 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 1).

AG News. Figure 10d 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 5 in the main text), suggesting that historical knowledge, even if partially relevant, helps prune more obviously unproductive hyperparameter regions.

DROP and SQuAD For QA, High bias priors achieve dramatic gains on SQuAD, raising $F1$ from 0.34 to 0.89 and cutting mean ET by 3×. On DROP, Moderate and High priors both improve $F1$ and reduce evaluation time, confirming cross-family transfer efficacy (Table 6).

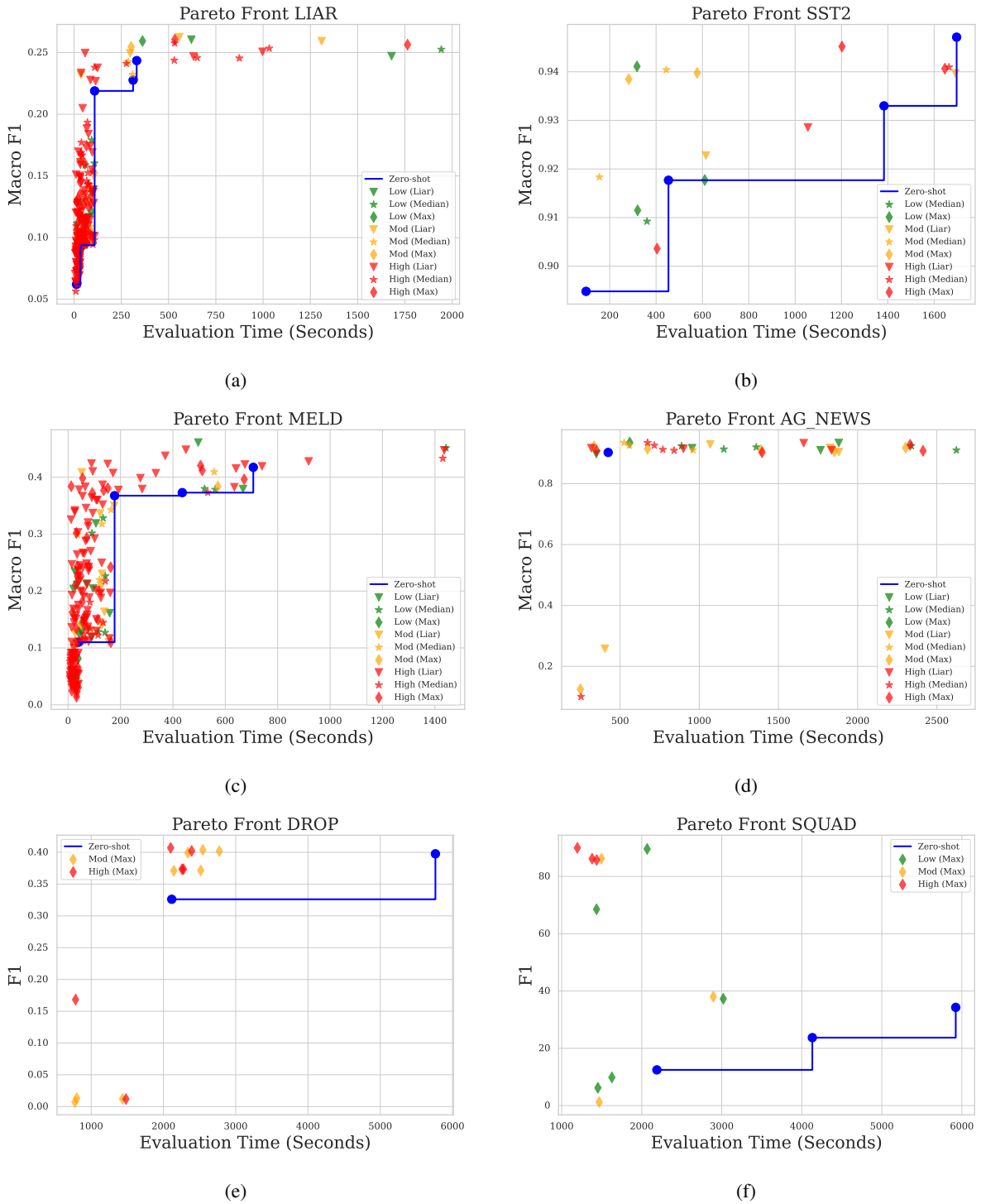


Figure 10: Comparison of Pareto fronts for zero-shot baseline (solid blue line) and warm-start priors at Low (green), Moderate (orange), and High (red) bias levels. Each point plots $(ET, F1_{\text{macro}})$ for classification tasks (a–d) or $(ET, F1)$ for QA tasks (e–f). Points to the left or above the baseline outperforms the zero-shot Pareto front.