Machine Learning (2024) 113:7011–7053
https://doi.org/10.1007/s10994-024-06568-1

Towards efficient AutoML: a pipeline synthesis approach leveraging pre-trained transformers for multimodal data

Ambarish Moharil[1,2] · Joaquin Vanschoren[1] · Prabhant Singh[1] · Damian Tamburri[2]

Received: 4 December 2023 / Revised: 7 May 2024 / Accepted: 9 May 2024 /
Published online: 19 July 2024

Abstract
This paper introduces an Automated Machine Learning (AutoML) framework specifically designed to efficiently synthesize end-to-end multimodal machine learning pipelines. Traditional reliance on the computationally demanding Neural Architecture Search is minimized through the strategic integration of pre-trained transformer models. This innovative approach enables the effective unification of diverse data modalities into high-dimensional embeddings, streamlining the pipeline development process. We leverage an advanced Bayesian Optimization strategy, informed by meta-learning, to facilitate the warm-starting of the pipeline synthesis, thereby enhancing computational efficiency. Our methodology demonstrates its potential to create advanced and custom multimodal pipelines within limited computational resources. Extensive testing across 23 varied multimodal datasets indicates the promise and utility of our framework in diverse scenarios. The results contribute to the ongoing efforts in the AutoML field, suggesting new possibilities for efficiently handling complex multimodal data. This research represents a step towards developing more efficient and versatile tools in multimodal machine learning pipeline development, acknowledging the collaborative and ever-evolving nature of this field.

Keywords Automated machine learning (AutoML) · Multimodal data · Pre-trained transformer models · Bayesian optimization (BO)

Editors: Sarunas Girdzijauskas, Myra Spiliopoulou.

✉ Ambarish Moharil
a.moharil@tue.nl

Joaquin Vanschoren
j.vanschoren@tue.nl

Prabhant Singh
p.singh@tue.nl

Damian Tamburri
d.a.tamburri@tue.nl

[1] Artificial Intelligence and Data Engineering Lab, Eindhoven University of Technology, Eindhoven, The Netherlands

[2] Jhernonimus Academy of Data Science, Eindhoven University of Technology, s'-Hertogenbosch, The Netherlands

1 Introduction

The evolution of Automated Machine Learning (AutoML) has been pivotal in addressing the complexity of developing specialized, end-to-end machine learning pipelines. However, existing AutoML frameworks often face a critical challenge: Efficiently and accurately handling multimodal data (Liu et al., 2021; Wistuba et al., 2019). The lack of generalised frameworks for multi-modality processing (Baevski et al., 2022) and dearth of systemic comparisons between varying information fusion techniques (Liang et al., 2021), prove to be the current hurdles for multimodal AutoML. Multimodal Neural Architecture Search (NAS) is notably resource-intensive, creating a barrier to efficient pipeline development (Elsken et al., 2018; Liu et al., 2018). This paper addresses this challenge by proposing a novel approach that leverages the power of pre-trained Transformer models, renowned for their effectiveness in diverse domains such as Natural Language Processing and Computer Vision (Du et al., 2022; Öztürk et al., 2022; Qiu et al., 2020).

Our approach aims to minimize the reliance on NAS for processing multimodal data. By integrating pre-trained Transformer models, we efficiently bridge the gap between different data modalities, transferring knowledge and reducing the computational overhead commonly associated with NAS. Furthermore, we enhance this integration with warm-started Bayesian Optimization technique which is an intelligent mechanism to guide the search process for optimal pipeline configurations. It leverages historical data and prior experiences, akin to the adaptability observed in human cognitive processes (Van Ackeren et al., 2018). By initiating the search from a promising region within the configuration space, we substantially lower computational demands, providing a practical and effective solution for multimodal AutoML tasks.

This research presents a comprehensive solution to the dual challenges of multimodal data processing in AutoML: reducing the dependency on expensive NAS methods (Wistuba et al., 2019) and effectively integrating pre-trained Transformer models (Liu et al., 2021; Zöller & Huber, 2019). Our approach not only streamlines the development of multimodal ML pipelines but also ensures their adaptability and efficiency, representing an advancement in AutoML for handling complex data modalities such as tabular-text, text-vision, and vision-text-tabular configurations. The major contributions of this paper include the design of a versatile search space (pipeline) for multimodal data, the strategic incorporation of pre-trained models within the pipeline architectures, and the implementation of warm-starting for SMAC using metadata derived from prior evaluations. This novel methodology underscores our commitment to enhancing AutoML's capability to navigate and optimize multimodal data processing efficiently.

2 Related works

2.1 Automated machine learning (AutoML)

Automated Machine Learning has become a crucial component in data-driven decision-making, enabling domain experts to utilize machine learning without needing extensive statistical expertise. AutoML primarily revolves around the Combined Algorithm Selection and Hyperparameter Optimization (CASH) concept, focusing on scalability and computational efficiency (Zöller & Huber, 2019). While frameworks like the Tree-Based Pipeline

Optimization Tool (TPOT) have showcased the potential for sophisticated pipelines in AutoML (Olson & Moore, 2016), challenges persist, especially in Neural Architecture Search (NAS) and hyperparameter optimization.

NAS, essential for customizing architectures, struggles with computational demands and optimization complexities (Wistuba et al., 2019). Meanwhile, advancements in Bayesian optimization have enhanced hyperparameter space navigation (Zöller & Huber, 2019). However, an integrated framework harnessing both NAS and hyperparameter tuning remains elusive. This research responds to this need by proposing a unified AutoML framework that synergizes NAS with hyperparameter optimization for efficient pipeline creation and tuning. A key focus of our approach is the concept of warm-starting, drawing parallels with human cognitive processes to efficiently adapt to new tasks through meta-learning (Barrett, 2017; Vanschoren, 2020). This involves using meta-features and prior evaluations to guide the configuration search, employing a meta-learner to predict performance and expedite the convergence to optimal solutions (Hospedales et al., 2020; Nguyen et al., 2014). Such an approach not only speeds up the optimization process but also imbues AutoML systems with cognitive-like adaptability, enhancing their capability to handle diverse and novel tasks.

2.2 AutoML and pre-trained transformer models

In the AutoML domain, CLIP's integration in AutoGluon marks a pivotal shift towards multimodal learning (Erickson et al., 2020; Radford et al., 2021). Despite its innovative approach to image-text pairings, CLIP's limitations in handling text-only or image-only data highlight a need for more versatile models (Radford et al., 2021). Transformers, known for their success in NLP tasks (Devlin et al., 2018; Lan et al., 2019; Liu et al., 2019), and Vision Transformers (ViT) for vision tasks (Dosovitskiy et al., 2020), offer a promising solution with their efficient handling of long input sequences.

Pre-trained vision-language transformer models in AutoML can significantly enhance automatic pipeline synthesis by adeptly handling multimodal data (Du et al., 2022). Recent developments in transformer architectures have produced single and dual-stream models, each with unique strengths in processing multimodal inputs. Single-stream models like OSCAR, VisBERT, and VLBERT offer a unified approach but face challenges with intra-modal interactions (Li et al., 2020, 2019; Su et al., 2019), while dual-stream models like LXMERT and ALBEF excel in cross-modal attention (Li et al., 2021; Tan & Bansal, 2019). Our research explores the impact of various transformer model architectures and pre-training objectives on AutoML systems. We focus on models such as FLAVA, a dual-stream architecture generating cross-modal embeddings (Singh et al., 2021), Albef, aligning modalities before feeding them to a multimodal Transformer (Li et al., 2021), and Data2Vec, a modality-agnostic model (Baevski et al., 2022).

2.3 Configuration space Θ

The construction of a structured configuration space remains central to our inquiry into integrating pre-trained Transformer models within AutoML systems. The configuration space of an AutoML system forms an integral and fundamental part of generating automated pipelines. The configuration space is a space that any search algorithm explores to find specific elements of a Machine Learning Pipeline (Hutter et al., 2019). This space is structured and parameterized to confine the search of a *search*

algorithm (Hutter et al., 2019; Öztürk et al., 2022). Current AutoML systems like AutoWEKA(Thornton et al., 2012), AutoGluon(Erickson et al., 2020) and AutoSklearn (Feurer et al., 2020) construct these spaces as a hierarchical space to enable a guided search strategy. Given n hyperparameters (continuous or categorical) $\lambda_1, \lambda_2 \cdots \lambda_n$ with domains $\Lambda_1, \Lambda_2 \cdots \Lambda_n$, the configuration space Θ is a subset of the crossproduct of these domains: $\Theta \subset \Lambda_1 \times \cdots \times \Lambda_n \cup \lambda_r$, where λ_r is a root-level hyperparameter. This subset is strict, such as when certain settings of one hyperparameter render other hyperparameters inactive, inducing a hierarchical structure within the configuration space (Thornton et al., 2012). This hierarchical structure of the configuration space remains *critical* as it prevents the sampling of incompatible hyperparameters. Incompatibility or a negative transfer can occur two-fold. The configuration space might sample hyperparameters that do not align with the *sampled* pre-trained model or the sampled pre-trained model, also as a hyperparameter does not align with the said task (λ_r), leading to a negative transfer. More formally, following (Thornton et al., 2012), we say that a hyperparameter λ_i is conditional on another hyperparameter λ_j, such that λ_i is only active if hyperparameter λ_j takes values from a given set $\mathcal{V}_i(j) \subsetneq \Lambda_j$ in this case, we call λ_j a parent of λ_i. Conditional hyperparameters can in turn be parents of other conditional hyperparameters, giving rise to a tree-structured space (Thornton et al., 2012).

Any selected search strategy to explore the structured hierarchical configuration space should handle the exploration-exploitation trade-off i.e. finding well-performing algorithms while avoiding premature convergence to a region of sub-optimal algorithms (Elsken et al., 2018). According to Vanschoren (Hospedales et al., 2020), a configuration space Θ^* (continuous, categorical or mixed) consisting of hyperparameter settings, pipeline components and/or network architecture components can be learned by *meta-learning* from evaluations of algorithms on some set of prior evaluations P.

2.4 Multimodal learning and fusion techniques

In light of the developments in Automated Machine Learning (AutoML), particularly in Neural Architecture Search (NAS) and Hyperparameter Optimization (HPO), this section focuses on the fusion strategy for integrating multimodal information. The motivation stems from the challenges in multimodal representation learning, as detailed by Liang et al. through MULTIBENCH, a comprehensive benchmark for multimodal learning spanning diverse datasets and modalities (Liang et al., 2021). Among various fusion paradigms like Early Fusion, Multiplicative Interactions, and Temporal Attention Models, Late Fusion (LF) emerges as notably effective. LF demonstrates a favorable balance between performance and robustness, outperforming more complex methods like MFAS or MuLT (Liang et al., 2021).

Furthermore, LF's adaptability across various domains is highlighted by Shi et al. (2021), who demonstrate its efficacy in synthesizing end-to-end ML pipelines for combined tabular and text modalities (Erickson et al., 2022). They also show LF's high accuracy in different AutoML strategies. Based on this empirical evidence and the need for robust, adaptable fusion methods in multimodal learning, Late Fusion is selected as the fusion strategy in our pipeline architecture. This choice aligns with the current research trends and promises enhanced performance and adaptability in handling multimodal datasets within our AutoML framework.

Table 1 Mathematical Notations and Their Representations

Notation	Representation
A	Set of learning algorithms
λ	Hyperparameters of algorithms
Λ	Hyperparameter space
M	Meta-dataset containing vectors of configuration settings and performance
$m(j, i)$	Vectors encapsulating configuration settings and their performance
ψ_L	Meta-learner that predicts performance metrics
μ_{ji}	Predicted mean performance by the meta-learner
σ_{ji}	Predicted standard deviation of performance by the meta-learner
$a_{\mathcal{M}l}$	Acquisition function used in selecting initial setups
λ_w	Initial configuration selected by warm-starting
λ_r	Root-level hyperparameter
R	Performance measure
D	Dataset
g	Pipeline structures
P	Pipeline configuration
L	Loss function
E	Expected value
$h(X)$	Hypothesis or model prediction function
Y	Target variable
$\hat{Y}$	Outcome
k	Number of folds in cross-validation

3 Problem formulation

In this subsection, we shall formally describe and present the problem studied in the scope of this work. To better understand the formalism, a list of all the mathematical notations can be found in Table 1.

3.1 Combined algorithm selection and hyperparameter optimization (CASH)

Our objective for incorporating pre-trained (Transformer) models in AutoML systems is to enable AutoML over unimodal as well as multimodal data. For that, we formulate a Combined Algorithm Selection and Hyperparameter Optimization (CASH) problem. CASH entails selecting optimal learning algorithms from a set $\mathcal{A}$, which includes pre-trained deep models ($A^{(i)}_{ptm}$) and classical machine learning models ($A^{(i)}_{m}$), and fine-tuning their hyperparameters (λ) for peak performance.

Formally:

- *Algorithm set* $\mathcal{A}$: Comprises m pre-trained models ($A^{(1)}_{ptm}, \ldots, A^{(m)}_{ptm}$) and n classical ML models ($A^{(1)}_{m}, \ldots, A^{(n)}_{m}$).
- *Hyperparameter space* Λ: A subset of the Cartesian product of individual hyperparameter domains, $\Lambda \subset \prod_{i=1}^{m} \Lambda^{(i)}_{ptm} \times \prod_{j=1}^{n} \Lambda^{(j)}_{m}$.

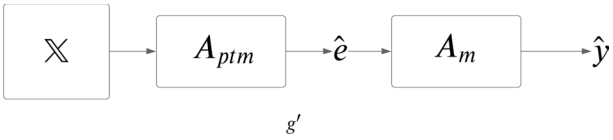


Fig. 1 Overview of the pipeline structure g' whose components need to be generated and hyperparameters need to be optimised in a combined fashion. A_{ptm} is the selected pre-trained model which generates the embedding $\hat{e}$, which is fed to a classical ML model A_m for mapping $\hat{e}$ to the target domain $\mathbb{Y}$

- *Pipeline structures set* G: All valid combinations of one pre-trained model and one classical ML model. The abstraction of a valid pipeline structure g' can be seen in Fig 1.

Embeddings ($\hat{e}$) from the pre-trained model lie in a high-dimensional space $\mathbb{R}^E$ and are inputs to classical ML models for mapping to the target domain $\mathbb{Y}$. The goal is to identify the optimal pipeline configuration g', which comprises a pre-trained model (A_{ptm}) and a classical ML model (A_m), each with its tuned hyperparameters (λ_i^* and λ_j^*, $i \neq j$). The true performance of a pipeline configuration is given by:

$$\hat{R}\left(\mathcal{P}_{g',\hat{A}^*,\hat{\lambda}^*,P}, P\right) = \mathbb{E}[\mathcal{L}(h(\mathbb{X}), \mathbb{Y})] = \int \mathcal{L}(h(\mathbb{X}), \mathbb{Y})\, dP(\mathbb{X}, \mathbb{Y}) \tag{1}$$

with the true distribution $P(X, Y)$ being unknown, we approximate performance with a dataset D:

$$\hat{R}\left(\mathcal{P}_{g',\hat{A}^*,\hat{\lambda}^*,D}, D\right) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(h(x_i), y_i) \tag{2}$$

where $\mathcal{L}$ is the loss function and $h()$ is the approximation function used to describe the true process. The objective is to minimize this estimated performance over k-fold cross-validation:

$$\left(g', \hat{A}^*, \hat{\lambda}^*\right)^* = \underset{\hat{A}^* \in \mathcal{A}^{|g'|}, g' \in G, \hat{\lambda}^* \in \Lambda}{\arg\min} \frac{1}{k} \sum_{i=1}^{k} \hat{R}\left(\mathcal{P}_{g',\hat{A}^*,\hat{\lambda}^*,D_{train}^{(i)}}, D_{valid}^{(i)}\right) \tag{3}$$

Optimization occurs across a hierarchical hyperparameter space, including a root-level hyperparameter (λ_r) for algorithm and pipeline structure selection. It is important to note that our approach does not assume that all pre-trained models will align with the specific learning tasks. Instead, we manage this issue through a structured hierarchical configuration space where the selection of a pre-trained model is analogous to choosing a hyperparameter. This configuration space is designed to activate certain pre-trained models only if the task-specific root-level hyperparameter λ_r permits their inclusion. This method ensures that only relevant and task-appropriate models are considered, thereby minimizing the risk of negative transfer.

3.2 The problem of warm-starting

In the context of AutoML's CASH problem, warm-starting is the process of initiating the optimization with informed configurations derived from prior knowledge, as opposed to random

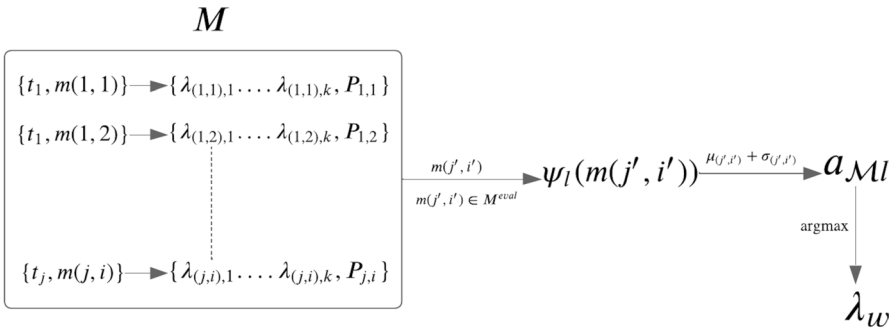


Fig. 2 Diagrammatic representation of the meta-learner ψ_L facilitated process for selecting the initial configuration λ_w to initiate the BO search

initial configurations in cold-starting. We define warm-starting as the idea to leverage historical data or results from related tasks to jump-start the current optimization task, aiming to reduce the time and computational resources required to reach an optimal solution. The assumptions that encompass the above definition are: sufficient availability of meta-data for initial configuration selection is presumed. Additionally, warm-starting presupposes a balance between exploiting known effective configurations and exploring new possibilities within the configuration space to prevent local optima, thus assuming a vast complex search space. Finally, it operates on the premise that beginning the optimization process near potentially optimal solutions incrementally enhances efficiency and effectiveness, favoring steady progress over random breakthroughs.

To facilitate the warm-starting process, we introduce a meta-dataset M, illustrated in Fig. 2, formalized as follows:

- The meta-datset M encompasses scalar performance metrics $P_{j,i}$, each derived from evaluating the ith configuration of an ML pipeline for a task t_j from a set of tasks T. Such configurations integrate a pre-trained model $A_{ptm}^{(m)}$ and a traditional ML model $A_m^{(n)}$, alongside their aggregate hyperparameters $\lambda_{j,i} \in \Lambda$, on a given task $t_j \in T$ in a pipeline structure g'. The meta-dataset M thus incorporates $P_{j,i}$ as real-valued outcomes or targets, juxtaposing a variety of categorical and numerical hyperparameters $\lambda_{j,i} \in \Lambda$ as the features within M for a task t_j, where $\lambda_{j,i} \in \mathbb{R}^k$.

Utilizing a meta-learner ψ_L we project performance indicators $(\mu_{j,i}, \sigma_{j,i})$ for these configurations ($\psi_L : \Lambda \mapsto \mathbb{R}$). Furthermore, by segregating M into distinct sets for training (M^{train}) and evaluation (M^{eval}), we employ an acquisition function $a_{\mathcal{M}l}$, to *accurately* identify an initial configuration λ_w. This configuration λ_w is aimed at finding an initial configuration that strikes a balance between predicted efficacy and potential for further exploration, formalized as follows:

$$\lambda_w \in \underset{\lambda_w, m(j',i') \in M^{eval}}{\arg\max} \sum_{i=1}^{K} \frac{1}{K} a_{\mathcal{M}l}\left(\psi_{L,M^{train}}\left(m\left(j', i'\right)\right)\right) \tag{4}$$

The ultimate goal in a warm-started Bayesian Optimization (BO) framework is to ascertain the optimal machine learning pipeline $\mathcal{P}_{g',\lambda^*}$. This pipeline is composed of a combination

of pre-trained and classical models denoted by λ^*, configured within a valid structural form g', where $g' \in G$. The optimization seeks to minimize the empirical risk $\hat{\mathcal{R}}$ over the pipeline configurations and structure:

$$\left(g', \lambda^*\right)^* \in \underset{\lambda^* \in \Lambda, g' \in G}{\arg\min} \frac{1}{k} \sum_{i=1}^{k} \hat{\mathcal{R}}\left(\mathcal{P}_{g', \lambda^*, D_{train}^{(i)}}, D_{valid}^{(i)}\right) \tag{5}$$

This is complemented by the meta-learning strategy, where a meta-learner ψ_L forecasts the mean and standard deviation of the performance for given configurations. The acquisition function is subsequently optimized across the evaluation set to select an initial configuration λ_w that maximizes the expected performance. This initial configuration λ_w is then employed to commence the BO search procedure. Figure 2 provides an abstract overview of the formulated process.

3.3 Configuration space (Θ) complexity

The Configuration Space, denoted as Θ, is delineated as a complex hybrid space composed of learning algorithms $\mathcal{A}$, the hyperparameter space Λ, and the pipeline structures G. Elements within Θ are categorized as either numerical or categorical. The diversity, $D_{\text{numerical}}$, across n numerical hyperparameters in Θ, where $H_{i,\max}$ and $H_{i,\min}$ signify the maximum and minimum allowable values for the ith numerical parameter, is defined as:

$$D_{\text{numerical}} = \prod_{i=1}^{n} (H_{i,\max} - H_{i,\min}) \tag{6}$$

For hyperparameters scaled logarithmically (e.g., weight decay and layer normalization ϵ), the term $(H_{i,\max} - H_{i,\min})$ is substituted by $\log(\frac{H_{i,\max}}{H_{i,\min}})$. Conversely, the diversity for m categorical hyperparameters, $D_{\text{categorical}}$, within Θ is the product of available categories for each hyperparameter:

$$D_{\text{categorical}} = \prod_{j=1}^{m} C_j \tag{7}$$

where C_j represents the number of categories for the jth categorical hyperparameter. The overall complexity of Θ is thus a function of both numerical and categorical diversity, alongside the aggregate possible configurations (T_c), formulated as:

$$T_c = D_{\text{numerical}} \times D_{\text{categorical}} \tag{8}$$

Incorporating values from Table 2:

- For categorical hyperparameters: Pretraining Model (8 options), Pretraining Processors (3 options), Downstream model (12 options), and Downstream processor (1 option).
- For numerical hyperparameters: Totaling 8, with their respective ranges considered for computation.
- The Θ space encompasses two subsets related to pre-trained models Θ_{ptm} and Neural Architecture Search (NAS) algorithms Θ_{NAS}.

Hence, the complexity, T_C, is determined as:

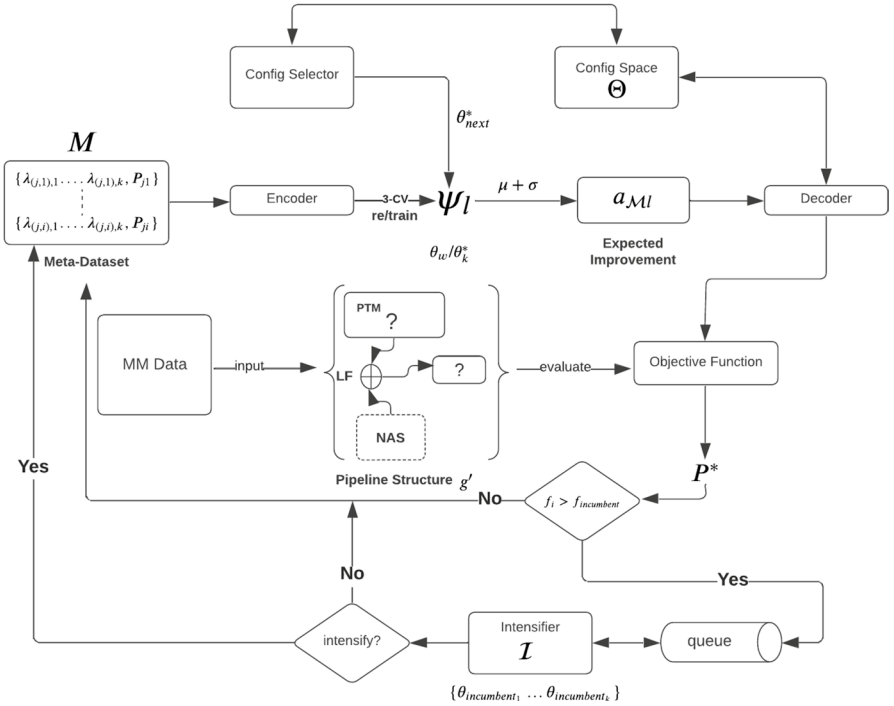


Fig. 3 Detailed overview of the Pretrained Transformer-based AutoML (PTA) system for warm-starting AutoML over multimodal data. LF denotes Late-Fusion

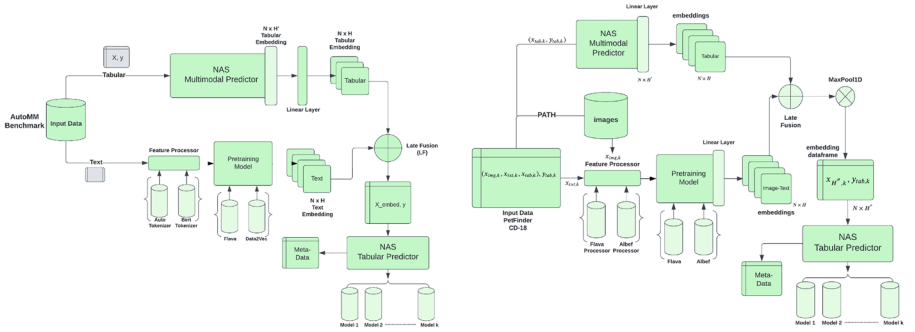
$$|T_c| = T_{c,\Theta_{ptm}} + T_{c,\Theta_{NAS}} = 9.38 \times 10^{14} \tag{9}$$

with $T_{C,\Theta_{ptm}}$ and $T_{C,\Theta_{NAS}}$ indicating the complexities of the spaces comprising hyperparameters associated with pre-trained models and NAS algorithms, respectively. Consequently, the estimated complexity of our constructed configuration space Θ approximates 938 trillion possible configurations, emphasizing the computational challenge in optimizing within this extensive space.

4 Pretrained transformer-based AutoML (PTA) methodology

4.1 Overall methodology

Our research integrates pre-trained deep neural models into Automated Machine Learning (AutoML) Systems to efficiently process multimodal data. Our methodology can be divided into 4 sequential steps namely: Prior Evaluation and Meta-Dataset Construction, Configuration Space (Θ) Construction, SMAC Setup and Execution, Evaluating the Optimisation. The rest of this section intends to explain each of the above-mentioned steps in detail. Figure 3 describes the overall workflow of our proposed Pre-Trained Transformer Based AutoML (PTA) framework. Meta-dataset M is constructed as a result of extensive prior (to the SMAC optimisation) evaluations across diverse pipeline configurations, tasks



(a) Pipeline Variant 1 for Tabular-Text Modality

(b) Pipeline Variant 1 for Tabular-Text-Vision Modality

Fig. 4 Architectural Designs of the Pipeline Variant 1 for the Tabular-Text and Tabular-Text-Vision Modality

as well as datasets following some pipeline structure g'. Moreover, Θ represented in the diagram is the constructed search space after building the meta-dataset M. Encoder, Meta-Model (ψ_L), acquisition function, decoder, objective function, intensifier, and configuration selector are all components within SMAC. We shall understand the interactions of these components in detail in Sect. 4.4.

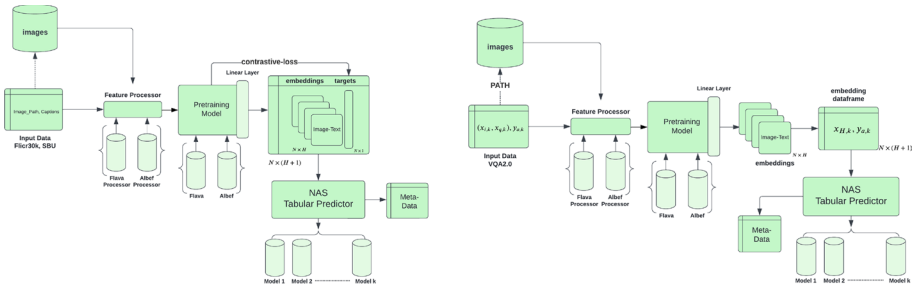
4.2 Prior evaluations and meta-dataset construction

4.2.1 Prior evaluations: pipeline variants of multimodal AutoML

In this step, we evaluate task-specific variants of multimodal pipeline architectures constructed in a specific pipeline structure g' across datasets belonging to the tabular-text, text-vision and tabular-text-vision modalities. Furthermore, we evaluate these variants of pipeline architectures across tasks like classification, regression, Image Text Matching (ITM), and Visual Question Answering (VQA). Our framework includes 3 specific pipeline variants, each designed for a specific modality-task combination.

Pipeline Variant 1

- *Tabular-Text Modality* This variant, focusing on classification and regression tasks, utilizes FLAVA and Data2Vec models for text data processing and NAS-derived MLP (multimodal-net) for tabular data. The late fusion strategy combines these embeddings, which are then processed by AutoGluon's Tabular Predictor. The pipeline architecture, as shown in Fig. 4a, is optimized for handling both binary (multi-label) classification as well as regression tasks. The AutoGluon Tabular Predictor explores various tabular architectures, including ensemble tree-based models, and records model performance to inform future meta-learning.
- *Tabular-Text-Vision Modality* For this modality, the pipeline employs FLAVA and Albef models to encode image-text data, and the multimodal-net for tabular data. The encoded data from all three modalities is mapped into a unified latent space, implementing late fusion and downsampling using translation invariant methods like MaxPool. The com-



(a) Pipeline Variant 2 for ITM Task (b) Pipeline Variant 3 for VQA Task

Fig. 5 Architectural Designs of the Pipeline Variant 2 and 3 respectively

bined embeddings are then processed by AutoGluon's Tabular Predictor for task-specific execution, as depicted in Fig. 4b. This variant aims to record pipeline performance over multimodal datasets, enriching the meta-dataset for future optimization strategies.

Pipeline Variant 2

Focused on the ITM task, this variant processes unlabelled image-text data using FLAVA and Albef models. Data from datasets like Flickr30k and SBU image captioning is batch-processed and encoded into unified latent embeddings. A regression task maps these embeddings to contrastive scores obtained from the pre-trained models, optimizing the mapping using AutoGluon Tabular, as shown in Fig. 5a. This approach aims to assess model performance and optimize pipeline configurations for the ITM task.

Pipeline Variant 3

Figure 5b designed for the VQA task, this variant focuses on labeled vision-language data from the VQA2.0 dataset. FLAVA and Albef models are used for encoding images and questions. Batch processing is conducted via a PyTorch class object, with the encoded data forming a unified latent space. The combined embeddings and answer targets create a new dataset, processed by AutoGluon Tabular for the VQA task. The focus is on deriving insights from visual-textual interplay and optimizing model selection for the VQA task. We record the performance evaluations (AUC scores) of the multi-label classification task conducted by AutoGluon's Tabular Predictor using different tree-based ensemble models. This prior knowledge is further incorporated in the form of the meta-dataset M.

Each variant demonstrates a unique approach to multimodal data processing, leveraging the capabilities of pre-trained models and the exploratory power of AutoGluon Tabular.

4.2.2 Meta-dataset construction

After designing the above 3 pipeline variants, a meta-dataset M is constructed by recording scalar performances $P_{j,i}$ corresponding to each of these 3 pipeline variants, across various tasks t_j selected from the set comprising of classification, regression, ITM and VQA tasks. Given the ith pipeline configuration $\lambda_{j,i}$ (pre-processing, pre-trained, traditional ML algorithm names as well as hyperparameters) for the jth task, M records $\lambda_{j,i}$ and their corresponding $P_{j,i}$. M is realized as a nested python dictionary object, where the keys of the

Table 2 Selected hyperparameters and their corresponding ranges in Θ

Hyperparameters in Θ	
Hyperparameters	Range
Attention drop-out	[0.0, 0.5]
Hidden drop-out	[0.0, 0.5]
Layer normalisation ϵ	$[10e^{-12}, 10e^{-2}]$
Pretraining model	['FlavaText', 'Data2VecText', 'FlavaVQA', 'AlbefVQA', 'Flava', 'Albef', 'FlavaITM', 'AlbefITM']
Pretraining processors	['FlavaProcessor', 'Data2VecProcessor', 'AlbefProcessor']
Linear hidden size	[256, 512]
MaxPooling kernel	[2, 6]
Weight decay	$[10e^{-8}, 10e^{-2}]$
Downstream model	['CatBoost', 'XGBoost'..]
Downstream processor	['AutoMLPipelineFeatureProcessor']
Iterations	[50, 100]
Max depth	[2, 10]
Number of boost rounds	[100, 500]
Max leaves	[50, 300]

dictionary are the hyperparameters or algorithm names and the values for the respective keys are the recorded experimental values (numerical or categorical). *M* also records the names of the pre-trained model along with the names of the traditional ML models used in the pipeline in a string format. The list of hyperparameters corresponding to different pre-trained as well as traditional ML models included within *M* can be found in Table 2.

4.3 Construction of configuration space Θ

The construction of the configuration space Θ is crucial for synthesizing effective machine-learning pipelines in our AutoML system. Θ serves as a search space for the Sequential Model-Based Optimization (SMBO) algorithm, containing various components such as pre-trained models, feature processors, and classical ML models, structured hierarchically. This hybrid space includes both categorical and numerical hyperparameters described in Table 2.The inclusion of pre-trained models like FLAVA, Albef, and Data2Vec in Θ is motivated by their distinct capabilities and performance metrics (Du et al., 2022; Khan et al., 2021). FLAVA, for instance, excels in various multimodal tasks and language benchmarks, making it a valuable inclusion for its broad applicability. Data2Vec's modality-agnostic nature is pivotal for generating universal representations, while Albef adds diversity with its specific strengths. In addition to these models, Θ encompasses conventional tree-based models, classical ML classification and regression models, and various preprocessing algorithms. This configuration space is designed to be a hybrid of categorical choices (such as selecting specific pre-trained models) and numerical hyperparameters (like layer normalization epsilon and dropout probabilities). The space is conditioned on the task type (λ_r), with hyperparameters rendered inactive based on the relevance to the

task at hand, ensuring efficient and targeted search during optimization (Refer Sects. 2.3 and 3.1).

4.4 SMAC setup and execution: warm starting PTA

Upon the assembly of the hybrid configuration space Θ, we initialise and *warm-start* the Sequential Model-Based Algorithm Configuration (SMAC) procedure. This process encompasses the meta-dataset M, meta-model ψ_L, acquisition function $a_{\mathcal{M}l}$, objective function f_θ, intensifier $\mathcal{I}$, and the configuration selector, aimed at addressing the Combined Algorithm Selection and Hyperparameter Optimization (CASH) challenge. The meta-dataset M, compiled during the preceding evaluation phase, encapsulates data on the pipeline configurations λ alongside their scalar performance evaluations P.

Following the creation of meta-dataset M, we establish the Scenario $\mathcal{S}$ for optimisation, which delineates the optimisation landscape, specifying iterations, budget, and the exploration bounds within Θ. The intricacies of our optimisation scenario $\mathcal{S}$ will be elaborated upon in the experiments and results segment. Post the configuration of $\mathcal{S}$, the meta-learner ψ_L is trained on M[1] employing 3-fold cross-validation. Having defined the modality and the task as a root level hyperparameter λ_r, the configuration selector samples only the hyperparameters activated under the defined subset of the structured configuration space. Furthermore, to ascertain the validity of the pipeline, a conditional logic is designed to check whether the sampled choice of the pre-trained model (hyperparameter) lies within the permitted zoo of models for the given type of input task and modality. With these we prevent incompatibility or negative-transfer. The configuration selector begins the optimization process by identifying a set of n *random* initial samples confined within the hierarchy and boundaries of our defined configuration space. These n configurations' performance metrics are predicted using the trained meta-learner ψ_L. The RandomForest meta-learner ψ_L then undertakes a regression task, mapping the hyperparameter configurations Λ, within a high-dimensional space $\lambda \in \mathbb{R}^k$, to a real value in $\mathbb{R}$, i.e., $\psi_L : \Lambda \mapsto \mathbb{R}$. With the mean predictive performance and uncertainty estimates derived from ψ_L, we maximize the Expected Improvement (EI) acquisition function $a_{\mathcal{M}l}$ to pinpoint the initial configuration with the *highest potential* for achieving optimal performance. The selected initial configuration λ_w, boasting the highest EI score, is extracted from the configuration space for *actual* assessment by the objective function. This function, f_θ, correlates the input features $X \in \mathbb{R}^d$ with a real-valued performance metric p, expressed as $f_\theta : \mathbb{R}^d \mapsto \mathbb{R}$. At this juncture, the pipeline, integrating the chosen configurations, is applied to the input data to fulfil the designated task objective. f_θ appraises this pipeline's efficacy for the specific configuration λ, including hyperparameters and models, converting these into a scalar metric such as AUC for classification or R^2 for regression tasks. Designed to accommodate multimodal data, f_θ adjusts its evaluations based on the data modality and the precise task, engaging diverse pre-trained models like FLAVA, Albef, and Data2Vec to ensure its assessments are both accurate and pertinent. Consequently, f_θ evaluates λ_w, updating the performance metric in M. λ_w is thus acknowledged as the initial incumbent configuration.

To select the subsequent configuration λ_{w+1}, ψ_L undergoes re-training with the refreshed M. The configuration selector then extracts m random configurations, applying a 10–12% perturbation rate around the incumbent configuration (λ_w), facilitating the exploration of well-performing setups. The configuration with the maximal expected improvement is

[1] ψ_L is trained on M by partitioning it into two disjoint subsets M^{train} and M^{eval}

earmarked for actual evaluation through f_θ. Should this configuration yield a performance metric surpassing that of the incumbent, it is incorporated into the intensifier queue, updating the incumbent and M concurrently. The intensifier $\mathcal{I}$ adopts an Aggressive Racing Strategy, probing the vicinities of promising incumbent configurations by instructing the configuration object to sample m configurations with the specified perturbation rate around these incumbents. The evaluations of these sampled configurations proceed in parallel, with the evaluation records continually refreshed in M, and ψ_L being re-trained prior to each sampling iteration. This cycle repeats until the optimisation budget depletes. Figure 3 furnishes a graphical representation of this warm-started SMAC optimisation procedure.

4.5 Evaluating the optimization: any-time learning metric

In our AutoML framework, evaluating the optimization process post-convergence of the SMAC loop is crucial. To achieve this, we adopt the 'any-time' learning metric, taking inspiration from Liu et al. (2021), emphasizing efficiency under time and data constraints. This is quantified through the Area Under Learning Curve (ALC), expressed as:

$$ALC = \frac{1}{\log\left(1 + \frac{T}{t_0}\right)} \int_0^T \frac{s(t)}{t + t_0} dt \tag{10}$$

where $s(t)$ is the scoring function. For classification tasks, we make use of the Normalised Area Under the Curve (NAUC)[2] as the scoring function. Moreover, for regression tasks, R^2 score of the model acts as the scoring function in the above equation. The ALC metric captures the learning trajectory over time, especially in the initial phases, reflecting the system's rapid learning efficiency. By computing the ALC for each dataset, we comprehensively evaluate the AutoML system's optimization process, focusing on its ability to adapt and learn effectively within limited time frames. This methodological approach thoroughly assesses the framework's learning behavior and operational efficiency in resource-constrained environments. Additionally, the hierarchical nature of the configuration space ensures the activation of only task-related hyperparameters, rendering task-unrelated hyperparameters including pre-trained models as well as their corresponding hyperparameters inactive. Thus, given this hierarchy, our evaluation function carefully evaluates scenarios where the pipeline components are entirely compatible with the learning task given a task and a dataset. The configuration sampler strives to avoid incompatibility and the hypothetical occurrence of any would result in consistent crashing and low ALC values. A high ALC value indicates consistent performance and effective sampling of learning algorithms by the AutoML framework, suggesting that the configurations chosen are generally well-suited to the tasks. In case a trial (sampled pipeline configuration) fails to complete its training within the trial budget, we record the performance of such a sample as $-inf$ or CRASH. Incompatible samples are handled similarly.

[2] NAUC $= 2 * \text{AUC} - 1$, where AUC stands for Area Under RO Curve.

5 Experiments and results

5.1 Experimental settings

Experiment 1: Prior evaluations of multimodal pipeline architectures

This experiment aims to collect prior insights by evaluating various pipeline configurations across 23 multimodal datasets, documenting the configurations $\lambda_{j,i}$ and their performance estimates $P_{j,i}$ in a meta-dataset M. We explore three variants of our proposed multimodal pipeline architectures that employ Late Fusion to integrate multiple modalities for tasks including classification, regression, image-text matching (ITM), and visual question answering (VQA). The experimental setup involves selecting a pre-trained multimodal vision-language transformer to *represent* vision-language data within a unified latent space. Additionally, we employ Neural Architecture Search (NAS) via AutoGluon to construct a dynamic multilayer perceptron (MLP) for tabular data, aligning it within the same latent space. The Late Fusion process linearizes the final embeddings, which are then utilized by AutoGluon Tabular to predict the respective targets.

For classification and regression tasks, we utilize the 18 AutoMM Benchmark Datasets introduced by Shi et al. (2021). To evaluate classification tasks we make use of the NAUC metric and for regression tasks we study thr R^2 score. The ITM task evaluations are conducted using the Flickr30k (Plummer et al., 2015) and SBU Image Captioning Dataset (Ordonez et al., 2011), while VQA task performance is assessed on the VQA 2.0 Dataset (Agrawal et al., 2015). Additionally, the PetFinder and CD-18 Datasets are employed to evaluate classification and regression tasks within the tabular-text-vision modality. The first variant of our pipeline architecture, as depicted in Figs. 4a, b, is utilized for evaluating tabular-text and tabular-text-vision modalities in classification and regression tasks, respectively. The second pipeline variant, shown in Fig. 5a, is used for ITM tasks, while the third variant addresses the VQA task. Each evaluation session spans $\approx$ 5–6 h to ensure optimal or near-optimal pipeline configurations are achieved. It is important to note that the pre-trained models' weights are kept frozen, with only the downstream ML models' weights being fine-tuned. This implies that the variation in performance observed during the optimization process arises solely from the adjustments in the hyperparameters (of the pre-trained models) rather than any changes in the model's weights. Freezing the pre-trained weights enables us to study the variation in the model performance especially when the pre-trained models are kept the same and the hyperparameters are varied and also when we vary the pre-trained models along with their hyperparameters altogether for a given task-modality and dataset.

Experiment 2: Assessing the efficacy of warm-started PTA

This experiment evaluates the SMAC optimization curve across varied pipeline variants for multimodal tasks, using the ALC metric that utilizes a scoring function $s(t)$[3] for configuration performance at times t. We focus on assessing our warm-started PTA framework's optimization quality by studying the ALC and $s(t)$ scores through the learning curves, identifying promising configurations for further analysis within a 45 min limit, as suggested by Liu et al. (2021). This Scenario $\mathcal{S}$ is set up, with trials capped at 20 min and overall optimization limited to 45 min. With this specific budget, we aim to highlight the efficiency

[3] NAUC for classification tasks and R^2 for regression tasks.

of our warm-started PTA framework in sampling *optimal* incumbent candidate/s within a vast search space under a limited budget. Unsuccessful trials are marked as *-inf* or CRASH, with successful ones $\gamma \subseteq \Gamma$ being evaluated for consistent exploration of high-performing candidates, represented by high ALC scores (≈ 1). We address the challenge of inappropriately chosen pre-trained models by including their evaluation in the ALC and marking mismatches or negative transfers with a performance score of "-inf," thereby avoiding their impact on the system's learning curve. This approach ensures that frequent incompatibilities lead to fewer evaluations and lower ALC scores, indicating areas for potential improvement. Then we analyze the learning curves across 23 multimodal datasets, plotting the scoring function $s(t)$-NAUC for classification and R^2 for regression-against log-scaled time (Fig. 6a, b) and compute the Area Under the Learning Curve (ALC) as an any-time learning metric to evaluate the consistency of optimization efforts for all the 23 multimodal datasets.

Moreover, we compare the efficacy of our warm-started PTA, leveraging pre-trained models, against an inherently cold-started NAS (*multimodal-net*) method implemented by Autogluon for handling multiple modalities. The main reason for this comparison is to showcase an efficient exploration of a complex search space under budgeted constraints, based on a *hybrid* pipeline approach (pre-trained transformers + NAS) infused with prior knowledge as compared to a computationally expensive cold-started, NAS method. Furthermore, we implement Late Fusion (LF) for integrating modalities as suggested by Shi et al. (2021), Liang et al. (2021), a method performing the best across 18 real-world datasets as showcased by Shi et al. (2021) with their experiments on Autogluon. Furthermore, the only pre-trained multimodal model implemented by Autogluon is CLIP, apart from its NAS methods for handling multimodal data, we *intend* to compare autogluon's performance against a set of a hybrid AutoML architecture, incorporating more sophisticated multimodal models.

For tabular-text modality, we benchmark against the *AutoMM Benchmark* by Shi et al. (2021), using a cold-started NAS-based *multimodal-net* over 5–6 h. Absent a public benchmark for vision-language tasks, we assess the warm-started PTA's (under 45 min budget constraint) performance against Autogluon's NAS-based *multimodal-net* to evaluate incumbent scores on a similar budget of 45 min. With this comparison, we aim to show that by reducing the dependency on NAS for multimodal processing, the efficiency of pipeline generation under a constrained budget could be significantly improved.

5.1.1 Results: experiment 1

In this section, we report some observed average metrics for the datasets and tasks selected for prior evaluations. For the AutoMM Benchmark datasets, we report the average AUC and R^2 scores (μ in Table 3) obtained by the FLAVA and Data2Vec pipeline variants across the classification and regression datasets from the AutoMM benchmark respectively. For the Flickr30k, SBU, VQA, Petfinder and CD-18 datasets, we report the average scores (μ) obtained across different pipeline hyperparameters and downstream traditional ML models fitted through Autogluon Tabular, after fixating a pre-trained multimodal model within the multimodal pipeline.

Tabular + Text modality

FLAVA demonstrates an average AUC score of 0.354 across the classification datasets and an average R^2 score of 0.415 across the regression datasets, with a minimal

Table 3 FLAVA, Albef, and Data2Vec prior evaluation results across 23 Datasets (18 AutoMM, Flickr30k, SBU, PetFinder, and CD-18) for the 3 modalities, over classification, regression, ITM, and VQA tasks

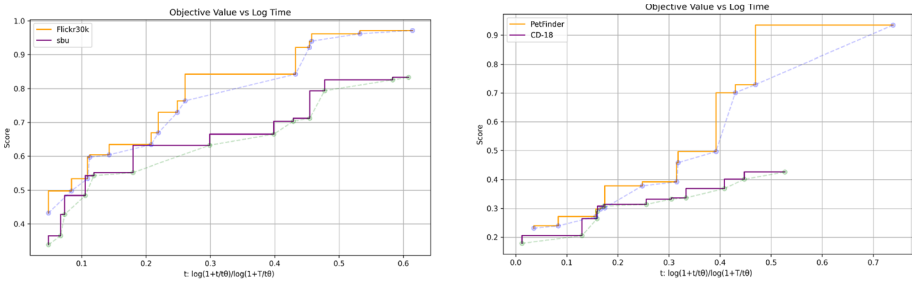
Modality	Data	σ^2	μ^*	Pred time (s)	Fit time (s)
FLAVA pipelines: avg performance (classification tasks)					
Txt-Vis	VQA2.0	0.026	0.931*	1535.239	4876.595
Tab-Txt	AutoMM clf	0.001	0.354	248.238	3282.349
Tab-Txt-Vis	PetFinder	0.101	0.373	1927.186	1387.345
FLAVA pipelines: avg performance (regression tasks)					
Tab-Txt	AutoMM reg	0.003	0.415*	164.497	4643.201
Txt-Vis	Flickr30k	0.042	0.205*	3691.097	7499.211
Txt-Vis	SBU	0.012	0.177*	4447.424	14814.139
Tab-Txt-Vis	CD-18	0.139	0.412*	215.790	906.433
Albef pipelines: avg performance (classification tasks)					
Txt-Vis	VQA2.0	0.001	0.933*	1927.186	4651.606
Tab-Txt-Vis	PetFinder	0.108	0.393	207.764	1464.711
Albef pipelines: Avg Performance (Regression Tasks)					
Txt-Vis	Flickr30k	0.052	0.216*	3.983	5139.134
Txt-Vis	SBU	0.013	0.181*	4.265	2886.672
Tab-Txt-Vis	CD-18	0.122	0.424*	178.171	816.697
Data2Vec pipelines: avg performance (classification and regression tasks)					
Tab-Txt	AutoMM clf	0.002	0.272	0.187	2105.509
Tab-Txt	AutoMM reg	0.001	0.236*	0.146	1607.344

Variance (σ^2) and mean (μ) are shown for scores, where σ^2 denotes the variance in score across different downstream ensemble models. A star (*) indicates R^2 scores values for regression tasks. Scores without the star (*) indicate the AUC scores

variance of 0.001 and 0.003 respectively, indicating consistent handling of tabular and text data. This performance is 23% and 55% higher than Data2Vec's average AUC and R^2 scores of 0.272 and 0.236 respectively. However, it's important to note that FLAVA requires a higher average prediction time across both classification and regression datasets compared to Data2Vec. Despite this longer prediction time, FLAVA's scores are not significantly higher than the ones obtained on the relatively faster Data2Vec pipelines.

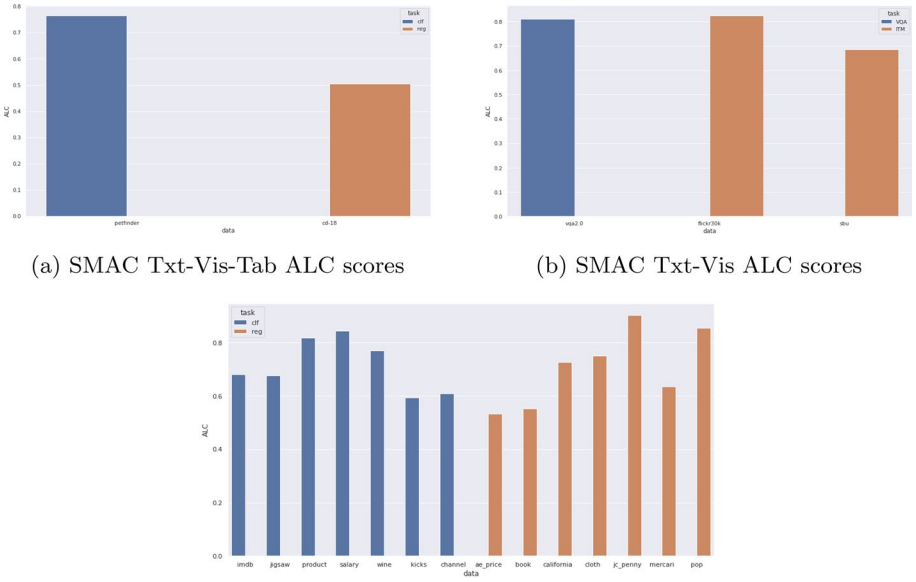
Text + Vision modality

For the Visual Question Answering task over the VQA2.0 dataset, Albef and FLAVA give ≈ similar performances, an AUC score of 0.933 and 0.931 respectively, with Albef pipelines performing slightly better on average across different downstream models. The observed difference in the average prediction and fit times across Albef and FLAVA pipelines appears to be relatively small, with both pipeline configurations showing consistent performances ($\approx 0\ \sigma^2$) across different downstream ML models. In the ITM task, Albef slightly outperforms FLAVA on the Flickr30k (0.216 vs. 0.205) and SBU datasets (0.181 vs. 0.177). However, Albef pipelines are extremely efficient than the FLAVA pipelines for prediction on ITM tasks. The average prediction time across the VQA dataset is approximately the same for the pipeline configurations consisting of FLAVA and Albef models.



(a) Flickr30K & SBU datasets (b) PetFinder & CD-18 datasets

Fig. 6 Learning curves depicting the obtained R^2 scores for the Flickr30k, SBU (6a) and CD-18 datasets (6b) and the NAUC scores for the PetFinder dataset (6b), as a function of time across different pipeline configurations evaluated during the SMAC optimisation process for the ITM and VQA tasks respectively. x-axis: $log(t)$, y-axis: $s(t)$



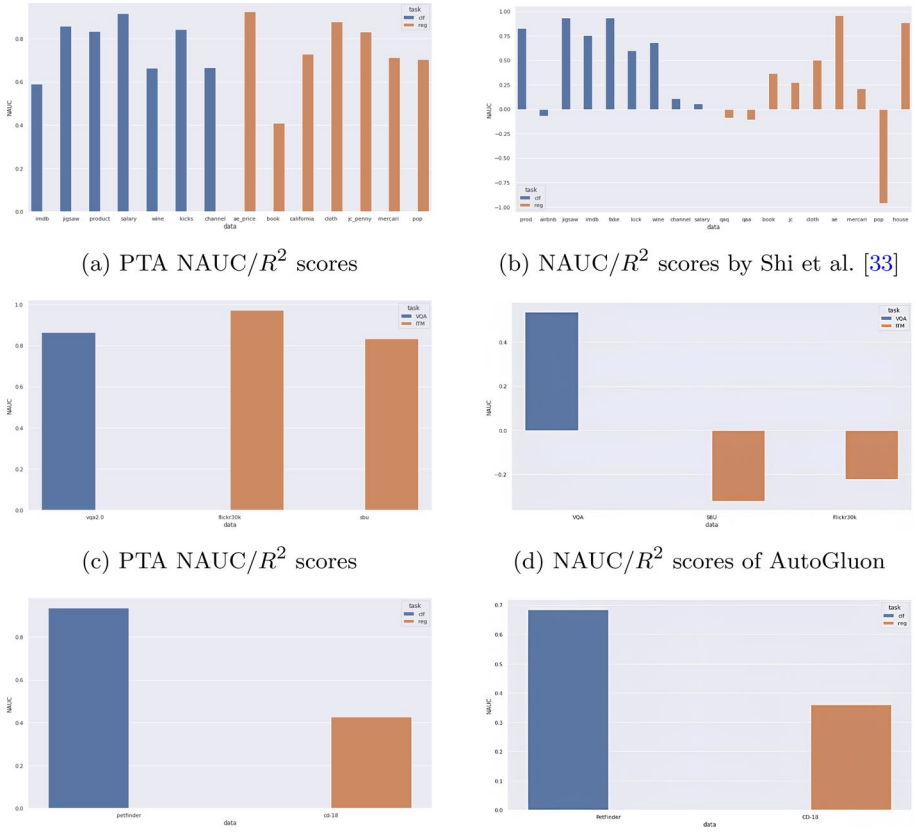
(a) SMAC Txt-Vis-Tab ALC scores (b) SMAC Txt-Vis ALC scores

(c) SMAC Txt-Tab ALC scores

Fig. 7 ALC scores obtained using our warm-started PTA for the 3 studied modalities under a budget T (45 mins). x-axis: datasets, y-axis: ALC scores. blue: classification data, orange: regression data

Tabular + Text + Vision modality

Albef's performance in the complex multimodal scenarios of PetFinder and CD-18 datasets (0.393 and 0.424, respectively) edges out FLAVA's scores (0.373 and 0.412), showing a 5% to 3% improvement. Albef also shines in computational efficiency, particularly in the CD-18 dataset, where its average prediction time is 178.171 s, nearly 20% faster than FLAVA's 215.790 s.



(a) PTA NAUC/R^2 scores

(b) NAUC/R^2 scores by Shi et al. [33]

(c) PTA NAUC/R^2 scores

(d) NAUC/R^2 scores of AutoGluon

(e) PTA NAUC/R^2 scores

(f) NAUC scores for AutoGluon

Fig. 8 Comparison of the NAUC/R^2 scores obtained using our PTA framework with AutoGluon for the Tabular-Text, Text-Vision and Tabular-Text-Vision modalities respectively (y-axis: 0 to 1). x-axis: datasets, y-axis: NAUC or R^2 ($s(t)$) scores. blue: classification data, orange: regression data

Stability Across Models

The stable performance of FLAVA and Albef, as indicated by low variance scores σ^2, highlights their reliability in AutoML strategies. Despite Data2Vec's modest performance, its modality-agnostic feature stands out. Our evaluation compares pipeline performances with pre-trained models such as FLAVA, Albef, and Data2Vec, underpinning the belief that high-quality, higher-dimensional representations are crucial for predictive accuracy. These representations are vital for the success of downstream models, with the optimization and quality of embeddings significantly impacting performance outcomes. High-quality embeddings from these pre-trained models are key to strong performance, underscoring their importance in our AutoML framework. The performances of FLAVA, Albef, and Data2Vec across various modalities offer insights into their distinct strengths, informing future AutoML model selection and strategy enhancements.

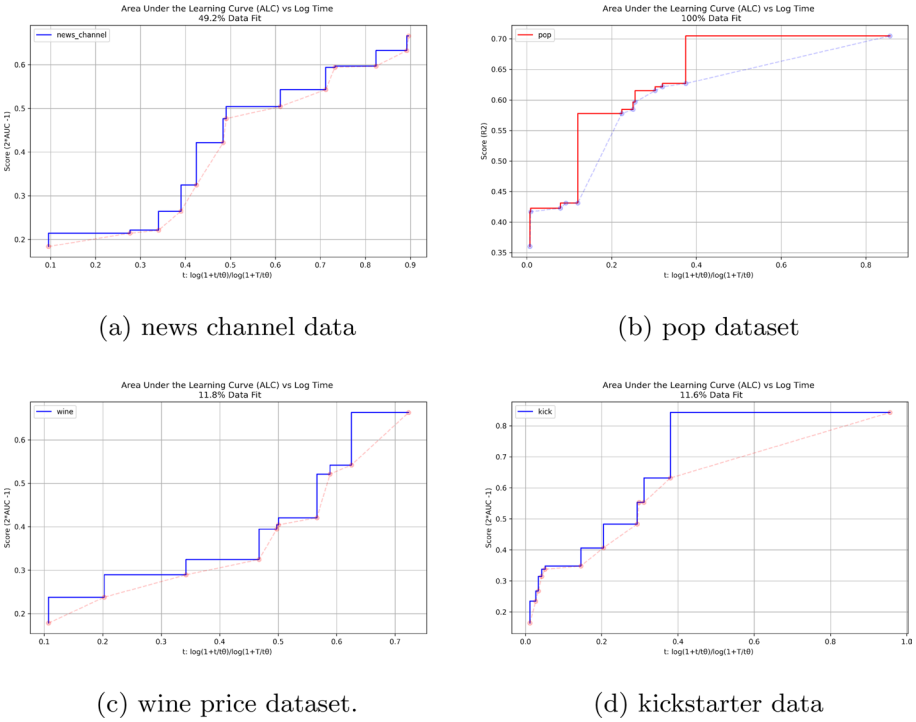
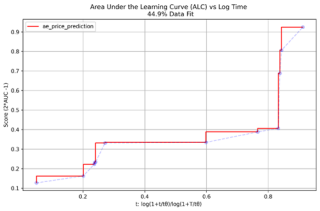


Fig. 9 Warm-Started SMAC Results for the AutoMM Benchmark (Shi et al., 2021)

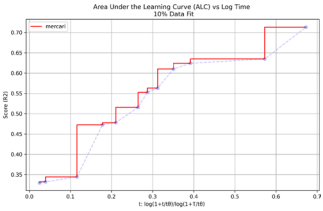
5.1.2 Results: experiment 2

Figures 7a–c illustrate the ALC values from the 23 multimodal datasets, highlighting the efficiency and efficacy of our warm-started PTA framework through high ALC scores of 0.762, 0.803, 0.815, and 0.693 for the Petfinder, VQA2.0, Flickr30k, and SBU datasets, respectively, across tabular-text-vision and text-vision modalities. The learning curves depicted in Fig. 6a, b, representing the Flicker30K, SBU, Petfinder, and CD-18 datasets, underscore the ALC values, with scores nearing 1 signifying the generation of *incumbents* with optimal $s(t)$ values throughout the 45 min optimization period.

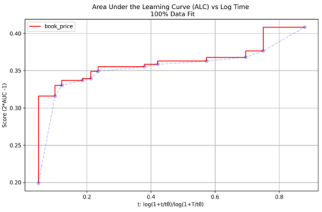
For the tabular-text classification task, the *salary* dataset achieved the highest ALC score of 0.819 (Fig. 7c). The average ALC score for the classification datasets within the AutoMM benchmark is 0.711, as shown in Fig. 7c, indicating substantial performance despite budget limitations. Additionally, an average ALC score of 0.702 was noted for regression datasets (Fig. 7c), demonstrating our framework's capability to produce effective pipeline configurations under time constraints. Comparing the *incumbents* synthesized by our warm-started PTA framework against those from a cold-started NAS method over a similar budget of 45 min, Fig. 8 contrasts the $s(t)$ scores across both approaches for selected datasets. The PTA *incumbents* for tabular-text classification datasets averaged higher performance (0.7521) compared to the NAS-based *multimodal-net* by Autogluon (0.530) (Fig. 8a, b), and for tabular-text regression tasks, PTA *incumbents* also outperformed (0.695 R^2) against the NAS-based approach (0.196R^2, Fig. 8a, b). Specifically, for



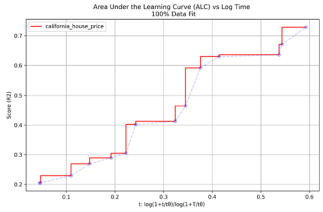
(a) ae price



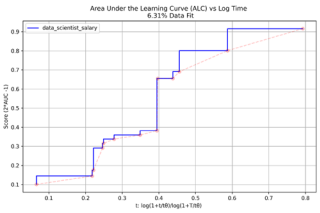
(b) mercari price data



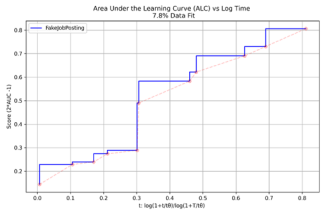
(c) book price pred



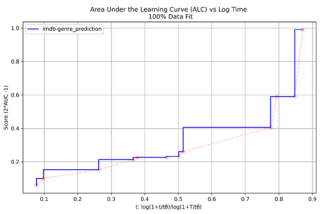
(d) california house data



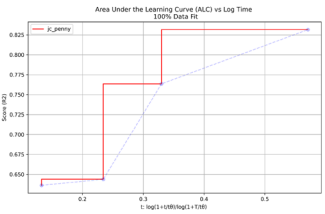
(e) data scientist salary



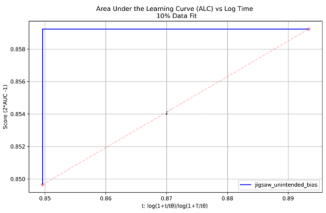
(f) fake job data



(g) imdb genre data



(h) jc penny data.



(i) jigsaw unintended

Fig. 10 Warm-Started SMAC Results for the AutoMM Benchmark (Shi et al., 2021)

the VQA2.0 dataset, the warm-started PTA framework achieved a NAUC score of approximately 0.820 (Fig. 8c), surpassing the NAS-based approach's score of approximately 0.433 (Fig. 8d). Similarly, ITM tasks on the Flickr30k and SBU datasets resulted in *incumbent* R^2 scores of 0.912 and 0.810 (Fig. 8c), respectively, demonstrating superior performance over the Autogluon MLP (Fig. 8d). For tabular-text-vision modalities, like the Petfinder and CD-18 datasets (Fig. 8e, f), the performance of PTA *incumbents* closely matched the NAS-based *incumbents*. Based on these results and additional results stated in the Appendix (Figs. 9, 10), we could infer that it takes Autogluon considerable amount of resources, both computational and temporal to construct a complex Neural Architecture from scratch, especially for non-tabular modalities. The high ALC values obtained for our approach across 23 datasets underscore the efficiency of an approach, utilising NAS only for the tabular modality and leveraging prior experiences for an informed search across a complex search space, tends to *consistently* generate well-performing pipeline configurations, under a constrained computational budget. Moreover,the combined cross and intra modal interactions facilitated by our pre-trained models, preserve and capture complex non-linear interactions ($\hat{e}$), which turn out to be more informative than the representations constructed by Autogluon's *multimodal-net*.

6 Conclusion and discussion

This study advances the Automated Machine Learning (AutoML) field, emphasizing multimodal data processing across visual, textual, and tabular inputs. By incorporating pre-trained models, meta-learning, and optimization strategies, we've explored innovative approaches for complex pipeline configurations. Our experiments demonstrate the framework's rapid convergence to optimal configurations across various modalities, highlighted by its performance on text-vision tasks using datasets like Flickr30k and SBU Image Captioning. The high NAUC scores and Area Under the Learning Curve (ALC) scores across 23 datasets attest to our framework's efficiency in crafting effective multimodal pipeline architectures within computational constraints in a consistent manner. Our comparisons with traditional NAS methods reveal our framework's superior efficiency, especially in time-limited scenarios, showcasing the strength of warm-starting and partial dependence on NAS along with potential areas for improvement. The framework's success in resource-limited settings indicates its *potential* applicability in real-world scenarios, when searching for well-performing architectures, laying a foundation for further research and the need for broader testing and validation in diverse environments.

Recognizing the limitations of our work, in our AutoML framework, we utilize a warm-start approach where pre-trained models are incorporated with their weights frozen. This ensures that performance variations during optimization arise solely from hyperparameter adjustments, not changes in model (pre-trained) weights. Our study focuses on how these hyperparameters impact the efficacy of a static, pre-trained model architecture. We do not fine-tune the pre-trained model weights; instead, we assess how different hyperparameter settings exploit the pre-trained models to generate and use latent representations of data across vision, text, or combined modalities. This method isolates performance variations to hyperparameter effects, ensuring clear attribution in our findings. Due to the complexity of the configuration space, this study does not examine the parameter manifold of pre-trained models, focusing instead on distinguishing between models that can still improve and those

that cannot. Future work will focus on expanding the framework's application in various settings, including sampling of parameters from parameter-spaces and further refining its capabilities to meet the evolving demands of AutoML solutions.

Appendix

A: Problem formulation

A.1: General algorithm selection problem

Let $\mathcal{A}$ be the given set of learning algorithms along with limited training data as $D = \{(x_1, y_1) \cdots (x_n, y_n)\}$. The goal is to find a pipeline $\mathcal{P}_{g,\hat{A},\hat{\lambda}}$, where $A \in \mathcal{A}$ such that $\mathcal{A}$ is a set of learning algorithms, λ is the set of corresponding hyperparameters and $g \in G$ is set of valid pipeline structures, with an optimal generalization performance. The generalization performance of such a pipeline can be evaluated by splitting D into the disjoint sets $D^{(i)}_{train}$ and $D^{(i)}_{valid}$ by applying A^* (more than one learning algorithms) to $D^{(i)}_{train}$ and evaluating the performance based on some empirical metric on $D^{(i)}_{valid}$. Here $\mathcal{L}(A, D^{(i)}_{train}, D^{(i)}_{valid})$ where $\mathcal{L} \Leftarrow\!\!\swarrow \Leftrightarrow \swarrow\!\!\Rightarrow$ is *some* loss. Thus, the general model selection problem becomes (Thornton et al., 2012; Zöller & Huber, 2019):

$$A^* \in \underset{A \in A}{\operatorname{argmin}} \frac{1}{k} \sum_{i=i}^{k} \mathcal{L}\left(A, D^{(i)}_{train}, D^{(i)}_{valid}\right) \tag{11}$$

where k represents k-fold cross validation that splits D into k equal-sized partitions[4].

A.2: Hyperparameter optimisation (HPO) problem

Let Λ be the hyperparameter space such that $\lambda \in \Lambda$ for a given algorithm A. Hyperparameter spaces are often high-dimensional and the hyperparameter λ is often continuous. Hence, given n hyperparameters $\lambda_1, \lambda_2 \cdots \lambda_n$ with domains $\Lambda_1, \Lambda_2 \cdots \Lambda_n$, the hyperparameter space Λ is a strict subset of the cross product of these domains: $\Lambda \subset \Lambda_1 \times \Lambda_2 \times \Lambda_n$ (Thornton et al., 2012). The subset of the hyperparameter space is strict, as in some settings certain hyperparameters may render others inactive. For instance, hyperparameters of a classification pre-processor should render hyperparameters of regression algorithms inactive. Hence, we can formally state this problem as:

$$\lambda^* \in \underset{\lambda \in \Lambda}{\operatorname{argmin}} \frac{1}{k} \sum_{i=1}^{k} \mathcal{L}\left(A_\lambda, D^{(i)}_{train}, D^{(i)}_{valid}\right) \tag{12}$$

[4] $D^{(i)}_{train} = D - D^{(i)}_{valid}$

B: Additional related works

B.1: Pipeline synthesis as a regression problem

In the realm of Automated Machine Learning (AutoML), the Challenge of AutoML Systems (CASH) problem, particularly the task of selecting optimal hyperparameters and pipeline configurations, has garnered substantial attention. To address this challenge, various model-based algorithms have been employed to optimize the selection process effectively. One noteworthy AutoML tool, AutoWEKA (Thornton et al., 2012), has demonstrated state-of-the-art performance in tackling the algorithm selection problem by utilizing Sequential Model-Based Optimization (SMBO) techniques, specifically Sequential Model-based Algorithm Configuration (SMAC) and Tree-structured Parzen Estimator (TPE) (Thornton et al., 2012).

The essence of solving the CASH problem can be conceptualized as a regression problem. We aim to model the relationship between hyperparameter configurations and the corresponding loss function values. This modeling process enables us to predict the most promising configurations that minimize the loss function. To achieve this, AutoWEKA and similar SMBO algorithms follow a systematic iterative approach:

1. *Model construction* The first step involves constructing a predictive model, denoted as $\mathcal{M}_L$, which captures the dependence of the loss function $\mathcal{L}$ on hyperparameter settings λ. This model serves as the cornerstone for making informed decisions about which hyperparameter configurations to evaluate.
2. *Loss evaluation* With the model $\mathcal{M}_L$ in place, the algorithm proceeds to evaluate a loss value c for a given hyperparameter and pipeline configuration λ. This evaluation is based on the loss function $\mathcal{L}$ associated with the selected configuration.
3. *Model updating* Following the loss evaluation, the algorithm updates the predictive model $\mathcal{M}_L$ with the newly acquired data point (λ, c). This iterative learning process allows the model to adapt and improve its predictions over time.
4. *Acquisition function* To select the next hyperparameter and pipeline configuration λ for evaluation, an acquisition function $a_{\mathcal{M}_L}$ is employed. This function, $a_{\mathcal{M}_L} : \Lambda \to \mathbb{R}$, quantifies the expected utility of evaluating a specific λ based on the predictive distribution of model $\mathcal{M}_L$. In essence, it helps determine which configuration is likely to yield the most valuable information.

Two widely used acquisition functions in SMBO are Sequential Model-based Algorithm Configuration (SMAC) and Tree-structured Parzen Estimator (TPE). SMAC employs the "positive expected improvement" (EI) function, which measures the improvement in performance over a given error rate threshold c_{min} for each configuration. The EI function is defined as:

$$I_{c_{min}}(\lambda) := \max\{c_{min} - c(\lambda), 0\}$$

While we may not have direct access to $c(\lambda)$, we can estimate its expectation with respect to the model $\mathcal{M}_L$. AutoWEKA often employs random forest models as $\mathcal{M}_L$ to predict μ_λ (mean) and σ^2_λ (variance) of $p(c|\lambda)$ as frequentist estimates, effectively modeling $p_{\mathcal{M}_L}$ as a Gaussian distribution $\mathcal{N}(\mu_\lambda, \sigma^2_\lambda)$. This Gaussian distribution enables the closed-form computation of the expected improvement as:

$$\mathbb{E}_{\mathcal{M}_{\mathcal{L}}}[I_{c_{min}}(\lambda)] = \sigma_{\lambda} \cdot [\mu \cdot \Phi(\mu) + \Psi(\mu)]$$

where $\mu = \frac{c_{min} - \mu_{\lambda}}{\sigma_{\lambda}}$, and Φ and Ψ represent the probability and cumulative density functions of a standard normal distribution, respectively (Thornton et al., 2012).

TPE, on the other hand, uses separate models for $p(c)$ and $p(\lambda|c)$. It distinguishes between configurations that perform well ($l(\cdot)$) and those that perform poorly ($g(\cdot)$) with respect to a chosen threshold c^*. TPE estimates the expected improvement as:

$$\mathbb{E}_{\mathcal{M}_{\mathcal{L}}}[I_{c_{min}}(\lambda)] \propto \left(\gamma + \frac{g(\lambda)}{l(\lambda)} \cdot (1 - \gamma)\right)$$

here γ is a predefined quantile value (often set to 0.15), and c^* is chosen as the γ-quantile of observed losses (Thornton et al., 2012).

Adoption in our methodology

The concept of treating pipeline synthesis as a regression problem, as demonstrated by model-based algorithms like SMAC and TPE, aligns with our approach in tackling the challenge of AutoML. In our methodology, we draw inspiration from these regression-based techniques to optimize the selection and configuration of hyperparameters and pipelines. By leveraging regression models and predictive modeling, we aim to enhance the efficiency and effectiveness of our AutoML framework, ultimately advancing the state of the art in automated machine learning.

B.2: Objective function

We provide the algorithm for the objective function constructed for warm-starting the search of pipeline configurations consisting of a pre-trained model for processing multimodal input data. The Algorithm 1 depicts the pseudo-code of the objective function used to evaluate the vision-language modality. By extending the if statements in the following function, we tackle the other two modalities. Additionally, within each modality by extending the if statements, we evaluate other pre-trained models (Albef, Data2Vec). As Data2Vec is not a multimodal model, we use Data2Vec for evaluating only the tabular + text modality. The objective function always returns a real-valued scalar value.

Algorithm 1 Algorithm for Objective Function

Let A_{ptm} be the pre-trained model, where $A_{\text{ptm}} \in \mathcal{A}$
Let $\Theta_A = \{hp_1, ..., hp_k\}$ be the hyperparameters of A_{ptm}
Let F_M be the pre-trained feature processor
Let L be the downstream model, where $L \in \mathcal{L}$
Let $\Theta_L = \{hp_{k+1}, ..., hp_{k+m}\}$ be the hyperparameters of L
Let F be the feature processor, where $F \in \mathcal{F}$
if modality is 'text_vision' **then**
  if M = 'FLAVA' **then**
    for batch in dataloader **do**
      Let (img_data, text_data) be a batch
      Let (img_vector, text_vector) = F_M(img_data, text_data)
      Let embedding = A_{ptm}(img_vector, text_vector)
      Let linear_embedding = LinearTransform(embedding)
    end for
    Let embeddings = stack(linear_embedding)
    Create a directory for the pre-trained model based on the seed
    Write the pretraining configuration to a JSON file
    Save the pretraining model state
    Create a DataFrame with embeddings and targets
    Convert embeddings to tensors and apply max pooling
    Calculate the max sequence length after max pooling
    Create a DataFrame for the pooled embeddings
    Let $(X_{\text{train}}, X_{\text{test}}, y_{\text{train}}, y_{\text{test}})$ be obtained from a hold-out split on the embedding data frame
    if downstream_task in CLF or REG **then**
      Train the fine-tuning models $l \in \mathcal{L}$ (e.g., LightGBM, XGBoost, CatBoost)
      Set hyperparameters $\Theta_l = \{hp_{k+1}, ..., hp_{k+m}\}$ based on the model type and configuration
      l.fit(X_{train}, y_{train})
      $\hat{y} = l$.predict(X_{test})
      Save the fine-tuned models and configuration
      obj_val = roc_auc($\hat{y}$, y_{test})
    end if
  end if
end if

B.3: Warm starting PTA

Previously, we provided an overview of the SMAC workflow (Fig 3) as mentioned in Sect. 4. Several, questions arise during understanding this optimisation process, we lay out some fundamental questions to provide a better understanding to the reader:

- **Q1:** What happens if the *actual* evaluation over the objective function is lower than that of the meta-learner ψ_l prediction?

Ans: Configurations that seem to perform worse than the prediction of ψ_l are tackled using a queuing strategy that is implemented by the Intensifier component of SMAC. Configurations that perform worse than predicted are added to the intensifier queue. The intensifier then chooses to re-evaluate and intensify these configurations, giving them another chance to prove their potential after the meta-learner ψ_l has been re-trained with a set of actual evaluations.

- **Q2:** How does the encoding of the configurations from the meta-dataset take place?

  Ans: Hyperparameter configurations need to be encoded into a format that can be used as input for the meta-learner ψ_l. SMAC3 provides different ways to encode configurations based on their types:

  Categorical Hyperparameters: Categorical hyperparameters are typically one-hot encoded. Each category becomes a binary feature, with one indicating the presence of the category and zero for other categories.

  Numerical Hyperparameters: Numerical hyperparameters are usually used as-is after normalization or scaling. They are represented as continuous values.

- **Q3:** How does the decoding of the configurations take place once the EI acquisition function selects a particular configuration for evaluation?

  Ans: For categorical hyperparameters that were one-hot encoded, SMAC3 identifies the active binary feature(s) in the encoded representation. These active features indicate the selected category for each categorical hyperparameter. Numerical hyperparameters, which were scaled or normalized during the encoding process, are decoded back to their original scale using the inverse of the normalization or scaling transformation.

  Once the categorical and numerical hyperparameters are decoded, SMAC3 constructs a valid configuration object. This involves setting the categorical hyperparameters to their selected categories and the numerical hyperparameters to their decoded values. Furthermore, it is possible that the decoded configuration doesn't satisfy the constraints defined by the configuration space like falling outside the allowed range. In such cases, SMAC3 applies constraint handling mechanisms to ensure that the configuration is valid. This might involve clipping the values to the valid range and rounding them.

  The Algorithm 2 provides an overview of the SMAC procedure for obtaining optimal pipeline configurations for multimodal inputs.

The Algorithm 2 provides an overview of the SMAC procedure for obtaining optimal pipeline configurations for multimodal inputs.

Algorithm 2 SMAC Optimization with Intensification

Initialize scenario $\mathcal{S}$ (includes configuration space and objective function)
Initialize meta-data M
Initialize meta-learner ψ_l (Random Forest with 3-fold CV)
Initialize intensifier $\mathcal{I}$ (Aggressive Racing)
while stopping criterion not met **do**
  if init state **then**
    for θ_i in M^{eval} **do**
      $\mu_{\theta_i}, \sigma_{\theta_i} = \psi_l$.predict($M^{eval}$)
    end for
    for $\mu_{eval}, \sigma_{eval}$ in M^{eval} **do**
      $\theta_w = \arg\max_i a_{\mathcal{M}l}(\mu_{eval}, \sigma_{eval})$
    end for
    Evaluate θ_w via objective function and get f_{θ_w}
    Update M: $M \leftarrow M \cup \{(\theta_\mathrm{w}, f_{\theta_\mathrm{w}})\}$
    Intensifier I intensifies this configuration by sampling θ^*_{next} by employing a Aggressive Racing strategy using a $[10 - 30\%]$ perturbation rate.
  else
    Sample candidate configurations $\theta_1, \theta_2, \ldots, \theta_n$ from configuration space based.
    for each candidate θ_i **do**
      Predict mean μ_i and uncertainty σ_i using surrogate model M
      Calculate acquisition function value $a_{\mathcal{M}l}(\theta_i) = \mathrm{EI}(\mu_i, \sigma_i, \mathrm{incumbent\_value})$
    end for
    Select the configuration with the highest acquisition value: $\theta_{\mathrm{next}} = \arg\max_i a_{\mathcal{M}l}(\theta_i)$
    Evaluate θ_{next} in true objective function and get $f_{\theta_{\mathrm{next}}}$
    Update M: $M \leftarrow M \cup \{(\theta_{\mathrm{next}}, f_{\mathrm{next}})\}$
    Update surrogate model ψ_l using updated run history M
    if $f_{\mathrm{next}} > f_{\mathrm{incumbent}}$ **then**
      Update incumbent: incumbent_value $\leftarrow f_{\mathrm{next}}$, $\theta_{\mathrm{incumbent}} \leftarrow \theta_{\mathrm{next}}$
    end if
    Apply intensification $\mathcal{I}$ to select configurations for true evaluations
    for each configuration θ_k selected by $\mathcal{I}$ **do**
      Evaluate θ_k in true objective function and get f_{θ_k}
      Update M: $M \leftarrow M \cup \{(\theta_k, f_{\theta_k})\}$
    end for
  end if
end while

B.4: Multimodal datasets in consideration

This appendix focuses on the analysis and selection of datasets integral to our research in multimodal machine learning, considering both unimodal and multimodal datasets. We

Dataset ID	#Train	#Test	#Cat.	#Num.	#Text	Task	Metric	Prediction Target
prod	5,091	1,273	1	0	1	multiclass	accuracy	sentiment associated with product review
salary	15,841	3,961	1	0	5	multiclass	accuracy	salary range in data scientist job listings
airbnb	18,316	4,579	37	24	28	multiclass	accuracy	price label of Airbnb listing
channel	20,284	5,071	1	15	1	multiclass	accuracy	news category to which article belongs
wine	84,123	21,031	0	2	3	multiclass	accuracy	which variety of wine (type of grape)
imdb	800	200	0	7	4	binary	roc-auc	whether film is a drama
fake	12,725	3,182	2	0	3	binary	roc-auc	whether job postings are fake
kick	86,502	21,626	3	3	3	binary	roc-auc	whether proposed Kickstarter project will achieve funding goal
jigsaw	100,000	25,000	2	27	1	binary	roc-auc	whether social media comments are toxic
qaa	4,863	1,216	1	0	3	regression	R^2	subjective type of answer (in relation to question)
qaq	4,863	1,216	1	0	3	regression	R^2	subjective type of question (in relation to answer)
book	4,989	1,248	1	2	5	regression	R^2	price of books
jc	10,860	2,715	0	2	3	regression	R^2	price of JC Penney products on their website
cloth	18,788	4,698	2	1	3	regression	R^2	customer review score for clothing item
ae	22,662	5,666	3	2	6	regression	R^2	price of American-Eagle inner-wear items on their website
pop	24,007	6,002	1	2	1	regression	R^2	online popularity of news article
house	37,951	9,488	1	18	20	regression	R^2	sale price of houses in California
mercari	100,000	25,000	3	0	6	regression	R^2	price of Mercari online marketplace products

Fig. 11 An overview of the 18 datasets proposed by Erickson et al. (2022) that form their public benchmark. '#CAT', '#NUM' and '#Text' count the categorical, numerical and text features in each of those 18 datasets. '#Train', and '#Test' show the number of training and test samples in each of those 18 datasets

delve into various publicly available datasets, assess their characteristics and relevance, and justify their inclusion in our study.

B.4.1: Multimodal datasets: tabular + text

- *Shi et al. Benchmark*: Shi et al. compiled 18 multimodal datasets (Fig. 11) focusing on tabular + text modalities, primarily derived from real-world applications. These datasets exhibit diversity in sample size, problem types, and feature categories.
- *Dataset characteristics*: Each dataset in this collection offers a mix of categorical, numerical, and text features, with varying lengths and types of text.
- *Use in research* The versatility and comprehensive nature of these datasets make them ideal for developing and testing multimodal AutoML systems, particularly for tabular + text tasks.

B.4.2: Multimodal datasets: image + text, image + text + tabular

- *Empirical analysis by Ferraro et al.* An extensive survey by Ferraro et al. (Fig. 12) presents a thorough evaluation of major vision-language datasets. They assess datasets based on various quality metrics like vocabulary size, syntactic complexity, and perplexity.
- *Datasets for vision-language tasks* We have included datasets such as VQA (Agrawal et al., 2015), SBU (Ordonez et al., 2011), Flickr30k (Plummer et al., 2015), and Flickr8k (Plummer et al., 2015) for tasks in visual question answering and image-text retrieval, aligning with our goal of generalizing over vision-language tasks.
- *Public access and empirical rigor* Despite a scarcity of datasets integrating tabular, text, and vision modalities, the PetFinder[5] dataset (focused on pet adoption prediction) and CD-18 dataset (Zehtab-Salmasi et al., 2021) (for mobile phone price prediction) stand out. These datasets provide a rich blend of multimodal data and have been utilized in AutoML frameworks like AutoGluon, underscoring their utility and relevance to our study.

[5] https://www.kaggle.com/competitions/petfinder-adoption-prediction/data.

		Size(k)				Language						Vision	
	Dataset	Img	Txt	Frazier	Yngve	Vocab Size (k)	Sent Len.	#Conc	#Abs	%Abs	Ppl	(A)bs/ (R)eal	BB
Balanced	Brown	-	52	18.5	77.21	47.7	20.82	40411	7264	15.24%	194	-	-
User-Gen	SBU	1000	1000	9.70	26.03	254.6	13.29	243940	9495	3.74%	346	R	-
	Deja	4000	180	4.13	4.71	38.3	4.10	34581	3714	9.70%	184	R	-
Crowd-sourced	Pascal	1	5	8.03	25.78	3.4	10.78	2741	591	17.74%	123	R	-
	Flickr30K	32	159	9.50	27.00	20.3	12.98	17214	3033	14.98%	118	R	-
	COCO	328	2500	9.11	24.92	24.9	11.30	21607	3218	12.96%	121	R	Y
	Clipart	10	60	6.50	12.24	2.7	7.18	2202	482	17.96%	126	A	Y
Video	VDC	2	85	6.71	15.18	13.6	7.97	11795	1741	12.86%	148	R	-
Beyond	VQA	10	330	6.50	14.00	6.2	7.58	5019	1194	19.22%	113	A/R	-
	CQA	123	118	9.69	11.18	10.2	8.65	8501	1636	16.14%	199	R	Y
	VML	11	360	6.83	12.72	11.2	7.56	9220	1914	17.19%	110	R	Y

Fig. 12 Summary statistics and quality metrics of a sample of major datasets provided by Ferraro et al. (2015)

The datasets selected for this research were chosen based on their empirical evaluation, diversity, and applicability to real-world scenarios. They provide a solid foundation for investigating and developing robust machine-learning models across various modalities and tasks.

C: Additional results

C.1: Warm-started SMAC procedure

In this subsection, we report detailed results for each of the 23 evaluated datasets. Along with the performance graphs, we also present the incumbent $\theta_{optimal}$ configurations for some of the datasets.

C.1.1: Best incumbents

Here we will list some of the best-observed pipeline configurations or incumbents $\theta_{optimal}$ for the AutoMM Benchmark (Shi et al., 2021), VQA2.0, Flickr30k, SBU, Petfinder and CD-18 datasets. The obtained incumbents as a result of our PTA framework for the AutoMM Benchmark (product sentiment and cloth), VQA2.0, Flickr30k, SBU Image Captioning, Petfinder and CD-18 Datasets are provided in Tables 4, 5, 6, 7, 8, 9, and 10 respectively.

Table 4 Best incumbent for the product machine data (AutoMM Benchmark (Shi et al., 2021))

Best incumbent θ_{optimal}	
Components and hyperparameters	Choices
Pre-training model	'Data2VecText'
Pre-training task	'Classification'
Layer normalisation ϵ	$3.799405790636786e{-}06$
Attention dropout	0.31379225175619446
Hidden droput	0.2555013087915553
Pretraining processors	'AutoTokenizer'
Linear hidden size	349
Weight decay	$1.45657361e-04$
Downstream model	'StackedEnsemble_ExtraTress_L2'
Downstream processor	'AutoMLPipelineFeatureProcessor'
Iterations	87
Max depth	2
Number of boost rounds	8
Max leaves	159

Table 5 Best incumbent for the cloth data [AutoMM Benchmark (Shi et al., 2021)]

Best incumbent θ_{optimal}	
Components and hyperparameters	Choices
Pre-training model	'Data2VecText'
Pre-training task	'Regression'
Layer normalisation ϵ	$1.2816543313722326e{-}06$
Attention dropout	0.030392025869493244
Hidden droput	0.008490381184591556
Pretraining processors	'AutoTokenizer'
Linear hidden size	344
Weight decay	$1.57842627e-03$
Downstream model	'WeightedEnsemble_r_LGB_L2'
Downstream processor	'AutoMLPipelineFeatureProcessor'
Iterations	64
Max depth	2
Number of boost rounds	7
Max leaves	128

Table 6 Best incumbent for the VQA2.0 data

Best incumbent θ_{optimal}	
Components and hyperparameters	Choices
Pre-training model	'AlbefVQA'
Pre-training task	'Classification'
, Downstream task	'Classification'
, Layer normalisation ϵ	$1.2816543313722326e{-}06$
Attention dropout	0.40062137440149
Hidden droput	0.39124685464340725
Pretraining processors	'AutoTokenizer'
Linear hidden size	438
Weight decay	$1.23562410e{-}04$
Downstream model	'WeightedEnsemble_ExtraTrees_L2'
Downstream processor	'AutoMLPipelineFeatureProcessor'
Iterations	53
Max depth	4
Number of boost rounds	6
Max leaves	138

Table 7 Best incumbent for the Flickr30k data

Best incumbent θ_{optimal}	
Components and hyperparameters	Choices
Pre-training model	'AlbefFeatureProcessor'
Pre-training task	'ITM'
Layer normalisation ϵ	$3.421060173004536e{-}06$
Attention dropout	0.40062137440149
Hidden droput	0.39124685464340725
Pretraining processors	'AutoTokenizer'
Linear hidden size	438
Weight decay	$1.45330165e{-}05$
Downstream model	'WeightedEnsemble_r_ExtraTrees_L2'
Downstream processor	'AutoMLPipelineFeatureProcessor'
Iterations	82
Max depth	4
Number of boost rounds	7
Max leaves	158

Table 8 Best incumbent for the SBU Image Captioning data

Best incumbent θ_{optimal}	
Components and hyperparameters	Choices
Pre-training model	'AlbefFeatureProcessor'
Pre-training task	'ITM'
Layer normalisation ϵ	$1.2268910169581294e{-}06$
Attention dropout	0.32449508667965
Hidden droput	0.008490381184591556
Pretraining processors	'AutoTokenizer'
Linear hidden size	517
Weight decay	$3.56890021e{-}04$
Downstream model	'StackedEnsemble_r_CAT_L1'
Downstream processor	'AutoMLPipelineFeatureProcessor'
Iterations	93
Max depth	2
Number of boost rounds	7
Max leaves	153

Table 9 Best incumbent for the PetFinder data

Best incumbent θ_{optimal}	
Components and hyperparameters	Choices
Pre-training model	'FlavaFeatureProcessor'
Pre-training task	'Classification'
Layer normalisation ϵ	$5e{-}06$
Attention dropout	0.2493445102
Hidden droput	0.25
Pretraining processors	'FlavaProcessor'
Linear hidden size	456
Weight decay	$1e{-}05$
Downstream model	'WeightedEnsemble_L2'
Downstream processor	'AutoMLPipelineFeatureProcessor'
Iterations	150
Max depth	5
Number of boost rounds	8
Max leaves	128

Table 10 Best incumbent for the CD-18 data

Best incumbent θ_{optimal}	
Components and hyperparameters	Choices
Pre-training model	'AlbefFeatureProcessor'
Pre-training task	'ITM'
Layer normalisation ϵ	$1.532967907935182e{-}06$
Attention dropout	0.18588839010002423
Hidden droput	0.23541206938552428
Pretraining processors	'AutoTokenizer'
Linear hidden size	389
Weight decay	$2.4674356e{-}04$
Downstream model	'WeightedEnsemble_r_RF_L1'
Downstream processor	'AutoMLPipelineFeatureProcessor'
Iterations	93
Max depth	2
Number of boost rounds	9
Max leaves	268

C.2: Prior evaluations

In this sub-section of the Appendix, we will provide detailed results of our experimentation. Tables 11, 12, 13, 14, 15, 16, and 17 enlists the obtained AUC scores over the 18 AutoMM benchamrk datasets released by Shi et al. (2021). Prior Evaluations over the

Table 11 Pipeline evaluations comprising of FlavaTextModel, Data2VecTextModel and several downstream models, on the *channel* and *pop* data from the AutoMM Benchmark

	FlavaTextModel			Data2VecTextModel		
	Score	Pred_time (s)	Fit_time (s)	Score	Pred_time (s)	Fit_time (s)
Data: channel						
WeightedEnsemble_L3*	0.442	26.105	590.098	0.437	57.330	2658.636
LightGBM_BAG_L2	0.440	13.692	346.585	0.437	30.691	2347.787
RFMSE_BAG_L2	0.439	12.055	270.667	0.436	28.294	1324.257
WeightedEnsemble_L2	0.434	1.480	73.270	0.435	2.301	1014.782
LightGBM_BAG_L1	0.396	24.458	510.932	0.359	54.925	1631.612
LightGBM_BAG_L2	0.357	10.568	195.668	0.318	25.988	307.675
Data: pop						
StackedEnsemble_L2	0.423	176.564	1278.900	0.446	486.345	2460.963
WeightedEnsemble_L3	0.420	183.786	1387.634	0.443	434.231	5478.935
LightGBMXT_BAG_L1	0.416	89.783	3416.620	0.432	256.780	3462.446
StackedEnsemble_L2	0.384	273.678	2461.119	0.406	315.889	6365.458
LightGBMXT_BAG_L2	0.377	32.489	6258.921	0.927	371.167	11,467.700
RFMSE_BAG_L1	0.362	107.512	3844.022	0.927	590.493	18,794.570
XGBoost*	0.358	1152.318	14,604.284	0.928	315.603	10,984.844
LightGBM_BAG_L2	0.372	1249.579	17,855.008	0.927	3.134	1133.980

Score represents the NAUC score for the classification tasks and R^2 for the regression tasks

Table 12 Pipeline evaluations comprising of FlavaTextModel, Data2VecTextModel and several downstream models, on the *product* and *salary* data from the AutoMM Benchmark

	FlavaTextModel			Data2VecTextModel		
	Score	Pred_time (s)	Fit_time (s)	Score	Pred_time (s)	Fit_time (s)
Data: product						
WEns_L2⋆	0.258	34.223	6094.538	0.3673	63.955	3890.093
ETMSE_BAG_L1⋆	0.288	13.287	77.712	0.369	87.545	103.432
WEns_L3⋆	0.256	1128.574	20,340.757	0.367	1061.557	18,765.396
RFMSE_BAG_L1	0.290	266.354	920.093	0.350	330.742	881.680
LightGBM_BAG_L2	0.281	574.719	19,221.342	0.321	617.396	22,389.342
XT_BAG_L2	0.260	574.879	1266.841	0.349	620.545	1381.093
RFMSE_BAG_L2	0.258	847.312	17,277.004	0.345	728.986	16,452.985
ETMSE_BAG_L2	0.257	835.327	16,345.931	0.357	745.634	18,437.789
Data: salary						
WEns_L2⋆	0.423	43.189	731.896	0.413	59.380	643.140
WEns_L3⋆	0.439	87.261	1480.172	0.431	123.986	1874.641
LightGBM_BAG_L2	0.426	46.520	975.922	0.421	120.820	1557.680
ETMSE_BAG_L2	0.428	83.923	1233.944	0.422	65.590	1229.150
LightGBM_BAG_L1	0.349	3.138	237.770	0.360	3.069	290.304
XT_BAG_L2	0.360	40.040	493.010	0.411	62.453	934.743

Score represents the NAUC score for the classification tasks and R^2 for the regression tasks

Table 13 Pipeline evaluations comprising of FlavaTextModel, Data2VecTextModel and several downstream models, on the *cloth* and *wine* data from the AutoMM Benchmark

	FlavaTextModel			Data2VecTextModel		
	Score	Pred_time (s)	Fit_time (s)	Score	Pred_time (s)	Fit_time (s)
Data: cloth						
LightGBM_BAG_L2	0.599	8.354	4244.582	0.562	20.46	684.204
ExtraTreesMSE_BAG_L1	0.566	105.487	46.383	0.569	287.024	4339.216
RandomForestMSE_BAG_L1	0.577	104.184	46.566	0.590	130.421	7886.433
WeightedEnsemble_L2⋆	0.621	218.020	4339.216	0.659	163.955	2970.863
LightGBM_BAG_L2	0.594	236.360	7887.643	0.594	217.696	5798.055
Data: wine						
ExtraTreesMSE_BAG_L2⋆	0.562	7.744	7859.070	0.512	234.449	1259.750
WeightedEnsemble_L3⋆	0.566	105.785	7744.040	0.531	189.521	1784.941

Score represents the AUC score for the classification tasks and R^2 for the regression tasks

Table 14 Pipeline evaluations comprising of FlavaTextModel, Data2VecTextModel and several downstream models, on the fake job and imdb data

	FlavaTextModel			Data2VecTextModel		
	Score	Pred_time (s)	Fit_time (s)	Score	Pred_time (s)	Fit_time (s)
Data: fake job						
StackedEnsemble_L2	0.237	176.876	1345.874	0.267	189.980	2456.986
LightGBMXT_BAG_L1	0.228	223.867	9821.567	0.253	176.523	1345.789
LightGBM_CAT_L2	0.213	234.879	7689.645	0.253	183.45	1298.563
RandomForestMSE_BAG_L2	0.210	456.345	12398.756	0.219	231.096	4562.512
CatBoost_BAG_L1	0.198	112.534	1246.453	0.214	124.466	5679.948
LightGBM_BAG_L2	0.187	7.080	2152.598	0.197	474.826	4586.376
RandomForestMSE_BAG_L1	0.177	1560.574	46.566	0.124	136.471	844.172
ExtraTreesMSE_BAG_L1	0.162	2432.172	46.383	0.121	287.762	2735.361
Data: imdb						
WeightedEnsemble_L2	0.346	125.384	12450.477	0.198	433.761	16321.556
WeightedEnsemble_L3	0.332	237.344	10485.536	0.192	200.60	14575.254
LightGBM_BAG_L2	0.315	119.342	5222.209	0.137	217.728	12671.667
StackedEnsemble_L2	0.313	123.756	1346.266	0.121	127.041	7269.032
ExtraTreesMSE_BAG_L2	0.309	280.233	2246.343	0.214	167.724	5679.948

Score represents the NAUC score for the classification tasks and R^2 for the regression tasks. Prediction and fit times are mentioned in seconds

Table 15 Pipeline evaluations comprising of FlavaTextModel, Data2VecTextModel and several downstream models, on the jigsaw and kick data

	FlavaTextModel			Data2VecTextModel		
	Score	Pred_time (s)	Fit_time (s)	Score	Pred_time (s)	Fit_time (s)
Data: jigsaw						
StackedEnsemble_L2	0.367	567.345	1257.567	0.456	163.475	2345.987
LightGBMXT_BAG_L1	0.358	443.973	2542.292	0.453	334.856	1245.846
LightGBM_CAT_L2	0.342	354.567	4563.236	0.345	464.353	1298.563
RandomForestMSE_BAG_L2	0.339	678.398	945.836	0.334	635.447	3866.335
WeightedEnsemble_CAT_L2	0.315	234.678	778.953	0.302	402.465	4735.255
WeightedEnsemble_RF_L2	0.308	134.578	1356.642	0.264	223.577	3756.375
StackedEnsemble_CAT_L1	0.288	421.934	889.755	0.234	345.684	2445.467
ExtraTreesMSE_CAT_L1	0.263	643.945	1534.673	0.211	416.788	3453.344
Data: kick						
WeightedEnsemble_L2	0.346	125.384	12450.477	0.198	433.761	16321.556
WeightedEnsemble_L3	0.332	237.344	10485.536	0.192	200.60	14575.254
LightGBM_BAG_L2	0.315	119.342	5222.209	0.137	217.728	12671.667
StackedEnsemble_L2	0.313	123.756	1346.266	0.121	127.041	7269.032
StackedEnsemble_L3	0.309	280.233	2246.343	0.214	167.724	5679.948

Score represents the NAUC score for the classification tasks and R^2 for the regression tasks. Prediction and fit times are mentioned in seconds

Table 16 Pipeline evaluations comprising of FlavaTextModel, Data2VecTextModel and several downstream models, on the *ae* and *book* data from the AutoMM Benchmark

	FlavaTextModel			Data2VecTextModel		
	Score	Pred_time (s)	Fit_time (s)	Score	Pred_time (s)	Fit_time (s)
Data: ae						
WeightedEnsemble_L3	0.378	456.382	2345.746	0.367	1834.446	18765.396
ExtraTreesMSE_BAG_L2	0.374	366.846	1242.345	0.334	889.345	4336.235
RandomForestMSE_BAG_L2	0.369	235.734	1278.345	0.331	654.234	6432.985
StackedEnsemble_CAT_L2	0.334	443.345	5673.235	0.323	555.345	3560.4460
LightGBMXT_BAG_L2	0.306	123.566	4567.221	0.349	445.345	4561.233
LightGBM_BAG_L2	0.298	235.435	3456.234	0.321	334.624	2339.244
StackedEnsemble_CAT_L1	0.287	1092.456	920.093	0.250	354.574	1821.380
WeightedEnsemble_L2	0.244	334.563	1982.346	0.249	123.567	3102.352
Data: book						
WeightedEnsemble_L3	0.493	100.556	1480.172	0.491	323.986	1874.641
ExtraTreesMSE_BAG_L2	0.488	283.923	1233.944	0.483	353.590	1229.150
StackedEnsemble_XT_L1	0.475	264.234	975.922	0.461	120.820	1557.680
WeightedEnsemble_L2	0.403	173.240	731.896	0.444	59.380	643.140
StackedEnsemble_CAT_L2	0.374	234.318	3493.010	0.431	62.453	934.743
LightGBM_BAG_L1	0.345	3.138	127.340	0.330	3.069	290.304

Score represents the NAUC score for the classification tasks and R^2 for the regression tasks

Table 17 Pipeline evaluations comprising of FlavaTextModel, Data2VecTextModel and several downstream models, on the *california* and *jcpenny* data from the AutoMM Benchmark

	FlavaTextModel			Data2VecTextModel		
	Score	Pred_time (s)	Fit_time (s)	Score	Pred_time (s)	Fit_time (s)
Data: california						
WeightedEnsemble_L3	0.434	567.867	2345.677	0.369	373.476	2546.578
ExtraTreesMSE_BAG_L2	0.413	563.467	1346.344	0.342	383.387	4567.788
RandomForestMSE_BAG_L2	0.367	445.346	2356.352	0.303	476.336	2345.567
StackedEnsemble_CAT_L2	0.324	134.578	2654.335	0.234	373.487	2344.557
LightGBMXT_BAG_L2	0.298	108.323	3227.663	0.224	336.443	7835.397
LightGBM_BAG_L2	0.264	87.432	4635.334	0.219	483.457	3365.386
StackedEnsemble_CAT_L1	0.243	92.664	3345.356	0.206	263.486	3753.263
WeightedEnsemble_L2	0.241	512.423	3354.325	0.192	234.632	4342.346
Data: jcpenny						
WeightedEnsemble_L3	0.287	288.387	3245.886	0.345	645.346	2573.475
ExtraTreesMSE_BAG_L2	0.261	344.398	2353.466	0.334	373.873	2763.397
StackedEnsemble_XT_L1	0.234	374.578	975.922	0.261	465.678	2357.340
WeightedEnsemble_L2	0.403	173.240	731.896	0.242	444.443	3433.443
StackedEnsemble_CAT_L2	0.374	234.318	2493.010	0.232	232.456	3454.563
LightGBM_BAG_L1	0.345	334.245	3366.3365	0.231	573.498	4570.451
WeightedEnsemble_CAT_L1	0.374	344.463	7456.338	0.201	9466.345	10934.743
LightGBM_BAG_L1	0.345	454.304	1345.332	0.198	836.373	9266.498
StackedEnsemble_XT_L2	0.374	387.345	3873.345	0.169	483.354	9864.836
LightGBM_BAG_L1	0.345	377.498	8763.448	0.127	374.356	8863.235

Score represents the NAUC score for the classification tasks and R^2 for the regression tasks

Flickr30k, SBU Image Captioning, VQA2.0, Petfinder and CD-18 Datasets are provided in Tables 18, 19, 20, and 21 respectively.

Table 18 Pipeline evaluations comprising of FlavaModel, AlbefModel and several downstream models, on the Flickr30k data

	FlavaITM			AlbefITM		
	Score	Pred_time (s)	Fit_time (s)	Score	Pred_time (s)	Fit_time (s)
Data: Flickr30k						
WeightedEnsemble_L2	0.187	339.057	11372.849	0.212	429.275	15428.518
LightGBMXT_BAG_L2	0.174	319.159	10485.536	0.212	20.460	11155.774
LightGBM_BAG_L2	0.190	59.803	5222.209	0.213	217.696	14576.774
RandomForestMSE_BAG_L1	0.233	10605.230	46.566	0.214	130.421	7233.592
ExtraTreesMSE_BAG_L1	0.253	8.246	46.383	0.215	287.024	4979.938
LightGBM_BAG_L2	0.209	7.080	5212.965	0.218	217.696	2426.696
RandomForestMSE_BAG_L1	6.082	2056.751	46.566	0.223	130.421	823.182
ExtraTreesMSE_BAG_L1	0.225	2432.172	46.383	0.260	50.217	724.991

Score represents the NAUC score for the classification tasks and R^2 for the regression tasks. Prediction and fit times are mentioned in seconds

Table 19 Pipeline evaluations comprising of FlavaModel, AlbefModel and several downstream models, on the SBU Image Captioning data

	FlavaITM			AlbefITM		
	Score	Pred_time (s)	Fit_time (s)	Score	Pred_time (s)	Fit_time (s)
Data: SBU						
WeightedEnsemble_L2	0.156	345.687	12450.477	0.198	433.761	16321.556
LightGBMXT_BAG_L2	0.132	327.863	10485.536	0.192	200.60	14575.254
LightGBM_BAG_L2	0.152	69.422	5222.209	0.137	217.728	12671.667
RandomForestMSE_BAG_L1	0.134	12398.756	46.566	0.121	127.041	7269.032
ExtraTreesMSE_BAG_L1	0.198	80.462	46.383	0.214	167.724	5679.948
LightGBM_BAG_L2	0.1388	7.080	2152.598	0.198	474.826	4586.376
RandomForestMSE_BAG_L1	0.162	1560.574	46.566	0.222	266.471	934.172
ExtraTreesMSE_BAG_L1	0.202	2432.172	46.383	0.201	87.762	798.541

Score represents the NAUC score for the classification tasks and R^2 for the regression tasks. Prediction and fit times are mentioned in seconds

Table 20 Pipeline evaluations comprising of FLAVA, ALBEF and several downstream models, on the VQA2.0 dataset

	FlavaVQA			AlbefVQA		
	Score	Pred_time (s)	Fit_time (s)	Score	Pred_time (s)	Fit_time (s)
Data: VQA2.0						
LightGBMXT_BAG_L2	0.862	3657.670	4494.950	0.952	6332.752	7116.29
WeightedEnsemble_L3	0.862	3657.700	4494.900	0.952	6333.195	7133.968
LightGBM_BAG_L2	0.843	3656.579	2577.151	0.945	5709.193	7560.163
LightGBMXT_BAG_L1	0.841	7.526	635.346	0.943	5723.687	8440.646
WeightedEnsemble_L2	0.841	7.556	639.464	0.942	1444.840	1437.269
LightGBM_BAG_L1	0.841	7.896	722.845	0.941	715.837	147.628
KNeighborsUnif_BAG_L1	0.826	108.743	43.880	0.941	705.919	142.511
ExtraTreesEntr_BAG_L1	0.813	852.282	65.708	0.939	1304.944	31.675
LightGBM_BAG_L2	0.843	3656.579	2577.151	0.939	1315.104	36.180
LightGBMXT_BAG_L1	0.841	7.526	635.346	0.926	619.605	386.650
RandomForestEntr_BAG_L1	0.813	843.022	166.568	0.926	680.374	543.421

The score represents the NAUC score for the classification tasks and R^2 for the regression tasks

Table 21 Pipeline evaluations comprising of FLAVA, ALBEF and several downstream models, on the *Petfinder* and *CD-18* data

	FLAVA			Albef		
	Score	Pred_time (s)	Fit_time (s)	Score	Pred_time (s)	Fit_time (s)
Data: PetFinder						
WeightedEnsemble_L2	0.390	135.525	1360.635	0.398	243.831	1121.246
LightGBMXT_BAG_L2	0.386	237.567	1283.252	0.389	220.42	1566.822
LightGBM_BAG_L2	0.384	269.865	442.403	0.373	113.528	1633.654
CatBoost_BAG_L1	0.381	238.567	436.634	0.366	127.241	2769.032
ExtraTreesMSE_BAG_L1	0.372	280.332	446.433	0.364	167.724	1739.484
LightGBM_BAG_L2	0.356	127.370	352.690	0.359	554.525	1586.267
RandomForestMSE_BAG_L1	0.334	1560.574	46.566	0.222	266.471	934.172
ExtraTreesMSE_BAG_L1	0.329	2432.172	46.383	0.201	87.762	798.541
Data: CD-18						
StackedEnsemble_L2	0.423	176.564	778.90	0.446	186.345	460.983
WeightedEnsemble_L3	0.420	183.786	887.634	0.443	434.231	478.985
LightGBMXT_BAG_L1	0.416	89.783	516.620	0.432	256.78	562.456
StackedEnsemble_L2	0.384	273.678	961.119	0.406	315.889	765.458
LightGBMXT_BAG_L2	0.377	324.678	630.456	0.390	123.967	672.107
LightGBM_BAG_L2	0.372	1249.579	765.678	0.388	334.102	706.431
ExtraTreesMSE_BAG_L1	0.358	123.459	456.782	0.325	167.674	956.383

Score represents the NAUC score for the classification tasks and R^2 for the regression tasks

Author contributions J.V. and P.S. conceived of the presented idea. A.M. developed the methodology and performed the implementations and computations. A.M., J.V., and P.S. verified the analytical methods. All authors discussed the results and contributed to the final manuscript. D.T. provided oversight of the writing process and grounding for future extensions of this work.

Funding This work was supported in part by TAILOR, a project funded by the EU Horizon 2020 research and innovation programme under Grant No. 952215. The authors would like to acknowledge the support of J. Vanschoren in this regard.

Data availability All datasets included in this work are public datasets, ensuring transparency and accessibility.

Code availability All code included in this work is available on GitHub at https://github.com/a-moharil/Pre-Trained-Transformer-Based-AutoML.git.

Declarations

Conflict of interest The authors declare no Conflict of interest. The institutions involved in this research are '@tue.nl' and '@jads.nl.

Ethical approval Not Applicable. This research did not involve human participants, their data, or animals.

Consent to participate Not Applicable. This research did not involve human participants.

Consent for publication Not Applicable. This manuscript does not contain any individual person's data in any form.

References

Agrawal, A., Lu, J., Antol, S., Mitchell, M., Zitnick, C. L., Batra, D., & Parikh, D. (2015). VQA: Visual question answering. *ArXiv*. https://doi.org/10.48550/ARXIV.1505.00468

Baevski, A., Hsu, W.-N., Xu, Q., Babu, A., Gu, J., & Auli, M. (2022). data2vec: A General framework for self-supervised learning in speech, vision and language. *arXiv*https://doi.org/10.48550/ARXIV.2202.03555. *https://arxiv.org/abs/2202.03555*

Barrett, L. F. (2017). *How emotions are made: The secret life of the brain*. Houghton Mifflin Harcourt. https://books.google.nl/books?id=hN8MBgAAQBAJ

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv*. https://doi.org/10.48550/ARXIV.1810.04805

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2020). An image is worth 16×16 words: Transformers for image recognition at scale. *arXiv*. https://doi.org/10.48550/ARXIV.2010.11929

Du, Y., Liu, Z., Li, J., & Zhao, W. X. (2022). A survey of vision-language pre-trained models. *arXiv*. https://doi.org/10.48550/ARXIV.2202.10936

Elsken, T., Metzen, J. H., & Hutter, F. (2018). Neural architecture search: A survey. *Journal of Machine Learning Research*. https://doi.org/10.48550/ARXIV.1808.05377

Erickson, N., Shi, X., Sharpnack, J., & Smola, A. (2022). Multimodal automl for image, text and tabular data. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining. KDD '22* (pp. 4786–4787). Association for Computing Machinery. https://doi.org/10.1145/3534678.3542616

Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M., & Smola, A. (2020). AutoGluon-tabular: Robust and Accurate AutoML for structured data. *arXiv*. https://doi.org/10.48550/ARXIV.2003.06505

Ferraro, F., Mostafazadeh, N., Huang, T.K., Vanderwende, L., Devlin, J., Galley, M., & Mitchell, M. (2015). A survey of current datasets for vision and language research. In L. Màrquez, C. Callison-Burch, J. Su, D. Pighin, Y. Marton (Eds) *Proceedings of the 2015 conference on empirical methods in natural language processing, EMNLP 2015, Lisbon, Portugal, September 17–21, 2015* (pp. 207–213). The Association for Computational Linguistics. https://doi.org/10.18653/v1/d15-1021

Feurer, M., Eggensperger, K., Falkner, S., Lindauer, M., & Hutter, F. (2020). Auto-sklearn 2.0: Hands-free automl via meta-learning. https://doi.org/10.48550/ARXIV.2007.04074

Hospedales, T., Antoniou, A., Micaelli, P., & Storkey, A. (2020). Meta-learning in neural networks: A survey. *arXiv*. https://doi.org/10.48550/ARXIV.2004.05439

Hutter, F., Kotthoff, L., & Vanschoren, J. (2019). *Automated machine learning: Methods, systems, challenges* (1st ed.). Springer.

Khan, S. H., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., & Shah, M. (2021) Transformers in vision: A survey. CoRR arxiv:2101.01169

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2019). ALBERT: A lite BERT for self-supervised learning of language representations. *arXiv*. https://doi.org/10.48550/ARXIV.1909.11942

Liang, P. P., Lyu, Y., Fan, X., Wu, Z., Cheng, Y., Wu, J., Chen, L., Wu, P., Lee, M. A., Zhu, Y., Salakhutdinov, R., & Morency, L. (2021). Multibench: Multiscale benchmarks for multimodal representation learning. In J. Vanschoren, & S. Yeung (Eds.) *Proceedings of the neural information processing systems track on datasets and benchmarks 1, NeurIPS datasets and benchmarks 2021, December 2021, Virtual*https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/37693cfc748049e45d87b8c7d8b9aacd-Abstract-round1.html

Li, J., Selvaraju, R. R., Gotmare, A. D., Joty, S., Xiong, C., & Hoi, S. (2021). Align before fuse: Vision and language representation learning with momentum distillation. *arXiv*. https://doi.org/10.48550/ARXIV.2107.07651

Liu, H., Simonyan, K., & Yang, Y. (2018). DARTS: Differentiable architecture search. *arXiv*https://doi.org/10.48550/ARXIV.1806.09055 . *https://arxiv.org/abs/1806.09055*

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach. *arXiv, 1*, 1. https://doi.org/10.48550/ARXIV.1907.11692

Liu, Z., Pavao, A., Xu, Z., Escalera, S., Ferreira, F., Guyon, I., Hong, S., Hutter, F., Ji, R., Júnior, J. C. S. J., Li, G., Lindauer, M., Luo, Z., Madadi, M., Nierhoff, T., Niu, K., Pan, C., Stoll, D., Treguer, S., … Zhang, Y. (2021). Winning solutions and post-challenge analyses of the ChaLearn AutoDL challenge 2019. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 43*(9), 3108–3125. https://doi.org/10.1109/TPAMI.2021.3075372

Li, L. H., Yatskar, M., Yin, D., Hsieh, C.-J., & Chang, K.-W. (2019). VisualBERT: A simple and performant baseline for vision and language. *arXiv*. https://doi.org/10.48550/ARXIV.1908.03557

Li, X., Yin, X., Li, C., Zhang, P., Hu, X., Zhang, L., Wang, L., Hu, H., Dong, L., Wei, F., Choi, Y., & Gao, J. (2020). Oscar: Object-semantics aligned pre-training for vision-language tasks. *arXiv*. https://doi.org/10.48550/ARXIV.2004.06165

Nguyen, P., Hilario, M., & Kalousis, A. (2014). Using meta-mining to support data mining workflow planning and optimization. *J. Artif. Intell. Res., 51*, 605–644. https://doi.org/10.1613/jair.4377

Olson, R. S., & Moore, J. H. (2016). Evaluation of a tree-based pipeline optimization tool for automating data science. In *Proceedings of the genetic and evolutionary computation conference 2016* (pp. 485–492). ACM.

Ordonez, V., Kulkarni, G., & Berg, T.L. (2011). Im2text: Describing images using 1 million captioned photographs. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. C. N. Pereira, K. Q. Weinberger (Eds.) *Advances in neural information processing systems 24: 25th annual conference on neural information processing systems 2011. Proceedings of a meeting held 12–14 December 2011, Granada, Spain* (pp. 1143–1151). https://proceedings.neurips.cc/paper/2011/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html

Öztürk, E., Ferreira, F., Jomaa, H., Schmidt-Thieme, L., Grabocka, J., & Hutter, F. (2022). Zero-shot AutoML with pretrained models. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, & S. Sabato (Eds.) *Proceedings of the 39th international conference on machine learning. Proceedings of machine learning research* (Vol. 162, pp. 17138–17155). PMLR. https://proceedings.mlr.press/v162/ozturk22a.html

Plummer, B. A., Wang, L., Cervantes, C. M., Caicedo, J. C., Hockenmaier, J., & Lazebnik, S. (2015). Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. *arXiv*. https://doi.org/10.48550/ARXIV.1505.04870

Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., & Huang, X. (2020). Pre-trained models for natural language processing: A survey. *Science China Technological Sciences, 63*(10), 1872–1897. https://doi.org/10.1007/s11431-020-1647-3

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., & Sutskever, I. (2021). Learning transferable visual models from natural language supervision. *arXiv*. https://doi.org/10.48550/ARXIV.2103.00020

Shi, X., Mueller, J., Erickson, N., Li, M., & Smola, A. J. (2021). Benchmarking multimodal AutoML for tabular data with text fields. *arXiv*. https://doi.org/10.48550/ARXIV.2111.02705

Singh, A., Hu, R., Goswami, V., Couairon, G., Galuba, W., Rohrbach, M., & Kiela, D. (2021). FLAVA: A foundational language and vision alignment model. *arXiv*. https://doi.org/10.48550/ARXIV.2112.04482

Su, W., Zhu, X., Cao, Y., Li, B., Lu, L., Wei, F., & Dai, J. (2019). VL-BERT: Pre-training of generic visual-linguistic representations. *arXiv, 1*, 1. https://doi.org/10.48550/ARXIV.1908.08530

Tan, H., & Bansal, M. (2019). LXMERT: Learning cross-modality encoder representations from transformers. *arXiv*. https://doi.org/10.48550/ARXIV.1908.07490

Thornton, C., Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2012). Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. *arXiv*. https://doi.org/10.48550/ARXIV.1208.3719

Van Ackeren, M. J., Barbero, F. M., Mattioni, S., Bottini, R., & Collignon, O. (2018). Neuronal populations in the occipital cortex of the blind synchronize to the temporal dynamics of speech. *eLife, 7*, 31640. https://doi.org/10.7554/eLife.31640

Wistuba, M., Rawat, A., & Pedapati, T. (2019). A survey on neural architecture search. *IBM Research AI*arXiv:1905.01392 *[cs.LG].*

Zehtab-Salmasi, A., Feizi-Derakhshi, A.-R., Nikzad-Khasmakhi, N., Asgari-Chenaghlu, M., & Nabipour, S. (2021). Multimodal price prediction. *Annals of Data Science, 10*(3), 619–635.

Zöller, M.-A., & Huber, M. F. (2019). Benchmark and survey of automated machine learning frameworks. *Journal of Artificial Intelligence Research*. https://doi.org/10.48550/ARXIV.1904.12054