Department of Mathematics and Computer Science
Artificial Intelligence and Data Engineering Lab

Automatic Synthesis of Machine Learning Pipelines consisting of Pre-Trained Models for Multimodal Data

Master Thesis

Ambarish Moharil

Supervisor:
Prof. Dr. Joaquin Vanschoren

Eindhoven, August 2023

Preface

One of the major goals of Automated Machine Learning (AutoML) is to democratize the field of Machine Learning (ML) and Deep Learning (DL). The process of constructing well-performing ML architectures is quite complex and opaque. This complexity can hinder the implementation of ML techniques for individuals lacking statistical and technical expertise in the field. The past decade has witnessed tremendous interest in this field, and several ML solutions have been proposed to address various industrial and social problems. As such, ML becomes inevitable for tackling these issues. With more people being drawn towards implementing ML solutions, having a guiding framework becomes important for proposing solutions in an informed manner. Furthermore, proficient data scientists possess a vast amount of statistical knowledge, yet they often lack in-depth domain expertise across various fields. Bridging this gap requires relying on domain experts who can provide a deep understanding of their respective fields. However, the transfer of knowledge can be insufficient, and in certain scenarios, domain experts themselves might benefit from creating systematic ML-based solutions to address problems in their field. It could be valuable to seek validation of these solutions from expert data scientists, rather than solely relying on their expertise for generating solutions.

AutoML can be a powerful tool in realizing this goal, enabling various domain experts to automatically generate more informed ML-based solutions. This tool can contribute to a fair democratization of the field. Furthermore, as advancements in the field of DL continue, AutoML needs to keep up with this progress to generate more reliable solutions. Despite the extensive research in AutoML, it remains unclear how AutoML systems can effectively incorporate pre-trained models for multimodal signals. Our research aims to address this knowledge gap. Transformer-based models have gained significant traction since their inception, mainly due to their adaptability to different domains. AutoML can be extended to include various pre-trained Transformer-based architectures, rather than solely relying on expensive Neural Architecture Search (NAS) methods for generating pipelines.

This shift could be pivotal in achieving a comprehensive expansion of the field and the insights gained could assist experts in further studying the obtained results to realize more general solutions.

Abstract

Machine Learning (ML)-based systems have become indispensable, finding extensive applications across diverse industries. Construction of a robust ML-based system demands a profound understanding of the domain and data, necessitating expert knowledge. This intricate process of pipeline synthesis within an ML system thus specifically requires the collaborative effort of both skilled data scientists and domain experts. **Automated Machine Learning** (AutoML) has shown that it is possible to automatically build extensive end-to-end ML pipelines that span numerous tasks without human intervention. Despite the substantial research conducted in the field of AutoML, there is still a lack of clarity regarding the ability of AutoML systems to create end-to-end ML pipelines for multimodal data without heavily relying on Neural Architecture Search (NAS) based methods. Additionally, there's a question of whether these systems can avoid designing architectures entirely from scratch for tasks involving both multimodal as well as unimodal data. Our research centers around crafting pipelines for multimodal inputs through the utilization of pre-trained multimodal models, complemented by a constrained degree of NAS. The goal of this research is to explore the integration of pre-trained models in AutoML systems for performing AutoML over multimodal data. We show that it is possible to automatically generate **universal** embeddings by incorporating multimodal pre-trained Transformer models in our AutoML system, which avoids a complete reliance on NAS methods that often tend to be very expensive. Furthermore, we demonstrate that using a *meta-learning* based algorithm for initiating the search of a Bayesian Optimisation (BO) process yields comparable, and at times, superior performance compared to conventional Neural Architecture Search (NAS) methods. This approach proves to be more efficient in automatically generating end-to-end Machine Learning (ML) pipelines when dealing with multimodal input data under a given budget.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Challenges

In the last decade, Machine Learning (ML) has gained tremendous importance. ML systems have become ubiquitous and are used to perform complex tasks such as speech recognition, autonomous driving and predictive maintenance of industrial equipment. These ML systems have become an integral part of the data-driven process that drives Industry 4.0 [49]. In order to model or solve these highly intricate tasks, the synthesis of highly complex ML pipelines is required. Usually, it takes a team of human ML and domain experts to come up with complex end-to-end pipelines for a given task. With the aid of the statistical knowledge possessed by data scientists and the domain knowledge acquired by the domain experts, it becomes possible to realize intricate pipelines consisting of relevant pre-processors, feature engineering algorithms and fine-tuned models. Thus, building intricate ML pipelines is a resource-consuming process. The major challenge in this process is to find relevant pre-processing and predictive models that generalize well over a domain. Furthermore, it becomes challenging to find the relevant set of hyperparameters, in a *guided* fashion such that the end-to-end pipeline *optimises well* over the data being modelled.

AutoML possesses the power to automatically come up with complex pipeline structures and mechanisms for finding the best hyperparameters suited for a specific task through processes such as hyperparameter optimization (HPO). In the past ten years, several AutoML pipeline synthesis tools have been created. These tools allow experts in specific fields to create complex pipelines without solely depending on data scientists. Current AutoML systems can be categorized into two main categories: 1) Systems that

build the neural networks but do not focus on optimizing the pre-processing [27, 8]. 2) Systems that learn machine learning pipelines along with the pre-processing but can't build neural networks as efficiently [44, 9, 13].

Even though we have benchmark pipeline generation tools, most of them are restricted to either unimodal inputs or inputs pertaining to only a specific set of modalities like Tabular + Text [39, 10]. It still remains unclear what design choices and architectures *best* [2] handle multimodal inputs and in what way these models can fuse information and can be implemented in an end-to-end ML pipeline to generate universal representations or embeddings. One can design infinitely many architectures based on the concepts of model fusion [47, 36], by *fusing* information originating from different sources (modalities) for generating embeddings that contain contextual information and relationships of the multimodal data. However, finding such architectures *manually* is quite a complex task. AutoML can be used to automate this task using techniques such as Neural Architecture Search (NAS), but as stated above, these techniques rarely focus on optimizing the pre-processing of the input signals. Although NAS methods achieve impressive performance, implementing full NAS methods along with pipeline synthesis can turn out to be computationally expensive [8]. Hence, the identification of motifs that lead towards better performance becomes necessary. Understanding why such motifs lead towards high performances and investigating whether these motifs generalize over different problems becomes a desirable property.

Pre-trained models often allow us to obtain strong performance faster (for a given task) than training deep neural networks from scratch [33]. In the domain of Natural Language Processing and Computer Vision, a better solution, especially under low resource constraints is typically found by fine-tuning a pre-trained deep neural model [37, 7]. Research conducted by Van Ackeren et al. [45] in the field of neuroscience reveals that the part of the human brain that is responsible for processing visual signals can learn to process other sorts of information like sound, touch etc. Taking this motivation into account, having pre-trained models that fixate on a fusion strategy of the multimodal signals and consist of pre-training objectives across single and mixed modalities, it is possible to generate latent representations that are generalizable across various tasks [7]. Being trained on a huge corpus of data, spanning across a *general domain* many real-world problems can be solved using these pre-trained models. By studying models that have been *pre-trained* on multiple input modalities and pre-training tasks, we hypothesize the possibility of integrating such models in an end-to-end ML pipeline, in order to perform downstream tasks on multimodal data. The question of *how to* optimally integrate these pre-trained models within a machine

learning pipeline framework and generate multi-modal universal embeddings using these models remains an open question in the field of AutoML. We aim to investigate this question through our exploratory research. Our research mainly focuses on tackling table + text, text-only, image-only, table + text + image, table + image and image + text modalities.

1.2 Problem Formulation and Objectives

As stated before, the main goal of this research is to automatically predict *well-performing and generalisable* (over various tasks) Machine Learning Pipelines for **multimodal** data which produce *universal* embeddings or *contextual representations* using multimodal pre-trained architectures. We tackle this by breaking it into 2 sub-problems. For the Research Question 1, we take inspiration from Thorton et al. [44].

1. **RQ1.** How can we incorporate **pre-trained** (Transformer) models in AutoML systems to enable AutoML over the provided unimodal as well as multimodal data?

2. **RQ2.** How can we ***warm-start*** the search of a search algorithm that automatically and simultaneously selects and optimizes the hyperparameters of different components of a ML pipeline?

RQ1: Before we formalise our problem, we will first formally introduce a general algorithm selection problem followed by a general hyperparameter selection problem to better understand our formulation. Having formulated these two problems, we then formulate our problem as a *combined algorithm selection and hyperparameter optimisation* problem.

Let $\mathcal{A}$ be the given set of learning algorithms along with limited training data as $D = \{(x_1, y_1)....(x_n, y_n)\}$. 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 and λ is the corresponding hyperparameters, 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 $\mathcal{L}(.,.)$ is *some* loss. Thus, the general model selection problem becomes [44, 49] :

$$A^* \in \underset{A \in A}{\operatorname{argmin}} \frac{1}{k} \sum_{i=i}^{k} \mathcal{L}(A, D^{(i)}_{train}, D^{(i)}_{valid}) \tag{1.1}$$

where k represents k-fold cross validation that splits D into k equal-sized partitions [1].

Now, we formulate a general hyperparameter optimization problem. This problem becomes similar to the model selection 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 $\lambda_1, \lambda_2 ... \lambda_n$ with domains $\Lambda_1, \Lambda_2 ... \Lambda_n$, the hyperparameter space Λ is a strict subset of the cross product of these domains: $\Lambda \subset \Lambda_1 \times \Lambda_2 \times \Lambda_n$ [44]. 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:

$$\lambda^* \in \underset{\lambda \in \Lambda}{\operatorname{argmin}} \frac{1}{k} \sum_{i=1}^{k} \mathcal{L}(A_\lambda, D_{train}^{(i)}, D_{valid}^{(i)}) \tag{1.2}$$

Having formulated the general algorithm selection problem and the hyperparameter optimisation problem, we now formally define our pipeline search problem as a *combined algorithm selection and hyperparameter* (CASH) optimization problem for an entire pipeline. Let $\mathcal{A}$ be a set of learning algorithms. The set $\mathcal{A}$ consists of pre-trained deep neural models as well as classical ML models. Let the set of algorithms be $\mathcal{A} = \{A_{ptm}^{(1)}, A_{ptm}^{(2)} ... A_{ptm}^{(m)}, A_m^{(1)}, ... A_m^{(n)}\}$ where $\{A_{ptm}^{(1)}, A_{ptm}^{(2)} ... A_{ptm}^{(m)}\}$ are m selected pre-trained models and $\{A_m^{(1)}, A_m^{(2)} ... A_m^{(n)}\}$ are n classical ML models. Let the associated hyperparameter spaces be a strict subset of the cross-product of the domains of the hyperparameters associated with the pre-trained as well as classical models such that $\Lambda \subset \Lambda_{ptm}^{(1)} \times \Lambda_{ptm}^{(2)} .. \times \Lambda_{ptm}^{(m)} \times \Lambda_m^{(1)} .. \Lambda_m^{(n)}$ and a let G be a set of all valid pipeline structures. Let e be the embedding generated by some pre-trained algorithm $A_{ptm}^{(k)} \ni k < m \in \mathcal{A}$ in a higher-dimensional space $\mathbb{R}^E$, such that $e \in \mathbb{R}^E$. It is to be noted that we do not intend to directly control $\hat{e}$, it is generated by the pre-trained model selected in the pipeline configuration. We narrow the scope of our CASH problem by investigating valid pipeline structures $g' \in G$ such that the pipeline consists of at most 1 pre-trained model and at most 1 classical ML model, constructed in a sequential manner (pre-trained model before ML model). This is depicted in fig 1.1.

The tuple $(g', (A_{ptm}, \lambda_i^*), (A_m, \lambda_j^*))$ such that $i \neq j$, represents a valid pipeline structure consisting of a pre-trained model A_{ptm}, its corresponding hyperparameters (more

[1] $D_{train}^{(i)} = D - D_{valid}^{(i)}$

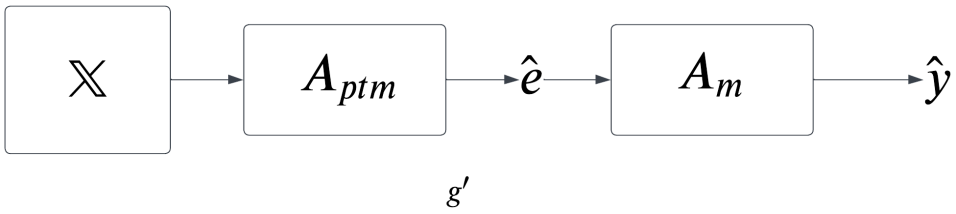


Figure 1.1: Overview of the pipeline structure g' whose components need to be generated and hyperparameters need to be optimised in a combined fashion. A_{ptm} is the selected pre-trained model which generates the embedding $\hat{e}$, which is fed to a classical ML model A_m for mapping $\hat{e}$ to the target domain $\mathbb{Y}$.

than one) λ_i^* followed by a classical ML model A_m as a downstream model along with its hyperparameters (more than one) λ_j^*. Since, $\mathcal{A}$ represents the set of all learning algorithms (pre-trained + downstream) and Λ is the hyperparameter space of the hyperparameters corresponding to the learning algorithms, for shorthand purposes we shall denote a valid pipeline configuration as (g', A^*, λ^*), where A^* denotes multiple learning algorithms, particularly 2 i.e a pre-trained model followed by a classical ML model. Similarly λ^* denotes more than one hyperparameter corresponding to the pre-trained as well as downstream models. Let $\mathbb{P}(X, Y)$ be a joint probability distribution of the feature space $\mathbb{X}$ and target space $\mathbb{Y}$. Let pipeline $\mathcal{P}$ consisting of a pre-trained and a classical ML model be $\mathcal{P}_{g',\hat{A^*},\hat{\lambda^*}}$. Thus the true performance of such a pipeline can be given as:

$$\hat{R}(\mathcal{P}_{g',\hat{A^*},\hat{\lambda^*},P}, P) = \mathbb{E}(\mathcal{L}(h(\mathbb{X}), \mathbb{Y}) = \int \mathcal{L}(h(\mathbb{X}), \mathbb{Y}) dP(\mathbb{X}, \mathbb{Y}) \tag{1.3}$$

with $h(\mathbb{X})$ being the predicted output of $\mathcal{P}_{g',\hat{A^*},\hat{\lambda^*},P}$. As $\mathbb{P}(X, Y)$ is intractable, we adapt equation 1.3 as:

$$\hat{R}(\mathcal{P}_{g',\hat{A^*},\hat{\lambda^*},D}, D) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(h(x_i), y_i) \tag{1.4}$$

Hence, the final objective becomes as [49, 44, 16]:

$$(g', \hat{A^*}, \hat{\lambda^*})^* \in \underset{\hat{A^*} \in \mathcal{A}|g'|, g' \in G, \hat{\lambda^*} \in \Lambda}{\operatorname{argmin}} \frac{1}{k} \sum_{i=1}^{k} \hat{\mathcal{R}}(\mathcal{P}_{g',\hat{A^*},\hat{\lambda^*},D_{train}^{(i)}}, D_{valid}^{(i)}) \tag{1.5}$$

Drawing inspiration from Thornton et al. [44], this problem can be formulated as a single combined hierarchical hyperparameter optimization problem with the hyperparameter space $\Lambda = \Lambda \subset \Lambda_{ptm}^{(1)} \times \Lambda_{ptm}^{(2)} .. \times \Lambda_{ptm}^{(m)} \times \Lambda_m^{(1)} .. \times \Lambda_m^{(n)} \bigcup \{\lambda_r\}$, where λ_r is the root level hyperparameter

that selects the algorithms $\{A^{(1)}_{ptm}, A^{(2)}_{ptm}...A^{(m)}_{ptm}, A^{(1)}_{m}, A^{(2)}_{m}..., A^{(n)}_{m}\}$ and the structure $g' \in G$.

RQ2: In the context of the Combined Algorithm Selection and Hyperparameter Optimisation (CASH) problem, *warm-starting* refers to initializing the optimization process with an initial configuration that is not random but is well motivated by a certain architectural design and some scalar evaluation score. The goal is to accelerate the search process by leveraging well-performing configurations as initial configurations instead of *cold-starting* the search process with any random configuration. Taking inspiration from [15, 49], we formulate the warm-starting problem as follows:

Let T be the set of all known tasks such that $t_j \in T$. For every task t_j, let $\{\lambda_{j,1}....\lambda_{j,k}\}$ represent the evaluated pipeline component as well as the hyperparameter configurations such that $\lambda \in \Lambda$, where Λ is a valid Configuration space. Let P be a set of all prior scalar evaluations for a task t_j on a set of pipeline and hyperparameter configurations over some empirical metric. If t_j represents a task then the i^{th} evaluation on that task be $P_{j,i}$ evaluated for configurations $\{\lambda_{(j,i),1}....\lambda_{(j,i),k}\}$. Each task t_j and its i^{th} evaluation can be described as a vector $m(j,i) = \{\lambda_{(j,i),1}....\lambda_{(j,i),k}, P_{j,i}\}$ [2] to construct a set of prior evaluations or *meta-dataset* M, where $m(j,i) \in M$, see fig 1.2. We define a meta-learner ψ_L, that predicts the mean performance $\mu_{j,i}$ and standard deviation $\sigma_{j,i}$ for some set of meta-features $m(j,i)$ (pipeline components and hyperparameter configurations) corresponding to a task t_j. By conditioning on the joint distribution of the set of meta-feature vectors $m(j,i)$ and Prior Evaluations P, the function mapping for ψ_l can be said as $\psi_L : \Lambda \rightarrow R$. The Meta-Data set can be divided into train M_{train} and evaluation set M_{eval} using a k fold cross-validation technique. Let $a_{\mathcal{M}l}$ be an acquisition function such that $a_{\mathcal{M}l} : \Lambda \rightarrow R$. By finding the highest value in the acquisition function among the evaluations from the set M_{eval}, where the evaluations $(\mu_{j',i'}, \sigma_{j',i'})$ are estimated by the *meta-learner* ψ_l for each vector $m(j',i')$ in M^{eval}, we can get an initial configuration λ_w. This configuration should have a high predicted average performance and a significant associated uncertainty. This is formally represented as:

$$\lambda_w \in \underset{\lambda_w, m(j',i') \in M^{eval}}{\operatorname{argmax}} \sum_{i=1}^{K} \frac{1}{K} a_{\mathcal{M}l}(\psi_{l,M^{train}}(m(j',i'))) \tag{1.6}$$

The final objective of the warm-start problem can be stated as, having started the search of a Bayesian Optimisation (BO) process using an initial configuration λ_w, finding an optimal pipeline $\mathcal{P}_{g',\lambda^*}$ consisting of a set of learning algorithms (pre-trained model and classical models) along with their hyperparameters together represented by λ^* in a valid structure

[2] $P_{j,i} \in P$ is the target variable

g', $g' \in G$.

$$(g', \lambda^*)^* \in \underset{\lambda^* \in \Lambda, g' \in G}{\operatorname{argmin}} \frac{1}{k} \sum_{i=1}^{k} \hat{\mathcal{R}}(\mathcal{P}_{g', \lambda^*, D^{(i)}_{train}}, D^{(i)}_{valid}) \tag{1.7}$$

The following figure depicts how the meta-learner ψ_l predicts the mean predictive performance and standard deviation associated with an input configuration from the evaluation set. The acquisition function is maximised for all values of the predicted mean and standard deviations from the evaluation set. Having maximised these values, λ_w is the pipeline configuration that yields the maximum value for the acquisition function. The pipeline search begins by evaluating λ_w.

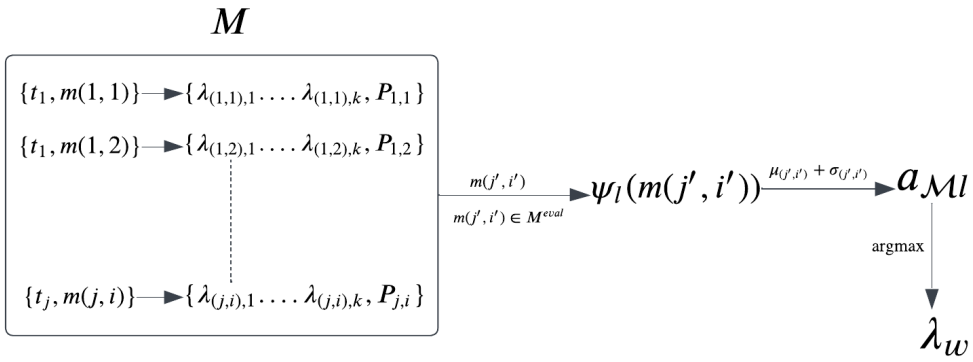


Figure 1.2: Overview of the process for selecting a well-motivated configuration λ_w for starting the search of a Bayesian Optimisation Process.

1.3 Knowledge Utilisation and Impact

Our research is aimed at a future integration into GAMA, allowing the synthesis of versatile ML pipelines for multimodal inputs. This empowers novice users to create universal architectures without extensive expertise, while experts can efficiently search for architectures by using the universal latent representation. Through the incorporation of state-of-the-art pre-trained vision-language models in Θ, we bridge AutoML and vision-language processing. This results in universal ML pipelines with transparently motivated skeletal structures.

Chapter 2

Background & Related Work

In this chapter, we provide an in-depth literature survey and background regarding the concepts and ideas relevant to this research. We research the latest pre-trained benchmark models that deal with mixing vision-language signals for automatically creating machine learning workflows. Additionally, we also investigate the techniques these models use to make sure these pipelines come together well and produce the best possible results. We further dive into the details of these pre-training architectures and their supposed argument towards achieving *generalizability*. This is done to investigate the inclusion of these models in the configuration space Θ that shall be constructed while designing our experiments. Furthermore, thorough research regarding the nature of the components in Θ as well as the structure of Θ itself is provided later in the section. Now, having discussed the existing techniques for creating a structured configuration space, we move on to investigate model-based algorithms that can select algorithms and perform HPO in a combined fashion. In the end, after having investigated the possible pre-trained multimodal models, and algorithms for the CASH problem, we investigate techniques for warm-starting the search of the sequential model-based bayesian optimisation (SMBO) process.

2.1 Pre-Trained Models

In Section 1.1 we define two kinds of AutoML systems and further explain why NAS methods tend to be expensive for the synthesis of ML pipelines. So, it becomes important to research the inclusion of pre-trained architectures (in AutoML systems) that are trained on different types of signals (multiple modalities). This helps in creating ML pipelines that can automatically and effectively generate a common understanding of data from different

sources. AutoML for tabular inputs has been studied widely, and many state-of-the-art pipeline synthesis tools exist [44, 9, 13]. Multimodal signals on the other hand often contain rich correlations between different modalities. Thus, we need architectures that distil this information in the most efficient manner. By having *universal* pre-training objectives (objectives that span across modalities), a more universal representation of the input signals can be found. Transformer models have been found to effectively tend to long input sequences due to the attention mechanism [46]. Thus, due to this reason and also due to their robustness in generalizing over downstream tasks, Transformer models have become ubiquitous in the field of Natural Language Processing (NLP) [5, 18, 28]. Dosovitskiy et al. [6] showcase the same robustness and computational efficiency of Transformer models, particularly the Vision Transformers (ViT), for processing the vision modality. Following these developments, pre-trained vision-language transformer models have been found to efficiently extrapolate the inter-dependency between vision and language modalities [7].

One of the main goals of this research is to investigate the inclusion of these pre-trained vision-language models in AutoML systems for performing AutoML over a set of tasks. Our intention is to investigate the *generalization* performance by incorporating knowledge of well-performing multimodal transformer-based model architectures as well as their pre-trained weights. In the last couple of years, extensive research has been conducted in the field of vision-language systems and large transformer-based model architectures have been proposed to achieve state-of-the-art performance on several vision-language tasks [7]. Incorporating this *prior* knowledge of well-performing, multimodal *pre-trained* models in the pipeline search spaces will fill a gap in the field of AutoML and will help it to catch up with the progress in the field of vision-language systems. Also, it will result in pipeline architectures that are more adaptable to domain shifts, as pre-training models show great robustness in terms of domain adaptation [33]. We study why these architectures result in *universal* contextual cross-modal latent representations. The *universal* nature of these latent representations stems from the nature of the pre-training objectives these models undergo. More universal pre-training objectives across modalities lead towards more universal representations [2]. We will further specifically study FLAVA [40], ALBEF [20] and Data2Vec [2] to understand their ability to better represent multimodal inputs in a unified latent space.

Our hypothesis is that the generalizable performance of these models against CLIP [38], (a dual encoder architecture based vison-language model), implemented by Autogluon (a state of the art (SOTA) AutoML system) for processing vision-language modality should

be better. CLIP is *pre-trained* only to predict a batch of N similar image-text pairs that appear in its training data. To achieve this objective, CLIP learns an embedding space by jointly training the image and text encoders and maximizes the cosine similarity of the image and text embeddings (of the N real pairs in its batch) while minimizing the cosine similarity of the $N^2 - N$ mismatched pairings [38]. As a result of this contrastive-learning *pre-training* objective CLIP fails to generalize over text-only/image-only and multimodal image-text modalities. Autogluon furthermore tackles multimodal embeddings

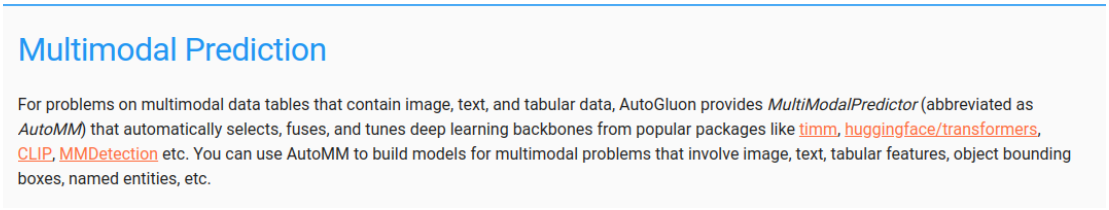


Figure 2.1: An overview of how AutoGluon processes multimodal inputs.

by processing respective modalities using unimodal models (sampled from its model zoo) and then combining the information (embeddings) from these unimodal models in a late-fuse fashion [9]. It further builds a Multilayer Perceptron (MLP) or a set of stacked ensemble models using NAS methods to map the late-fused embeddings concatenated from the respective unimodal models to the target domain. Several works as [6, 7] have shown that, transformer-based vision-language models perform better than CNN-based architectures as well as unimodal (vision or language) Transformer models for processing multimodal vision-language modalities. Some of the *pre-trained* vision-language models such as ALBEF [20] implement information fusion like the aligning respective unimodal embeddings before performing multimodal learning. We feel that it is important to extrapolate these design choices to generate more generalizable end-to-end pipelines rather than doing a *naive cold-search* across architectures. The prior knowledge about the architecture of these pre-trained vision-language models has been vastly supported by their authors by empirically evaluating them against several CNN-based and unimodal transformer-based approaches [7]. Hence, it becomes important to study vision-language models of varying architectures and design choices for obtaining *universal representations*.

In the last few years, several vision-language transformer models have been proposed. These vision-language models showcase different methods of data fusion such as early fuse, and late fuse and can be architecturally categorized into single-stream and dual-stream models [7, 17]. The flexibility of these vision-language transformer models is symptomatic of the pre-training objectives these models undergo [17, 7, 2, 40].

Single-stream architectures assume a simple correlation between the two modalities and argue that they can be handled using a single transformer. This argument of a unified framework is supported by the unordered representation nature of the attention mechanism. OSCAR, VisBERT and VLBERT [24, 21, 42] are a few examples of the single stream vison-language transformer models. However, the limitation of these single-stream architectures is that they apply self-attention directly on the two modalities (vision and language) and do fail to account for intra-modal interactions [17, 7]. See fig 2.2.

Dual stream architectures apply cross-modal attention on multimodal inputs where the key vectors are from one modality (language or text) and the query vector is from another modality (language or text), fig 2.2. LXMERT [43] and ALBEF [20] are examples of dual-stream architectures. ALBEF applies cross-modal attention to unimodal vision and language embeddings before multimodal fusion using another Transformer model. It is possible to process unimodal inputs via ALBEF. Dual encoders consist of unimodal transformer models for generating embeddings for respective modalities which are then fused using another transformer model or late-fusion techniques [17]. Storing pre-computed vision and language embeddings is more effective for retrieval tasks as compared to fusion encoders.

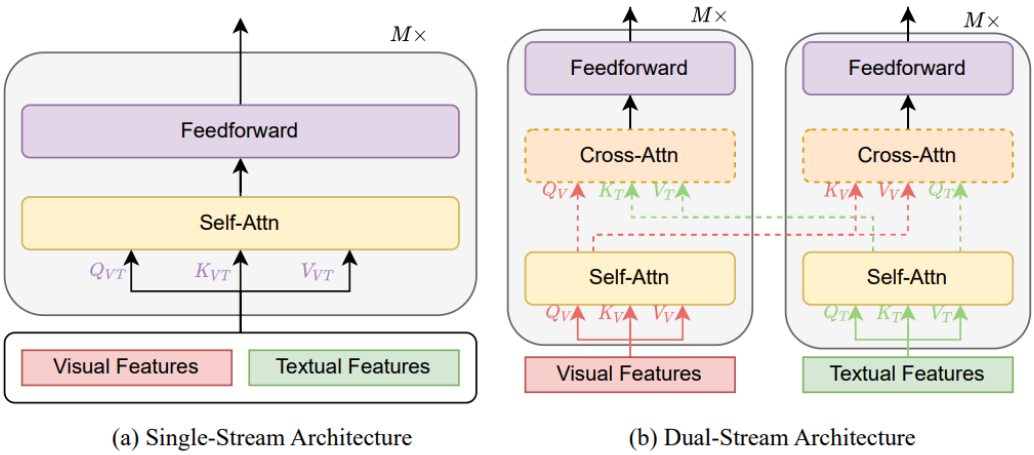


Figure 2.2: Illustration of the single stream and dual stream architectures [7].

We shall now study some of these models and their pre-training objectives broadly and some in detail. UNIMO [22] aims to learn representations in a latent vision-language (high dimensional) space using a contrastive loss. Along with contrastive learning, UNIMO also possesses modality-specific objectives that enable it to process unimodal and multimodal

signals. To process both unimodal and multimodal modalities, Bao et al. propose VLMo [4]. They introduce *Mixture-of-Modality-Experts* (MOME), where each block consists of a modality-specific expert and share a common attention layer. MOME enables VLMo to generate more universal representations by *fusing* embeddings that have been generated using unimodal and multimodal pre-training objectives like masked language modelling (MLM), image-text-contrastive learning (ITC), image-text matching (ITM) respectively. It is natural to understand that the more a pre-trained model learns across modality-specific cross-modal and multimodal objectives, the more universal the generated representations tend to be. Furthermore, Li et al. [23] propose UNIMO-2 as an end-to-end pre-training framework for both aligned and unaligned multimodal as well as unimodal inputs. UNIMO-2 aims to jointly learn from both aligned and unaligned image and text corpora in an end-to-end fashion, effectively alleviating the limitations of the corpus, and learning more generalized visual and textual representations on large volumes of different types of multimodal/unimodal corpora [23].

For studying the effect of different model architectures, and pre-training objectives and to induce architectural flexibility in an AutoML system (by avoiding complete reliance on NAS methods), we select to study the following models in detail:

1. FLAVA [40], a dual-stream architecture yielding cross-modal as well as multimodal embeddings.
2. ALBEF [20], a dual-stream architecture that aligns modalities stemming from respective encoders before feeding them to a multimodal Transformer.
3. Data2Vec [2] a completely modality agnostic pre-trained model.

2.1.1 FLAVA: A Foundational Language and Vision Alignment Model

FLAVA [40] is an example of a fusion + dual encoder architecture that can process both unimodal and multimodal inputs. The authors of FLAVA propose *rewriting-task*, a technique for re-writing image captions on different granular levels like words, sub-words etc.. and creating positive and negative pairs from a single image-text pair for a contrastive learning objective. FLAVA first creates a modality-specific embedding using a modality-specific transformer and then learns a combined representation with the help of a fusion encoder (Transformer model) using contrastive learning. FLAVA [40] architecture consists of a Vision Transformer (ViT) [6] as the Image Encoder, for text encoding, they also use a

ViT but with different parameters, more specifically both encoders are a ViT-B/16. For cross-modal interactions, FLAVA further has a multimodal transformer based on the same ViT architecture initialised with different parameters. The image encoder generates an encoding $h_{CLS,I}$, text encoder generates an encoding $h_{CLS,T}$ and finally the multimodal Transformer encoder generates an encoding $h_{CLS,M}$. To realize *universal* embeddings, FLAVA implements two types of *pre-training* objectives viz. unimodal pre-training and multimodal pre-training objectives. The multimodal objectives are as follows:

1. Global Contrastive Loss: By projecting $h_{CLS,I}$ and $h_{CLS,T}$ into an embedding space, followed by L2-normalization, dot product and a softmax loss, they maximize the cosine similarity between the matched image-text pairs.

2. Masked Multimodal Modelling (MMM): MMM L_{MMM} masks both image patches and text tokens and jointly learns on both modalities by predicting the masked tokens and patches.

3. Image Text Matching (ITM): They further model an ITM loss as done in previous literature [6].

Unimodal *pre-training* objectives are as follows:

1. Masked Image Modeling (MIM): On unimodal image data, they mask the extracted image patched using BEiT [3] and reconstruct them from other image patches. $L_V = \mathbb{E}_{V \epsilon D} log f_\theta(v_m | v_{mask})$

2. Masked Language Modelling (MLM): A fraction (15%) of the text tokens are masked in the input and are reconstructed using a classifier over h_T.
$L_{bidirectional} = -\mathbb{E}_{W \epsilon D} log P_\theta(w_m | w_{mask})$

For unimodal pre-training, they first pre-train the text encoder on the MLM objective on unimodal text data. For pre-training the image encoder they experiment by first training it on the MIM objective before joint training on both unimodal and multimodal datasets. Once, the unimodal pre-training has been realized, FLAVA is trained with round-robin sampling on three types of datasets. Depending on each of the datasets they apply MIM, MLM, MMM and ITM concerning the data from each modality. A comprehensive analysis of the performance of FLAVA against other vision-language Transformer models is provided in the Appendix.

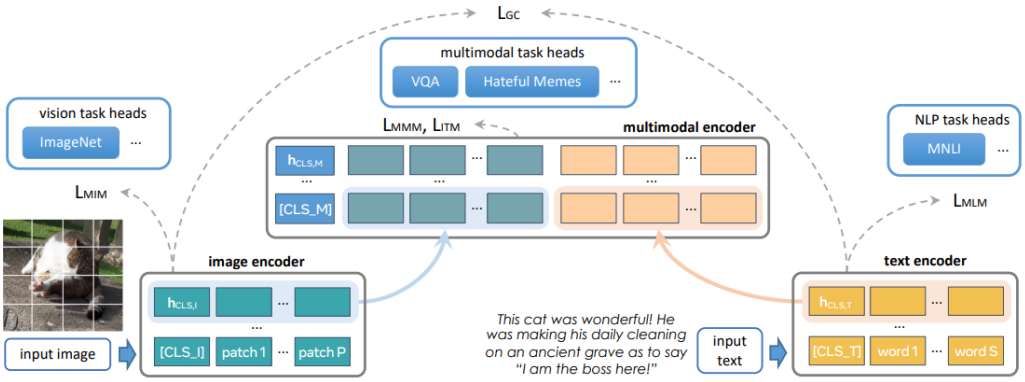


Figure 2.3: An overview of the FLAVA model, with an image encoder transformer to capture unimodal representations, a text encoder to process unimodal text information, and a multimodal encoder transformer that takes as input encoded unimodal image and text and integrates their representations for multimodal reasoning. During pretraining, masked image modelling (MIM) and mask language modelling (MLM) losses are applied to the image and text encoders over a single image or text piece, respectively, while contrastive, masked multimodal modelling (MMM) and image-text-matching (ITM) loss are used over paired image-text data [40].

2.1.2 ALBEF: Align Before Fuse

ALBEF contains an image encoder, a text encoder, and a multimodal encoder. It implements a 12-layer visual transformer ViT-B/16 [38] as the image encoder, initialized with weights pre-trained on ImageNet-1k from [31]. ALBEF implements a 12-layer visual transformer ViT-B/16 [3, 6] as the image encoder. The weights are initialized with the weights pre-trained on the ImageNet-1k data. An input image I is encoded into a sequence of embeddings: $v_{cls}, v_1, ..., v_N$, where v_{cls} is the embedding of the [CLS] token. ALBEF uses a 6-layer transformer model for both the text encoder and the multimodal encoder. The text encoder is initialized using the first 6 layers of the BERTbase [5] model and the multimodal encoder is initialized using the last 6 layers of the BERTbase. The text encoder transforms an input text T into a sequence of embeddings $w_{cls}, w_1, ..., w_N$, which is fed to the multimodal encoder. ALBEF then fuses the image features with the text features through cross-attention at each layer of the multimodal encoder.

ALBEF uses Image-Text Contrastive Learning (ITC), Masked language modelling (MLM), Image Text Matching (ITM) objectives for pre-training the model. The objectives are as follows:

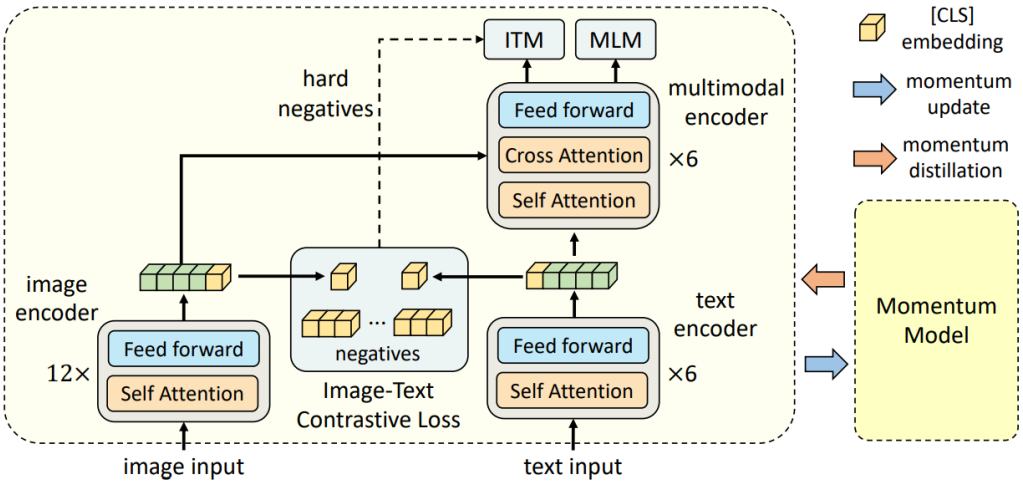


Figure 2.4: ALBEF consists of an image encoder, a text encoder, and a multimodal encoder. ALBEF uses an image-text contrastive loss to align the unimodal representations of an image-text pair **before fusion**. An image-text matching loss (using in-batch hard negatives mined through contrastive similarity) and a masked-language-modeling loss are applied to learn multimodal interactions between image and text. To improve learning with noisy data, ALBEF generates pseudo-targets using the momentum model (a moving-average version of the base model) as additional supervision during training [20].

1. **Image Text Contrastive Learning (ITC)**: The purpose of this objective is to learn better representations before fusion. It learns a similarity function $s = g_v(v_{cls}^T)g_w(w_{cls})$ so that the image text pairs that are in parallel have a higher similarity score. g_v, g_w are linear transformations that map the embeddings in a 256−d space. The normalised features from the momentum encoder are represented as $g'_v(v'_{cls}), g'_w(w'_{cls})$. Then, $s(I,T) = g_v(v_{cls}^T)g'_w(w'_{cls})$ and $s(T,I) = g_w(w_{cls}^T)g'_v(v'_{cls})$. For each image text pair, a softmax image text and text image similarity are computed as follows:

$$p_m^{i2t}(I) = \frac{exp(s(I,T_m)/\tau)}{\sum_m^M exp(s(I,T_m)/\tau)} \tag{2.1}$$

$$p_m^{t2i}(T) = \frac{exp(s(T,I_m)/\tau)}{\sum_m^M exp(s(T,I_m)/\tau)} \tag{2.2}$$

where τ is a learnable parameter. Let $y^{i2t}(I), y^{t2i}(T)$ denote the ground truth labels.

The image-text contrastive loss is defined as the cross entropy H between p and y:

$$\mathcal{L}_{itc} = \frac{1}{2}\mathbb{E}_{(I,T)\tilde{}D}[H(y^{i2t}(I), p^{i2t}(I)) + H(y^{t2i}(T), p^{t2i}(T))] \tag{2.3}$$

2. The **MLM** and the **ITM** tasks are the same as that of FLAVA and can be described as:

$$\mathcal{L}_{mlm} = \mathbb{E}_{(I,\tilde{T})}[H(y^{msk}, p^{msk}(I, \tilde{T})] \tag{2.4}$$

where y^{msk} and $p^{msk}(I, \tilde{T})$ are the 15% masked input tokens.

$$\mathcal{L}_{itm} = \mathbb{E}_{(I,T)\tilde{}D}[H(y^{itm}, p^{itm}(I, T)] \tag{2.5}$$

where y^{itm} is a 2-dimensional one-hot vector representing the ground-truth label.

2.1.3 Data2Vec: A General Framework for Self-supervised Learning in Speech, Vision and Language

Baevski et al. propose a modalitity agnostic pre-training architecture called Data2Vec [2]. Their core idea is to predict latent representations with the help of a masked view of the input using a self-distillation setup based on a Transformer model architecture. Data2Vec [2] proposes a completely model agnostic *pre-training* architecture. They do not implement multimodal pre-taining but the idea is to adapt this entire architecture across several modalities. We can extrapolate this flexible nature of their *pre-training* objective to generate modality-specific embeddings and then follow a *Late-Fusion* (LF) strategy to fuse these embeddings and further use a stack ensemble method to generate cross-modal representations as proposed by Shi et al. [10]. Further explanation on *why* LF is provided in Section 4.1.3. Data2Vec uses a standard transformer architecture to process the embeddings, where text inputs are processed using a Byte-Pair-Encoding strategy and image inputs are encoded by extrapolating 16×16 image features using ViT. The idea is to mask a fraction (15%) of the input vector and adopt an objective to predict these masked tokens. The key idea is that the encoding of the unmasked input is parameterized by a teacher model via the exponential moving average (MVA) of the model parameters $\theta\ \Delta \leftarrow \tau\Delta + (1-\tau)\theta$ where τ is a learnable temperature parameter. For generating contextualized targets, they normalize the outputs of the top K blocks of an L layered Transformer $y_t = \frac{1}{K}\sum_{l=L-K+1}^{L} \hat{a}_t^l$ where $\hat{a}_t^l$ is the output of each of the top K attention layers. Thus, the *pre-training* objective is defined

as:

$$\mathcal{L}(y_t, f_t(x)) = \begin{cases} \frac{1}{2}(y_t - f_t(x))^2/\beta \; |y_t - f_t(x)| \leq \beta \\ (|y_t - f_t| - \frac{1}{2}\beta) \text{ otherwise} \end{cases} \tag{2.6}$$

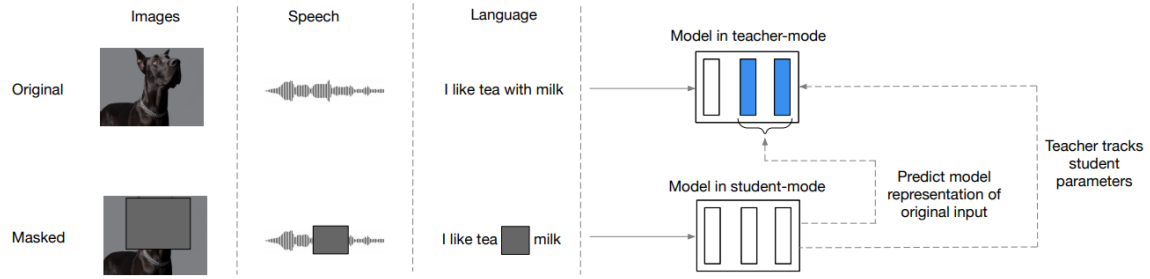


Figure 2.5: Illustration of how Data2Vec follows the same learning process for different modalities. The model first produces representations of the original input example (teacher mode) which are then regressed by the same model based on a masked version of the input. The teacher parameters are an exponentially moving average of the student weights. The student predicts the average of K network layers of the teacher (shaded in blue) [2].

2.2 Configuration Space

After studying several pre-trained vision-language models, we move on to discuss the *configuration space* and its associated relevant literature. 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 [16]. This space is structured and parameterized to confine the search of a *search* algorithm [33, 16]. Current AutoML systems like AutoWEKA[44], AutoGluon[9] and AutoSklearn [13] construct these spaces as a hierarchical space to enable a guided search strategy. Fig 2.6 illustrates an abstract, high-level view of selecting an algorithm A from a configuration space or search space $\mathcal{A}$ based on some performance estimation criterion. Any selected search strategy to explore the 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 [8]. According to Vanschoren [15], 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.

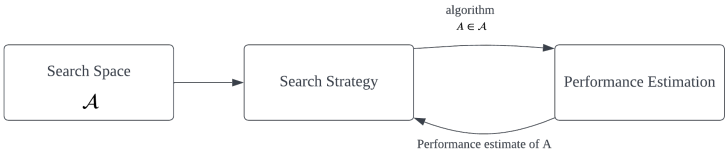


Figure 2.6: Illustration of selecting an algorithm A from the search space $\mathcal{A}$ [8].

2.3 Pipeline Synthesis as a Regression Problem

Having constructed this high-dimensional configuration space, the problem of automatically predicting ML pipelines can be solved as a *combined algorithm selection and hyperparameter optimization* (CASH) problem [44]. Many search strategies, such as grid search, random search, Bayesian Optimization (BO), gradient-based optimization, evolutionary algorithms, Multi-armed Bandit and Sequential Model-Based Optimization (SMBO) learning exist to explore a given configuration space [8, 49]. According to Zoller et al [49] and Feurer et al[14], random search is expensive in terms of computation and grid search does not scale well for large configuration spaces. Additionally, Zoller et al. [49] state that a CASH problem can be treated as a regression problem and can be optimized by SMBO techniques like *sequential model-based algorithm configuration* (SMAC) and Tree-structured Parzen Estimator (TPE). AutoWEKA tackles the CASH problem using both SMAC and TPE [44].

We shall now provide a detailed overview of how the CASH problem can be solved using some model-based algorithms. We take motivation from AutoWEKA [44], a SOTA AutoML tool that tackles the algorithm selection problem using both SMAC and TPE. Equation 1.5 can be tackled using SMBO algorithms like SMAC and TPE which are versatile algorithms that tackle both continuous and categorical hyperparameters and exploit the hierarchical nature of the configuration space Θ. We can proceed to tackle the problem by first constructing a model $\mathcal{M}_{\mathcal{L}}$ and capturing its dependence on some loss function $\mathcal{L}$ on hyperparameter setting λ. The algorithm shall iterate in the following manner: First use $\mathcal{M}_{\mathcal{L}}$ to determine a promising hyperparameter and pipeline configuration λ, evaluate a loss c over the loss function $\mathcal{L}$ and then update the model $\mathcal{M}_{\mathcal{L}}$ with the new obtained data-point (λ, c). To select the next hyperparameter and pipeline configuration λ using the model $\mathcal{M}_{\mathcal{L}}$ an *acquisition function* $a_{\mathcal{M}_{\mathcal{L}}}$, $a_{\mathcal{M}_{\mathcal{L}}} : \Lambda \to \mathbb{R}$ is used. $a_{\mathcal{M}_{\mathcal{L}}}$ uses the predictive distribution of model $\mathcal{M}_{\mathcal{L}}$ at arbitrary hyperparameter and pipeline configurations $\lambda \in \Lambda$ to quantify in closed-form manner, how useful the knowledge about certain λ would be. Then, we can simply maximize this function over Λ to select the most useful configuration λ that

would be evaluated next. Of several acquisition functions, SMAC uses the *positive expected improvement* (EI) function attainable over an existing given error rate c_{min}. Let $c(\lambda)$ be the error of the hyperparameter configuration λ. Then, the positive improvement function over c_{min} is defined as follows:

$$I_{c_{min}}(\lambda) := \max\{c_{min} - c(\lambda), 0\} \tag{2.7}$$

We do not know $c(\lambda)$ but it its expectation with respect to the model $\mathcal{M}_{\mathcal{L}}$ can be computed as:

$$\mathbb{E}_{\mathcal{M}_{\mathcal{L}}}[I_{c_{min}}(\lambda)] = \int_{-\infty}^{c_{min}} \max\{c_{min} - c, 0\} \cdot p_{\mathcal{M}_L}(c|\lambda)dc. \tag{2.8}$$

AutoWEKA [44] uses random forest models as $\mathcal{M}_L$ since they tend to perform well with discrete and high-dimensional data. Although random forest models are not probabilistic models, SMAC thus obtains a predictive mean μ_λ and variance σ_λ^2 of $p(c|\lambda)$ as a frequentist shall estimate over the predictions of its individual trees for λ [44]. It the models $p_{\mathcal{M}_L}$ as a Gaussian $\mathcal{N}(\mu_\lambda, \sigma_\lambda^2)$. Under this defined $p_{\mathcal{M}_L}$ EI in closed form can be realised as follows:

$$\mathbb{E}_{\mathcal{M}_{\mathcal{L}}}[I_{c_{min}}(\lambda)] = \sigma_\lambda \cdot [\mu \cdot \Phi(\mu) + \Psi(\mu)] \tag{2.9}$$

where $\mu = \frac{c_{min} - \mu_\lambda}{\sigma_\lambda}$ and Φ, Ψ denote the probability and cumulative density functions of a standard normal distribution respectively. Authors of AutoWEKA claim that SMAC is designed for robust optimization under noisy function evaluations. Further, it implements a special mechanism to keep track of the best-known configuration and assures high confidence in its estimate of that particular configuration. Due to the nature of SMAC, for a particular configuration λ to outperform its previous competitors it has to remain incumbent in every comparison made [44].

While SMAC models $p(c|\lambda)$ directly, TPE uses separate models for $p(c)$ and $p(\lambda|c)$ [44]. TPE specifically models $p(\lambda|c)$ as one of the two density estimates, conditional on whether c is greater or less than a certain threshold c^*:

$$p(\lambda|c) = \begin{cases} l(\lambda), \ if c < c^* \\ g(\lambda), \ if c \geq c^* \end{cases} \tag{2.10}$$

c^* is chosen as the γ-quantle of the losses TPE observed so far. The default value of $\gamma = 0.15$ [44]. $l(\cdot)$ is a density estimate learned for all the previous hyperparameter configurations λ with a corresponding loss smaller than c^*. Similarly, $g(\cdot)$ is a density estimate learned for

all the previous hyperparameter configurations λ with a corresponding loss greater or equal to c^*. Hence, we have two density estimators, $l(\cdot)$ for configurations that perform well and $g(\cdot)$ for configurations that perform poorly with respect to c^*. EI for TPE is formulated as follows:

$$\mathbb{E}_{\mathcal{M}_\mathcal{L}}[I_{c_{min}}(\lambda)] \propto \left(\gamma + \frac{g(\lambda)}{l(\lambda)} \cdot (1-\gamma)\right) \tag{2.11}$$

By maximizing equation 2.11, TPE generates many candidate hyperparameter configurations λ at random and picks the one with the smallest $\frac{g(\lambda)}{l(\lambda)}$. The density estimators $l(\cdot)$ and $g(\cdot)$ have a hierarchical structure with discrete, continuous and conditional variables reflecting various hyperparameters and their relationships [44].

2.4 Warm Starting

Efficiency becomes an essential factor when designing AutoML systems. It is not only the possibility of generating end-to-end *complex* pipelines that are desired but this should be realised efficiently. One way of making an AutoML system efficient is by *warm-starting* the search for the optimal set of configurations. Humans when learning new skills rarely start completely from scratch. We garner prior experiences and utilise those experiences to perform well on newer unseen tasks. In general we *learn how to learn across tasks* [15]. This in general forms a motivation for *quickly* finding pipeline architectures that perform well on a set of tasks. Significant work has been done towards investigating methods that enable a Bayesian Optimisation method to *warm-start* its search for optimal configurations.

Vanschoren [15] provides an extensive survey into transferring knowledge from prior experiences and methodologies based on incorporating these experiences for making new predictions. Several methods using *surrogate* models based on prior evaluations of pipeline configurations $\theta^* \in \Theta$ have been studied. Zoller et al. and Vanschoren[49, 15] recommend collecting *meta-features* M and evaluations of prior tasks P and learning this joint representation using a *meta-learner* $f_L : M \times \Theta \rightarrow R$ to predict the scalar performance of any configuration θ_i. Model-based optimization approaches like SMBO tend to benefit greatly given an initial set of promising configurations [15]. Ngyuen et al. further [31] investigate constructing new pipelines using a beam search that is focused on components predicted by a meta-learner and is itself trained on the examples of prior successful evaluations. Furthermore, Feurer et al. recommend exploring the configuration space by imposing a fixed structure on the pipeline that is described by the set of its hyperparameters and then finding optimal pipelines using Sequential Model-Based Bayesian Optimization

(SMBO) [12].

From the works of Hutter et al. and Zoller et al., [16, 49], we infer that having a general well-performing baseline for searching the configuration space for a pipeline synthesis problem results in faster convergence enabling an efficient way for generating pipeline configurations. Furthermore, it also makes sense to induce the *prior* knowledge regarding well-performing pipeline evaluations (baselines) as it makes our search strategy *more* informed and guided in nature [30]. *meta-learning* enables us to warm start the synthesis of such pipelines. Apart from this, Hutter et al. [33] and Liu et al. [29] in their research work on transferring the knowledge regarding a model's architecture and pre-trained weights by integrating the pre-trained models in their Configuration Space Θ. Thus, the pipeline synthesis can be warm-started by evaluating pre-trained model architectures (as a pipeline component) that tend to be robust in terms of domain adaptation. Additionally, more knowledge regarding a domain can be transferred by initializing the weights (of the pre-trained model) using the weights optimised during the pre-training of these models. It will be of interest to study the warm start of a Bayesian Optimisation (BO) process (particularly SMBO) for a given meta-dataset (M) of prior evaluations, meta-learner ψ_l and a configuration space consisting of consisting of pre-trained models initialised using their pre-trained weights.

Chapter 3

Methodology

3.1 Overall Methodology and Decomposition

To investigate the problem of integrating pre-trained deep neural models in AutoML Systems to perform AutoML over multimodal data, we first validate the performance of a pipeline and its structure for multimodal datasets by making prior evaluations. During these prior evaluations, we experiment with fusing tabular embeddings generated using a NAS method with multimodal vision-language (also only-vision or only-language) embeddings generated by a pre-trained multimodal transformer model. The goal is to extract a pipeline structure that yields well-performing prior evaluations in order to achieve the final purpose of automatically generating and optimizing components (pre-trained + classical model) in this pipeline structure (g'). The hypothesis is that a well-performing pipeline for multimodal data should have an *optimal* universal latent representation $\hat{e} \in R^{\mathcal{E}}$, that is generated by the pre-trained model given its corresponding hyperparameters. These prior evaluations are then used to construct a meta-dataset M. Having experimented on different pipeline components like various pre-trained models as well as classical ML models, alongside the NAS method (if tabular signals are present), a hierarchical configuration space is constructed. The next task that is performed is to define the objective function of the optimisation process that shall compute the objective value for actually evaluating the sampled configuration. Having done this, a meta-learner ψ_l is constructed along with an acquisition function $a_{\mathcal{M}l}$. By maximizing $a_{\mathcal{M}l}$ over the evaluations from an evaluation meta-dataset M^{eval}, a configuration θ_w with a high predictive mean and uncertainty is found that is used to warm start the search of an SMBO process. The aim is to converge to a set of *near-optimal* configurations θ_k^* *efficiently*. Moreover, to evaluate the Bayesian

Optimization process for generating optimised configurations θ_k^* consisting of pre-trained models, we use an 'any-time' evaluation metric to study optimisation under limited data and a restricted time budget [29]. We provide a breakdown of our methodology to tackle **RQ.1** and **RQ.2** in the following steps:

1. Prior Evaluation and building of Meta-Dataset.
2. Construction of the configuration space Θ.
3. Defining the objective function
4. Setting up and performing SMAC.
5. Defining any time learning metric to evaluate the optimisation process.

The following figure 3.1 indicates the overall view of our methodology. M is the meta-dataset of prior evaluations over multimodal tasks T. These evaluations consist of information regarding the pre-trained and classical ML models constructed in a structure g'. The pipeline may also consist of a NAS-based component for handling Tabular inputs (if present) [1]. The formulated Combined Algorithm Selection and Hyperparameter optimization problem (CASH) is solved using an SMBO process called the Sequential Model-Based Algorithm and Configuration selection procedure (SMAC). The configuration space Θ, meta-learner ψ_l and acquisition function $a_{\mathcal{M}l}$ are all components of SMAC that interact together to produce a warm-starting configuration θ_w and next-configurations (θ_k^*) for the optimisation process.

[1] represented in a dotted box

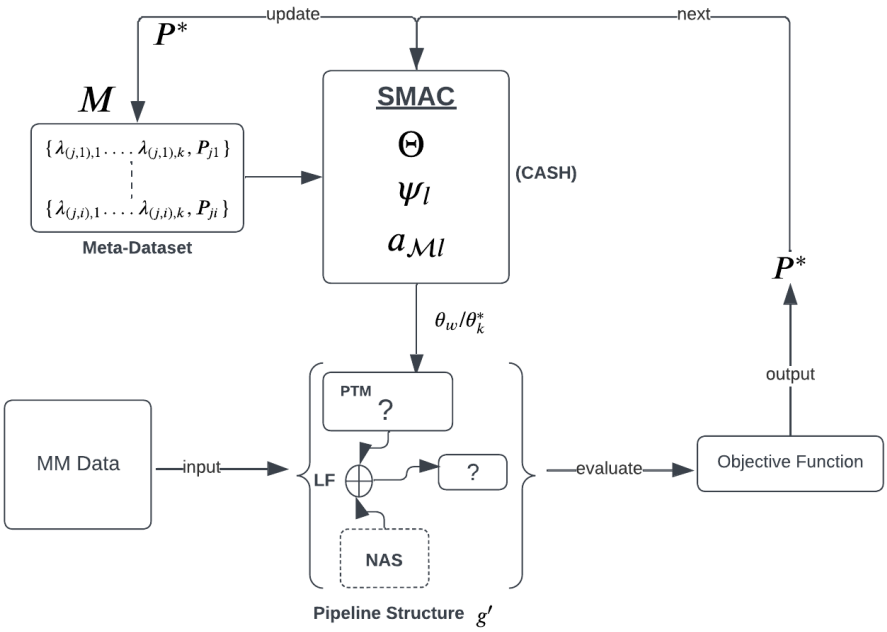


Figure 3.1: The above figure provides an overview of our methodology for integrating pre-trained models in AutoML systems for *warm-starting* AutoML over a given multimodal data. Here, LF indicates *late-fuse*.

3.2 Prior Evaluations and Building Meta-Dataset M

AutoGluon [9] has emerged as a leading solution for multimodal processing, particularly in scenarios that involve combining vision and text, leveraging the capabilities of CLIP [38] (section 2.1). It further employs a dual-stream approach for processing respective modalities followed by a fusion step using MLPs or ensemble models for subsequent mapping through classical machine-learning models. Inspired by the findings of [39], who demonstrated the effectiveness of dual-stream processing and late-fusion, we introduce a novel pipeline structure g' (depicted in fig 3.1). In this pipeline structure, NAS-based methods are exclusively dedicated to processing the tabular data, a strategic choice informed by the observations made by Liang et al. [25] and further substantiated by Shi et al. [10, 9]. Our central hypothesis postulates that the integration of pre-trained models, designed for well-established multimodal benchmarks and characterized by their effective generalization, can significantly expedite the convergence of Sequential Model-Based Optimization (SMBO). To substantiate

this, our experimental scope encompasses three distinct multimodal configurations: tabular + text, text + vision, and tabular + text + vision.

Current literature, however, exhibits a noticeable dearth in terms of comprehensive benchmark surveys addressing end-to-end pipeline architectures featuring pre-trained models for multimodal data processing. Although VALSE [34] contributes to the understanding of vision-linguistic grounding capabilities, it falls short of constituting a thorough exploration of complete multimodal pipelines. Our work endeavours to bridge this gap by evaluating various architecture paradigms transferred through diverse pre-trained models in the context of downstream tasks. This empirical assessment is conducted within specified time constraints, yielding the requisite performance metrics and contributing to the construction of the essential meta-data M.

To integrate pre-trained multimodal transformer-based models in AutoML systems to perform AutoML over multimodal data, we select the following tasks: classification, regression, image text matching (ITM) and visual question answering (VQA). As mentioned previously, we study 3 modalities and for processing the Tabular modality, we set up a NAS-based method. For processing the other two modalities (vision and language), we make use of pre-trained multimodal transformer models for obtaining *universal* latent representations of the multimodal input data. Given the 4 tasks and 3 modalities, we set up 3 variants of our pipeline structure g' for 1. Conducting prior evaluations. 2. To automatically select and optimise components of the selected pipeline structure and the respective variant. The pre-trained multimodal models form the core aspect of this pipeline structure, hence we shall discuss their setup and workflow separately in the Core Component subsection. Once we have discussed our core component, we discuss how this core component is set up in 3 different variants for the 4 selected tasks across 3 modalities for performing prior evaluations and SMAC experiments.

In the context of the **tabular + text** modality, our approach is inspired by Shi et al.'s work [10] that proposes a *late-fusion* strategy called *multimodal net* for processing text and tabular inputs. We adopt a similar architecture to validate the processing of the three mentioned modalities: tabular + text, text + vision, and text + vision + tabular. Shi et al.'s approach has shown promising predictive performance, making it a suitable foundation. We propose a pipeline architecture (g') that utilizes pre-trained multimodal models, such as FLAVA, ALBEF, and Data2Vec, to extract multi-modal interactions. These models are initialized with their pre-trained weights, facilitating knowledge transfer. NAS methods, efficient for single and tabular modalities, are constrained to process only the tabular

modality. We evaluate these models over various tasks such as classification, visual question answering, image text matching, and regression, to generate a set of prior evaluations referred to as the meta-dataset M. The pipeline (g') abstract structure is depicted in

AutoMM Datasets				
Data ID	#Train	#Test	Task	metric
prod	5,091	1,273	multiclass	accuracy
salary	15,84	3961	multiclass	accuracy
airbnb	18,316	4,579	multiclass	accuracy
channel	20,284	5,071	multiclass	accuracy
wine	84,123	21,031	multiclass	accuracy
imdb	800	200	binary	roc_auc
fake	12,725	3,182	binary	roc_auc
jigsaw	86,052	21,626	binary	roc_auc
qaa	4,863	1,216	regression	r2
qaq	4,863	1,216	regression	r2
book	4,989	1,248	regression	r2
jc	10,860	2,715	regression	r2
cloth	18,788	4,698	regression	r2
ae	22,662	5,666	regression	r2
pop	24,007	6,002	regression	r2
house	24,007	6,002	regression	r2
mercari	100,000	25,000	regression	r2

Table 3.1: List of the 18 AutoMM Benchmark Datasets released by Shi et al. [39], github. We studied these datasets for evaluating multimodal pipeline configurations over the above-listed metrics.

Figure 3.1 and 3.8. For each task, we initialize the relevant processor, encode the input, and generate high-dimensional representations ($\hat{e}$) using pre-trained models like FLAVA or Albef. These models leverage their architecture and pre-trained weights for cross-modal and multimodal interactions. Then, we late-fuse $\hat{e}$ with tabular embeddings, generated using NAS methods. Optimal downstream classical ML models are then determined via NAS, mapping the late-fused representation to the target domain. This ensures well-performing or sub-optimal configurations for the construction of the meta-dataset M. We evaluate the pipeline using the Area Under the Curve (AUC) metric for classification and visual question answering tasks, and R^2 for regression and image text matching tasks.

The prior evaluations and resulting meta-dataset M guide the construction of configuration space Θ for the subsequent Sequential Model-Based Optimization process. The entire procedure is detailed in the subsections below:

3.2.1 Core of the Method

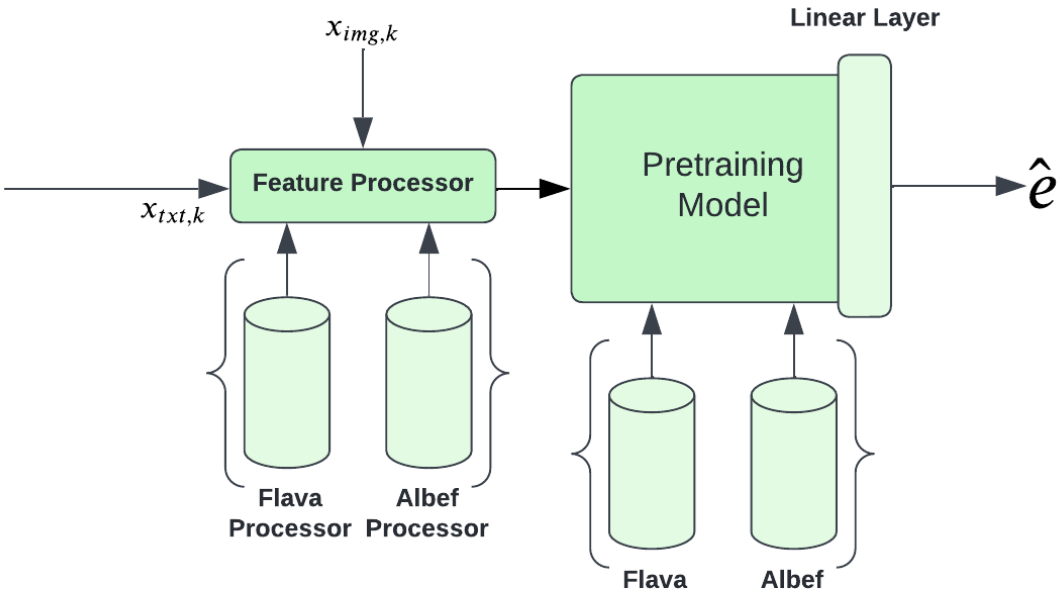


Figure 3.2: The above figure depicts the core component of our pipeline architecture. We demonstrate two of our three selected pre-trained models along with their respective feature processors. The models output a unified latent representation $\hat{e}$ provided vision-language input data.

The core ideology behind the methodology for constructing a pipeline architecture g' is to realise a universal latent representation of the input multimodal data. Additionally, we want to generate these multimodal representations efficiently. We conjecture that processing multiple modalities together and extracting intricate cross-modal interactions in the form of *unified* latent representations using sophisticated architectures shall lead towards a better representation of the input data. Pipeline configurations that possess the capability of generating such *universal* multimodal representations, shall lead towards an optimal performance evaluated over some scalar evaluation metric. Thus, we collect evaluations for pipeline configurations by setting up relevant pre-trained models (alongside NAS for tabular signals if present) as valid components for generating universal latent representations. It is to be noted that these pre-trained models are initialised using their respective pre-trained weights. The performance score obtained over these configuration evaluations acts as a proxy for the predictive signal contained in the unified latent representation $\hat{e}$ generated

by the multimodal pre-trained model. An embedding $\hat{e}$ containing a rich predictive signal should lead to an overall better performance of the pipeline for some task t_j. As mentioned previously, we study 3 pre-trained models such as FLAVA, Data2Vec, and Albef which are completely different from each other in terms of their overall architecture. As pre-trained models have specific pre-processors for processing the raw input data, we do not run an algorithm search over these processors for our pre-trained models. Further details regarding our tokenization strategy are provided in section 4.1. Figure 3.2 represents the core component of this methodology i.e. the pre-trained model preceded by its relevant feature processors and followed by a linear layer on top.

The feature processor of the pre-trained models is fed the raw inputs in batches. Apart from the tabular modality, the input to the pre-trained model could be text, image or both together or $(x_{txt,k}, x_{img,k}), (x_{txt,k},), (, x_{img,k})$. Given various multimodal pre-training objectives during the course of its pre-training, a multimodal pre-trained model learns to generate a representation $\hat{e}$ of the unimodal or multimodal input signal in a *unified* high-dimensional vector space. This ability itself helps the pipeline to generalise over unimodal as well as multimodal tasks, preventing us from building multiple modality-specific encoders in parallel in our final pipeline structure. We realise $\hat{e}$ by initializing the pre-trained models using their pre-trained weights. The argument for this is to enable the transfer of knowledge across domains to build justified baselines, which are to be further evaluated for hyperparameter optimisation.

We shall now discuss the constructed variants of our pipeline structure depending on different modalities and tasks.

3.2.2 Pipeline Variant 1

To make our system efficient, we restrict performing NAS for finding suitable configurations (for a given dataset and task) only for tackling one modality i.e. the tabular modality, in the described multimodal setting. Our argument as stated in the above sub-section is to process vision-language multimodal or unimodal signals in a sophisticated manner i.e. by using pre-trained models for generating well-motivated universal representations of the vision-language multimodal or unimodal signals. Liang et al. and Shi et al. [25, 10] have previously stated the robustness of NAS methods for tackling the tabular modality, which lays the groundwork for our inspiration for constructing these pipeline variants.

Overall, our pipeline framework can be categorised into two broad categories such

as 1. Pipeline architecture consisting of a NAS method for tackling the tabular modality 2. Pipeline architectures not implementing NAS due to the absence of the tabular modality.

Pipeline Variant 1 is designed to tackle multimodal input data which is labelled and comprises the tabular modality. We use this framework to tackle multimodal labelled data about the tabular + text and tabular + text + vision modality.

Tabular + Text Modality

We tackle this modality by performing evaluations of the classification and regression tasks. The classification tasks can be of further two types i.e. Binary and Multi-Class Classification tasks. Figure 3.3 shows the framework of the adopted pipeline structure for tackling the tabular + text modality over classification and regression tasks. For this modality, we evaluate two vision-language pre-trained models along with their pre-trained weights such as FLAVA and Data2Vec along with a Pure NAS method implemented by Autogluon [9]. The argument for evaluating these two models is that FLAVA [40] is a model that belongs to the Fusion + Dual Encoder Architecture class [7]. Data2Vec [2] on the other hand consists of a completely modality-agnostic pre-training architecture, that aims at predicting not only the masked but also the entire input embeddings, making it possible for the model to generalise robustly across various domains of particular modality. It is to be noted that Data2Vec is not a multimodal model, meaning it does not handle multimodal multiple modalities simultaneously as it lacks a multimodal loss head but it can be used for processing multiple modalities (one at a time). We argue that, by experimenting on models with a variable architectural degree, it is possible to induce flexibility in the configuration space, for the sampling of *generalisable* configurations for a given Machine Learning task. We decided to evaluate the models in the pipeline architecture in fig 3.3 through the **zero-shot** evaluation strategy. We aim to explore a robust optimisation strategy that will converge noisy functions under budgeted time and limited data. Thus, opting for a **zero-shot** evaluation strategy seems the most suitable choice as the inferencing can be done fairly quickly without creating unknown bottlenecks in the inferencing procedure [33].

We initialize our AutoMM benchmark [10] by sequentially fetching the 18 datasets curated by Shi et al. For each dataset, we manually identify the text and tabular columns, and then independently process their embeddings. To process tabular data, we utilize AutoGluon's multimodal predictor. Unlike the tabular predictor, which employs tree-based models, the multimodal predictor employs Neural Architecture Search (NAS) to construct a multi-layer perceptron (MLP), referred to as the "multimodal-net". This model, often

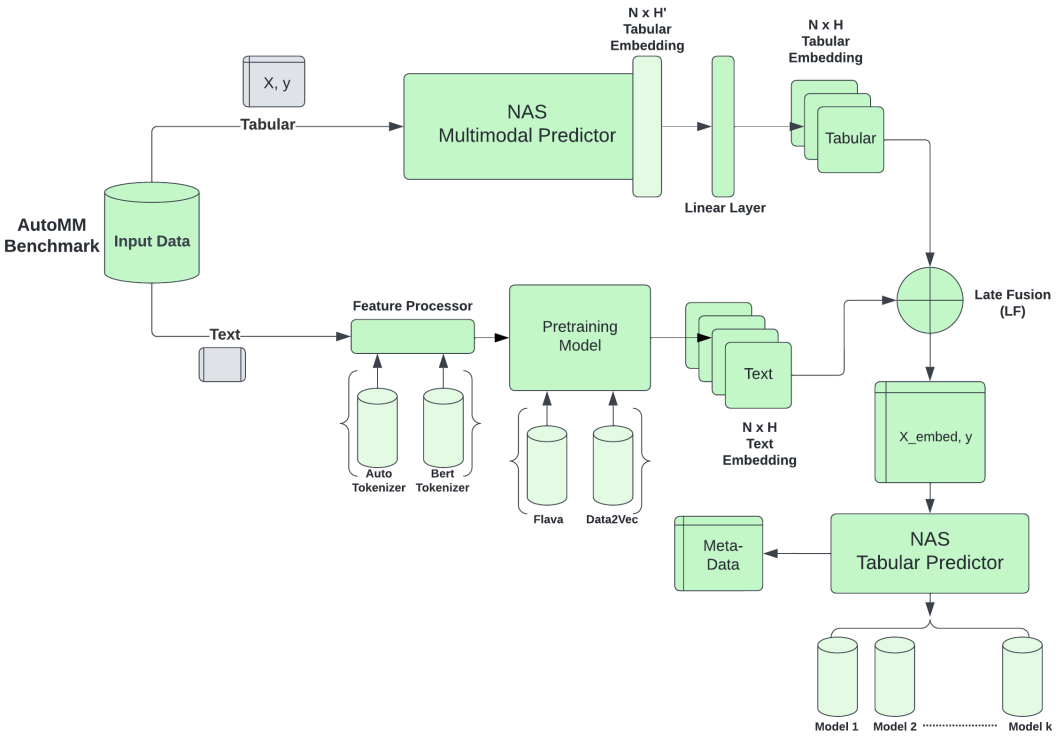


Figure 3.3: Architectural Design of the end-to-end pipeline used to obtain prior evaluations P over the AutoMM Benchmark. Multimodal Predictor in the diagram refers to the Multimodal Predictor class of AutoGluon and similarly, Tabular Predictor refers to the Tabular Predictor class of AutoGluon. N is the number of data points. H' represents an intermediary hidden dimension and H is the final hidden dimension of the multimodal embeddings.

composed of around 109 million parameters, is used due to its capacity to handle complex interactions between modalities. Given the dual-stream approach of multimodal processing, we consider that predictive signals could reside exclusively in text or tabular features, or involve intricate interactions. Inspired by the work of Liang et al. [25] and motivated by Shi et al. [39], we implement a Late-Fusion (LF) strategy to combine these embeddings. Tokenized text is input to a pre-trained model like FLAVA or Data2Vec, generating multimodal embeddings of size $N \times H$ (N being data size, H latent dimension). Similarly,

the multimodal-net processes tabular features to yield tabular embeddings ($N \times H'$). To match dimensions, we map tabular embeddings to $\mathbb{R}^H$.

Afterwards, *late-fusion* combines text and tabular embeddings in a way invariant to concatenation order. The resulting combined embeddings Z_{mm} are then used to construct a new tabular dataset alongside the original targets. This dataset is passed to the AutoGluon Tabular predictor. Due to the intricacies and the Neural Architecture Search involved, this pipeline component search proves computationally expensive, taking about 4-5 hours per run on average with a GPU. Despite this, the studied pipelines are regarded as optimal baselines for the 18 datasets in the AutoMM Benchmark. Post-fitting tree-based models and ensembles, a meta-dataset is derived from the Tabular Predictor, containing model names, scalar evaluations, fit time, hyperparameters, stacking information, and default pre-trained model configurations for constructing M. This exhaustive process is iterated across all the 18 datasets present in the AutoMM Benchmark.

Tabular + Text + Vision Modality

Similar to the modality discussed above, we set up the framework for tackling tasks about the tabular + text + vision modality over labelled input data. The main components of the corresponding pipeline architecture are the two evaluated pre-trained models, FLAVA and Albef along with Autogluon's pure NAS method used for approximating the tabular modality. Our focus is to find a general end-to-end pipeline solution over classification and regression tasks by studying our method over The PetFinder Dataset (multi-class classification) and CD-18 (regression) Mobile Dataset.

Let the image data be $X_{\text{images}=\{(x_{\text{img},1}).....(x_{\text{img},N})\}} \ni X_{\text{images}} \in \mathcal{R}^{\mathcal{N}\times\mathcal{I}}, x_{\text{img},k} \in \mathcal{R}^{\mathcal{I}} \ni k \leq N$. Now, let the text data be $X_{\text{text}=\{(x_{\text{txt},1}).....(x_{\text{txt},N})\}} \ni X_{\text{text}} \in \mathcal{R}^{\mathcal{N}\times\mathcal{L}}, x_{\text{txt},k} \in \mathcal{R}^{\mathcal{L}} \ni k \leq N$ and $X_{\text{tab}=\{(x_{\text{tab},1},y_{\text{tab},1}).....(x_{\text{tab},N},y_{\text{tab},k})\}} \ni X_{\text{tab}} \in \mathcal{R}^{\mathcal{N}\times\mathcal{T}}, x_{tab,k} \in \mathcal{R}^{\mathcal{T}} \ni k \leq N$ and $y_{tab,k} \in \mathcal{R}^{\mathcal{C}}, C \in 0, 1..c$ or $\mathcal{R}$. Thus, our multimodal training dataset D_{train} becomes as $D_{\text{train}} = \{((x_{\text{img},1}, x_{\text{txt},1}, x_{\text{tab},1}), y_{\text{tab},1}).....((x_{\text{img},N}, x_{\text{txt},N}, x_{\text{tab},N}), y_{\text{tab},N})\} \ni D_{\text{train}} \in \mathcal{R}^{\mathcal{N}\times\mathcal{I}\times\mathcal{L}\times\mathcal{T}}$. The problem then becomes to find the pipeline $\mathcal{P}_{g,\hat{A}^*,\hat{\lambda}^*}$, such that $\mathcal{P}_{g,\hat{A}^*,\hat{\lambda}^*} : \mathcal{R}^{\mathcal{I}\times\mathcal{L}\times\mathcal{T}} \rightarrow \mathcal{R}^{\mathcal{C}}$ and $\mathcal{P}_{g,\hat{A}^*,\hat{\lambda}^*} : \mathcal{R}^{\mathcal{I}\times\mathcal{L}\times\mathcal{T}} \rightarrow \mathcal{R}$ for the classification and regression tasks respectively.

Fig 3.4 provides an architectural overview of the pipeline designed to fetch pipeline configurations for the classification and regression tasks about the tabular + text + vision modality. Data processing (D_{train}) entails extracting image-text ($x_{\text{img}}, x_{\text{txt}}$) and tabular ($x_{\text{tab}}, y_{\text{tab}}$) attributes. Vision-language pre-trained models handle image-text data, while

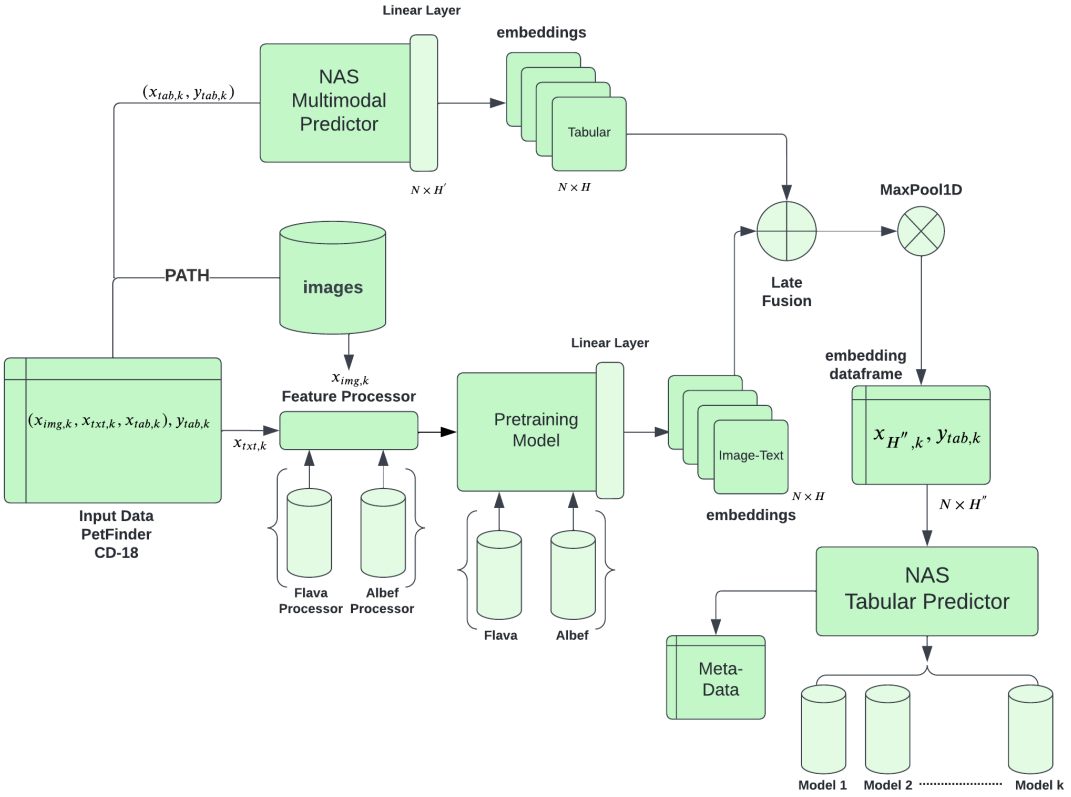


Figure 3.4: Architectural Design of the end-to-end pipeline used to obtain prior evaluations P over the Petfinder and CD-18 datasets for the classification and regression task. $x_{H,k} \in \mathcal{R}^{\mathcal{H}} \ni k \leq N$ is the unified latent representation of dimension H, generated by the pre-training model.

AutoGluon's *multimodal-net* with NAS processes tabular data. For FLAVA and Albef models, encoded image-text data passes through feature processors. Regardless of the model, processed data enters a unified latent space ($N \times H$). The *multimodal-net* maps tabular embeddings into $N \times H'$, then transforms to $N \times H$. Embeddings $\hat{e}$ undergo a late fusion strategy. If the fused embeddings are large ($\geq 50k$), 1D MaxPooling downsamples them to $N \times H''$, maintaining the order invariance post-late-fuse. Processed tabular embedding data $(x_{H''}, y_{tab})$ then feeds into the AutoGluon Tabular Predictor, which processes features and fits J downstream models. Relatively well-performing models (k^*) are chosen from J. Information encompassing performance, fit time, and child hyperparameters is extracted, enriching the meta dataset M. This comprehensive process is repeated for both PetFinder

and CD-18 datasets.

3.2.3 Pipeline Variant 2

We design the second pipeline variant for processing vision-language modalities in a sophisticated manner given the absence of tabular features and labels in the input data. This pipeline architecture deals with tackling unlabelled image-text data used for solving the image-text matching (ITM) task. The aim of the experimentation setup is simple, collect as much information regarding the performance of pipelines consisting of pre-trained FLAVA, ALBEF and various downstream models for the ITM tasks and construct the meta-dataset M. Fig 3.5 represents the architecture of the pipeline used to obtain pipeline evaluations for the ITM task. After acquiring image and JSON annotation files for the Flickr30k and SBU

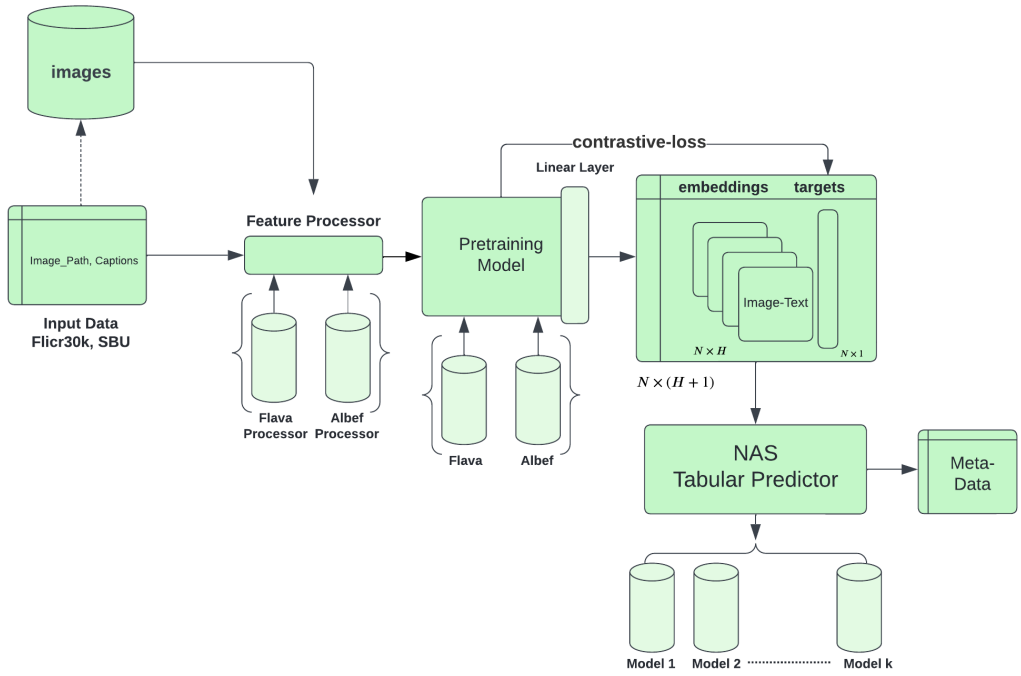


Figure 3.5: Architectural Design of the end-to-end pipeline used to obtain prior evaluations P over Flickr30k and SBU Image Captioning datasets for the Image Text Matching task. The image and text inputs are represented in a unified space of dimension $\mathcal{R}^{\mathcal{H}}$.

image captioning datasets, we aggregate this data into a single CSV file containing image paths and corresponding text captions. In the case of Flickr30K, there are five captions per image. A PyTorch DataLoader is established to process batches of size *batch_size*, typically ranging from 100 to 200 for experimentation. Batches are then directed to relevant feature

processors, determined by the pre-trained model in the pipeline.

For Flava, the FlavaFeatureProcessor, which employs a Vision Transformer (ViT) [6] for image encoding and the Vit/B-16 transformer model for textual encoding, is used. Albef, on the other hand, utilizes a 12-layer visual transformer ViT-B/16 model for image encoding and a 6-attention-head transformer model for text and multimodal encoding. FlavaFeatureProcessor is intended for FLAVA models, while AlbefFeatureProcessor complements Albef Models. Post-processing, encoded inputs enter pre-trained models to generate cross-modal latent representations. An image-text contrastive loss is computed for each image and corresponding caption. However, our objective extends beyond matching; we aim to assess the quality and generalizability of embeddings $\hat{e}$. To achieve this, we design a downstream task $f_l : \mathcal{R}^{\mathcal{H}} \rightarrow \mathcal{R}$, mapping embeddings to contrastive scores. We treat this as a regression problem, with RMSE as the evaluation metric. To optimize f_l and its weights, we employ AutoGluon Tabular. The extracted $N \times H$ embeddings and contrastive loss targets of dimension $N \times 1$ are input to AutoGluon Tabular Predictor, yielding optimal f_l. After completing the Neural Architecture Search and selecting top k^* f_l, we extract a meta dataset M containing meta-features.

3.2.4 Pipeline Variant 3

The third variant proposes a way to tackle the vision-language modality, given the absence of the tabular modality for labelled datasets. To validate this framework, we design experiments for solving a classification task in the form of visual question answering (VQA) over a labelled dataset (VQA2.0) using the proposed pipeline framework. Figure 3.6 provides an overview of the architecture of the pipeline designed to obtain the prior evaluations P for the text + vision modality. The VQA2.0 dataset is provided as JSON files containing image IDs, corresponding questions, and annotations (answers). An accompanying directory holds all listed images. Utilizing a Python crawler, JSON content is parsed and converted into CSV format, yielding tabular data with image paths, associated questions, and target answers, denoted as $(x_{i,k}, x_{q,k}), y_{a,k}$ in Figure 3.6.To process data, a PyTorch DataLoader is constructed to load images, queries, and answer targets in batches of size 100. Data then undergoes feature processing based on the chosen model. FLAVA leverages the ViT/B-16 transformer for image and text encoding, while Albef employs ViT/B-16 for images and a 6-attention head transformer for text and multimodal processing.

Encoded images and questions are fed into FLAVA or Albef models, generating unified latent cross-modal embeddings ($N \times H$). Hidden embedding size for each data point

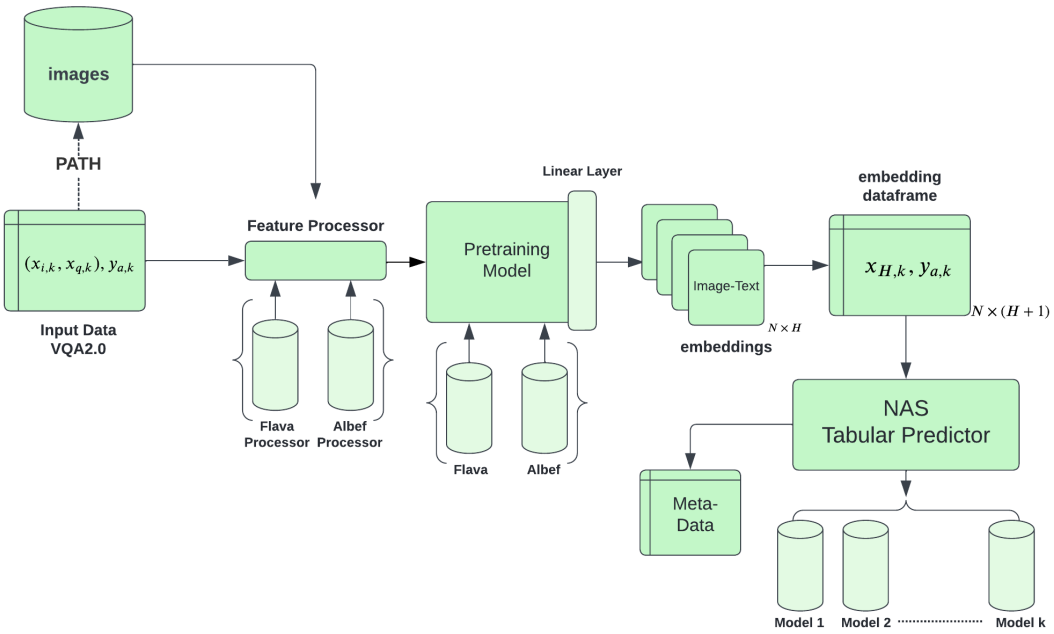


Figure 3.6: Architectural Design of the end-to-end pipeline used to obtain prior evaluations P over the VQA2.0 dataset for the VQA task. $x_{H,k} \in \mathcal{R}^{\mathcal{H}} \ni k \leq N$ is the unified latent representation of dimension H, generated by the pre-training model.

$(k \leq N)$ is $N \times H$. H' represents the default latent embedding size, which is mapped to H through linear transformations. The final embedding size, or hidden size of the last layer, is H. These embeddings $x_{H,1}...x_{H,k}...x_{H,N}$ are concatenated with original targets y_{ans} to form final tabular data $D_{\text{embed}} = (x_{H,1}, y_{a,1})...(x_{H,N}, y_{a,N})$. D_{embed} is then input to AutoGluon Tabular to derive prior evaluations P. AutoGluon Tabular fits around 5-15 (J) downstream models, obtaining zero-shot evaluations. From these, we select k^* models $(k < J)$ based on the objective metric NAUC for classification. Information about these pipelines is included in our meta-dataset M.

3.2.5 Construction of Meta-Data M

The enclosed box 3.2.5 below shows a sampled instance of our meta-data M constructed by performing the prior evaluations across modalities as mentioned above. We refer to the structure of the meta-dataset M as discussed in Chapter 1 while discussing figure 1.2. As seen in this block 3.2.5, a type of task t_j can have multiple (i) configurations by initialising the meta-dataset as a dictionary object. Due to space constraints, we only depict one configuration corresponding to a task t_j in the following meta-dataset 3.2.5, but M consists of multiple evaluations for a task in reality. The *score* key is the evaluated performance $P_{j,i}$ of the i^{th} configuration on the j^{th} task.

```
meta_dataset = {
    'task_clf_clf': {
        'config': {
            'pretraining_task': 'classification',
            'downstream_task': 'classification',
            'pretraining_linear_hidden_size': 256,
            'pretraining_model': 'FlavaFeatureProcessor',
            'pretraining_feature_processor': 'FlavaProcessor',
            'pretraining_attention_dropout': 0.0,
            'pretraining_hidden_dropout': 0.0,
            'pretraining_layer_norm_eps': 1e-12,
            'pretraining_pooling_kernel': 2,
            'downstream_model': 'CatBoostClassifier',
            'catboost_iterations': 100,
            'catboost_depth': 4

        },
        'score': 0.8

    },
    'task_reg_reg': {
        'config': {
            'pretraining_task': 'regression',
            'downstream_task': 'regression',
            'pretraining_linear_hidden_size': 256,
            'pretraining_model': 'FlavaTextModel',
            'pretraining_feature_processor': 'FlavaProcessor',
            'pretraining_attention_dropout': 0.0,
            'pretraining_hidden_dropout': 0.0,
            'pretraining_layer_norm_eps': 1e-12,
            'pretraining_pooling_kernel': 2,
```

```
            'downstream_model': 'weighted_ensemble_r_CAT_L1_2',
            'lgbm_max_depth': 4,
            'lgbm_num_leaves': 130
        },
        'score': 0.256069262627141
    },
    'task_itm_reg': {
        'config': {
            'pretraining_task': 'ITM',
            'downstream_task': 'regression',
            'pretraining_linear_hidden_size': 256,
            'pretraining_model': 'FlavaITM',
            'pretraining_feature_processor': 'FlavaProcessor',
            'pretraining_attention_dropout': 0.0,
            'pretraining_hidden_dropout': 0.0,
            'pretraining_layer_norm_eps': 1e-12,
            'pretraining_pooling_kernel': 2,
            'downstream_model': 'stacked_ensemble_r__RF_L1',
            'lgbm_max_depth': 4,
            'lgbm_num_leaves': 128
        },
        'score': 0.257076860741332
}
```


3.3 Construction of the Configuration Space Θ

To construct the configuration space Θ, prior knowledge regarding the performance of multimodal pipelines on benchmark datasets is necessary. In order to tackle the problem of automatically synthesising end-to-end machine learning pipelines, the task of constructing a configuration space is extremely pivotal. A configuration space Θ is simply a search space that the SMBO algorithm shall span through during the optimisation process. We shall proceed to mention the components of the configuration space i.e. the pre-trained models, feature processors and classical ML models along with its hierarchical structure. Furthermore, we shall also mention the range of the selected hyperparameters. To include the information regarding the models, categorical and numerical hyperparameters, Θ needs to be a hybrid space accounting of both categorical as well as numerical values.

We decided to include FLAVA [40], ALBEF [20] and Data2Vec [2] in Θ for the following reasons. FLAVA provides evidence [40] to perform better than all the above-stated

models (except UNIMO) on multimodal tasks like VQA, VE and IT. It also performs better on most of the language tasks except natural language inference (NLI) and performs better on ImageNet linear evaluation. Thus, we decide to include FLAVA in Θ for the above performance-based evaluation apart from the reasons justified for its architecture. Data2Vec [2] promises a visionary *modality-agnostic* pre-training architecture, which we feel is important for generating *universal* representations and hence we include it in Θ. For additional diversity, Θ shall also comprise Albef [20]. We further include 40 conventional tree-based (RandomForest, Decision Tree) models and classical ML classification and regression models like stacked and weighted tree-based ensemble models in the configuration space Θ. Pre-processing algorithms for feature extraction Transformer models for ViT-based patch features, augmentation algorithms like standard scaler, Imputers, One hot encoder, Byte-Pair-Encoding (BPE) models etc. are also included in Θ [7]. This configuration space Θ can be constructed with the help of tools like GAMA [2], AutoML Tool [3].

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	$[50, 300]$

Table 3.2: Selected hyperparameters and their corresponding ranges in Θ

When spanning through the configuration space to synthesise end-to-end ML pipelines, the Bayesian optimisation-based algorithm should render hyperparameters of

[2]https://openml-labs.github.io/gama/master/benchmark.html
[3]https://tinyurl.com/mryr6sxp

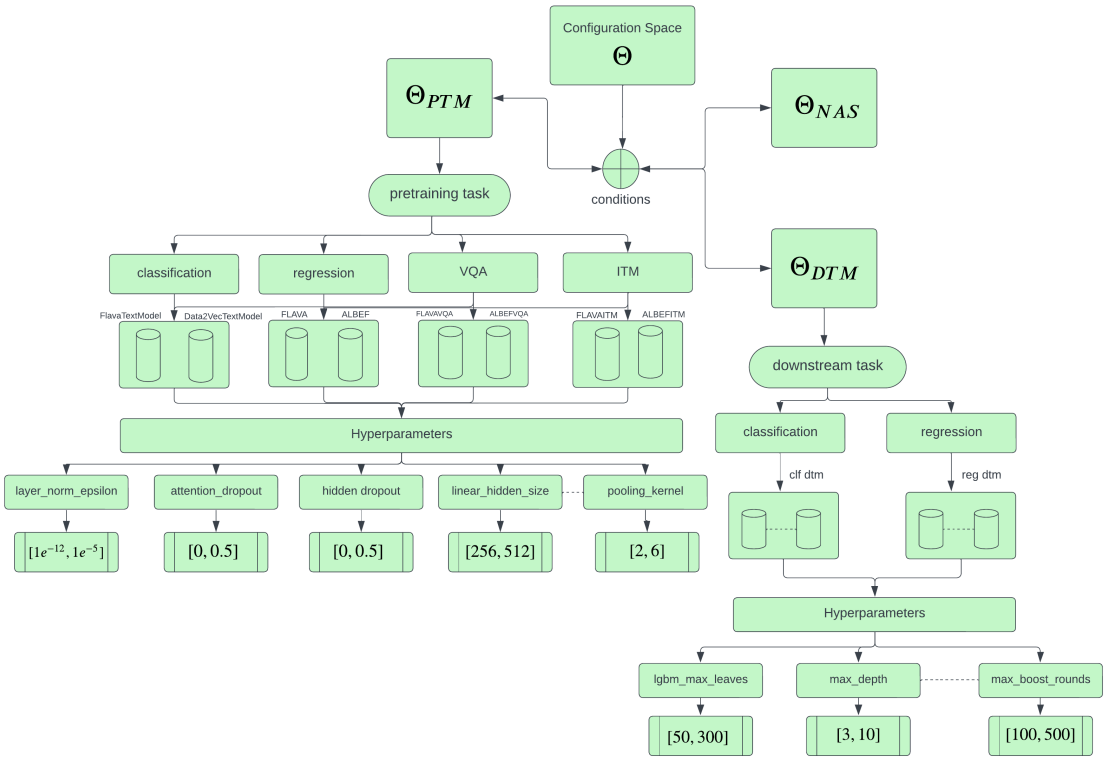


Figure 3.7: Hierarchical Structure of the Configuration Space Θ.

a classification pre-processors As inactive for the models solving a regression problem and vice versa, we refer to Section 1.2, where we formally discussed this formulation. the configuration space Θ_{NAS} refers to the search space of a Neural Architecture Search (NAS) method implemented by Autogluon. Let Θ_{PTM} be a configuration space of the pre-trained models that contain categorical choices for the pre-trained model as ['FlavaTextModel','Albef','FLAVA','FLAVAVQA','ALBEFVQA',
'Data2VecTextModel',
'FLAVAITM','ALBEFITM']. These models are conditioned on the type of task viz. classification, regression, VQA, ITM. When the *pretraining task* hyperparameter is active and set to *classification*, then the regression and ITM pre-trained models are rendered inactive. Similarly, when the pre-training task is set to *regression*, the pre-trained models

for the classification and VQA tasks are rendered inactive. Each of the pre-trained models, corresponding to their respective task categories contain hyperparameters such that only those hyperparameters were selected which were present across all the pre-trained models in Θ. For the pre-training model, the hyperparameters in the configuration space are pretraining layer normalisation epsilon, attention dropout probability, hidden dropout probability, linear hidden dimension size, MaxPooling kernel stride, learning rate, and weight decay. Table 3.2 lists all the hyperparameters and their corresponding ranges. Furthermore, another configuration space Θ_{DTM} is constructed that consists of the downstream models and their corresponding hyperparameters. Θ_{DTM} is conditioned on Θ_{PTM} i.e when the *pretraining task* hyperparameter is *classification* or *VQA* then the downstream models under the downstream task value *regression* are rendered as inactive. Similarly, when the pre-training task value is either *regression* or *ITM*, then the downstream models and their corresponding hyperparameter values under the downstream task value *classification* are rendered inactive. Thus our configuration space is a strict subset, $\Theta \subset \Theta_{NAS} \times \Theta_{PTM} \times \Theta_{DTM}$. The hierarchical structure of the configuration space is depicted in fig 3.7.

After constructing the required Configuration Space Θ, we define the objective function that *actually* generates evaluations by fitting the corresponding model components along with their hyperparameters generated during the CASH procedure.

3.4 Defining the Objective Function

The Objective Function is a user-defined function that computes the objective that is aimed at optimising. It takes a set of hyperparameters as input and returns a scalar value representing the performance metric to be optimized (e.g., accuracy, loss, AUC, etc.) The objective function acts as the bridge between the optimization algorithm and the machine learning pipeline whose hyperparameters are being optimised. We construct the objective function as a real-valued function. We aim to optimize this function using a sequential model-based algorithm configuration (SMAC) process. This function represents the performance or quality metric, typically the AUC (for classification, VQA tasks) or R^2 (for regression-based tasks) that we want to improve by finding the optimal configuration of hyperparameters for a given pipeline. The objective function can be denoted as $f(\theta)$ where f is the function that takes a configuration θ as the input. θ is a vector of hyperparameters and models that defines a particular pipeline configuration g'. 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 text + vision 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

3.5 Setting Up & Warm-Starting SMAC:

Having set up the configuration space Θ and our meta-dataset M, we shall now solve the CASH problem using a sequential model-based algorithm configuration selection (SMAC)

procedure. We make use of the SMAC3 [4] library to set up the Scenario of the experiment, to train a Random Forest model given our constructed meta-dataset M and for warm-starting and selecting the next configurations for our SMBO experimentation. We now explain the components of SMAC and the way they interact to obtain a set of optimal configurations:

1. Scenario $\mathcal{S}$
2. Configuration Space Θ
3. Objective Function f
4. Meta Learner ψ_l
5. Acquisition Function $a_{\mathcal{M}l}$
6. Configuration Selector
7. Intensifier $\mathcal{I}$

Once, we have constructed our meta-dataset M consisting of evaluated pipeline configurations for a task t_j and a scalar evaluation score $P_{(j,i)}$ we proceed with the SMAC procedure. We assess pre-trained multimodal designs along with their weights and merge them into the setup of our AutoML system by performing prior evaluations and then integrating them in a hybrid configuration space Θ. This introduces a way to use pre-trained multimodal models within AutoML to handle data with multiple modalities. SMAC first studies the structure of the configuration space Θ using a Scenario $\mathcal{S}$ component and samples a random configuration within the pre-defined bounds of Θ specified for each hyperparameter to begin the sequential model-based algorithm configuration selection procedure. $\mathcal{S}$ defines the optimization scenario and contains various settings that control the optimization process. This includes the number of optimization iterations, the maximum time or budget for each iteration, the number of initial random configurations, and other configurations related to the optimization process itself. The scenario allows us to customize how SMAC explores the configuration space and how many resources it uses for the optimization. Sampling initial random configurations and performing a cold start can be inefficient in terms of reaching convergence (under a specified budget) for the *optimal* hyperparameter search. Thus, we *warm-start* the sampling of the initial configuration using a meta-learner ψ_l given the meta-data M. For obtaining the initial sample, ψ_l trains on M by splitting M into a disjoint set of M^{train} and M^{eval} in a 3-fold cross-validation manner. For M^{eval} within each cross-validation loop, ψ_l predicts the mean predictive performance and associated uncertainty corresponding to every configuration in the evaluation meta-dataset. The

[4] SMAC3

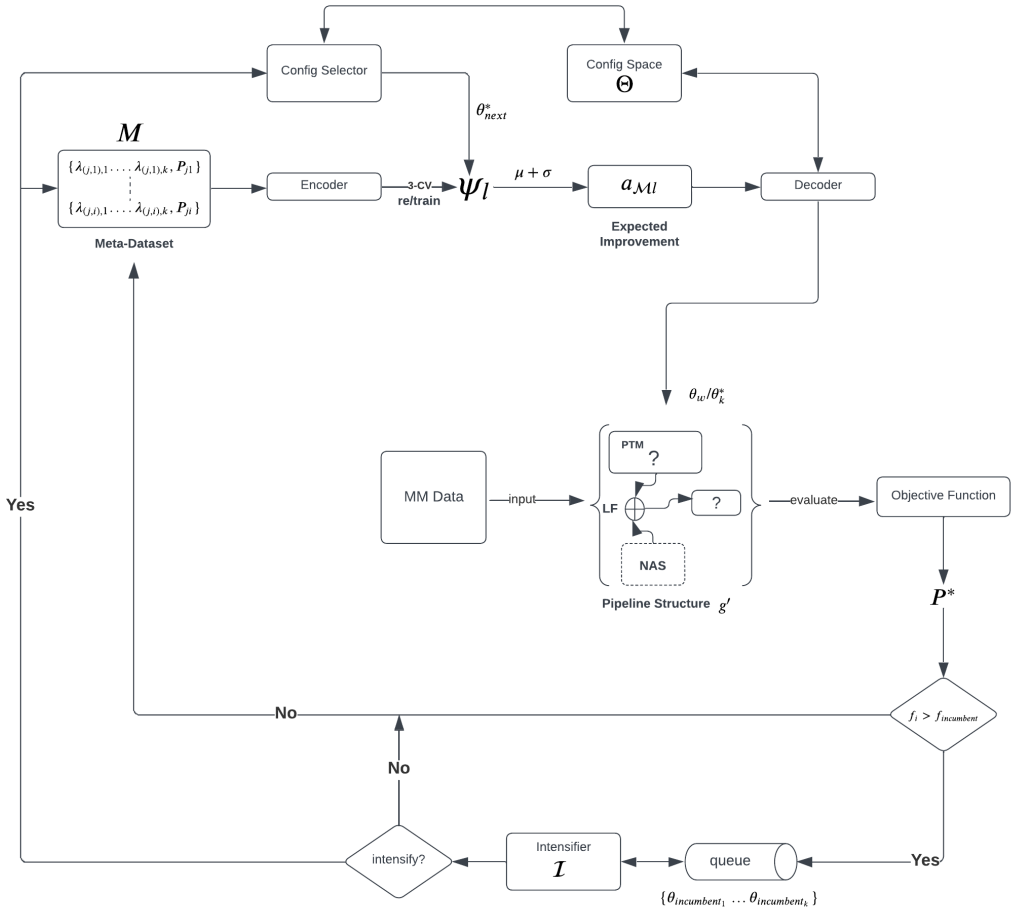


Figure 3.8: Detailed workflow portraying the interactions between various components of the SMAC procedure.

predicted average performance and standard deviation scores by ψ_l are fed to an Expected Improvement (EI) acquisition function $a_{\mathcal{M}l}$. The EI acquisition function then sends this configuration to the objective function that calculates the performance by performing the *actual* evaluation of this configuration. Thus, by maximizing the EI function for all the evaluations in M^{eval} (3-fold CV), a warm-start configuration θ_w is predicted by the acquisition function. By selecting an initial configuration that yields a high-predictive mean and associated uncertainty, the acquisition function balances the exploration and exploitation trade-offs. Having, sampled θ_w using ψ_l and $a_{\mathcal{M}l}$, the actual evaluation is obtained by passing θ_w to the defined objective function. The objective function returns a real-valued scalar (AUC or R^2) that captures the performance of the configuration being evaluated. The initial warm-starting configuration is treated as the first incumbent configuration. This evaluation is then appended in the meta dataset M and ψ_l is retrained.

The optimisation process aims to maximise the acquisition function by sampling a set of configurations in a guided manner. Having evaluated θ_w, SMAC then selects the configuration that has the highest Expected Improvement value. This is done by re-training ψ_l and again obtaining evaluations. This configuration is the one that is predicted to lead to the greatest performance improvement, according to the meta-learner ψ_l. The RunHistory component in SMAC3 helps in logging the track record of all previously evaluated configurations. SMAC repeats this process till it obtains an objective value (real evaluation) greater than that of the previously observed incumbent. If this is the case then the Intensifier component adds these evaluations to its queue for *intensifying* these configurations. The intensifier then calls the configuration selector object of the configuration space Θ, which samples configurations around the incumbent. To sample configurations around the incumbent, the configuration selector applies a perturbation strategy to the incumbent configuration. This strategy involves making small changes to the incumbent configuration's hyperparameter values. These changes are typically random and controlled to ensure that the perturbed configurations are within the defined configuration space. The magnitude of perturbation (perturbation factor) determines how far the perturbed configurations will be from the incumbent. The perturbation factor is defined as a small percentage [10% - 30%] of the hyperparameter's domain width. The idea behind using a percentage of the domain width is to ensure that the perturbed configuration remains within a reasonable range while exploring the space around the current configuration. For example, we have a continuous hyperparameter with a valid range [0, 1], and the default perturbation factor is set to 10%, the perturbed configuration will be sampled within the range [0.1, 0.9], which is 10% away from the current value. The

intensifier applies an Aggressive Racing strategy, where it calls the configuration selector to sample n configurations around the incumbent. These n sampled configurations are then fed to ψ_l to obtain their predictive mean performance and associated uncertainty value. The EI acquisition function is maximised over these n evaluations and of these n evaluations, it selects k configurations ($k < n$) for obtaining their actual evaluations. SMAC calls the objective function, where these k configurations are evaluated in parallel. For these k evaluations if $f_k > f_{\theta_{\text{incumbent}}}$, then add it to the intensifier queue, append to the meta-dataset M and retrain ψ_l. Fig 3.8 portrays the workflow of the SMAC procedure, given its components. This figure explains in detail how the acquisition function is maximised by selecting hyperparameters in a guided fashion.

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.

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: $M \leftarrow M \cup \{(\theta_\text{w}, f_{\theta_\text{w}})\}$
    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 $a_{\mathcal{M}l}(\theta_i) = \text{EI}(\mu_i, \sigma_i, \text{incumbent\_value})$
    end for
    Select the configuration with the highest acquisition value: $\theta_\text{next} = \arg\max_i a_{\mathcal{M}l}(\theta_i)$
    Evaluate θ_next in true objective function and get f_{θ_next}
    Update M: $M \leftarrow M \cup \{(\theta_\text{next}, f_\text{next})\}$
    Update surrogate model ψ_l using updated run history M
    if $f_\text{next} > f_\text{incumbent}$ **then**
      Update incumbent: incumbent_value $\leftarrow f_\text{next}$, $\theta_\text{incumbent} \leftarrow \theta_\text{next}$
    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

3.6 Metric for Optimisation Evaluation

Once the SMAC optimisation loop converges successfully, the next task becomes to evaluate the optimisation process. Selecting a metric to evaluate the optimisation process is extremely pivotal. An appropriate metric should be such that it provides an empirical value to compare the quality of learning of a given pipeline configuration for a task on a dataset and also captures the learning behaviour of the configuration selection algorithm.

We mentioned using AUC and R^2 scores as scalars returned by the objective function. To evaluate the quality of learning behaviour, we compute an 'any-time' learning metric. To do that, we first normalise the AUC scores obtained for the classification tasks as follows:

$$NAUC = (2 * roc\_auc) - 1 \tag{3.1}$$

To encourage learning in a short time budget and with limited data, we intend to use an "any-time learning metric", taking inspiration from Liu et al. [29]. Thus, as done by Liu et al. [29], we define the Area Under Learning Curve (ALC) as :

$$ALC = \int_0^1 s(t)d\tilde{t}(t) = \int_0^T s(t)\tilde{t}'(t)dt = \frac{1}{log(1+\frac{T}{t_0})}\int_0^T \frac{s(t)}{t+t_0}dt \tag{3.2}$$

The normalised Area Under the Curve (NAUC) is computed by averaging the performance over all the classes. Multi-class classification metrics are not being considered, i.e. e. each class is scored independently. Since several predictions could be made during the learning process, it enables us to plot the learning curve as a function of time as suggested by Liu et al. [29]. The time axis is log scaled (with time transformation $\tilde{t}$) to put more emphasis on the beginning of the curve [29]. For each dataset, we compute the Area Under Learning Curve (ALC) to evaluate the learning (optimisation) behaviour of our AutoML system.

Chapter 4

Experiments & Results

In the previous chapter, we explained our methodology for generating *optimal* pipeline components consisting of a pre-trained model for processing multimodal data. We design and conduct experiments to find answers to our two research questions, **RQ1.** and **RQ2.** discussed in the previous chapters. We conducted experiments to implement our methodology and to resolve the claims surrounding the hypotheses made during this research. To make the structure of our research clear to the reader, we divide this section first into a subsection that discusses the experimental design of the experiments. This includes discussing the objective of our experimentation, and analysis of the datasets over which the experiments are to be conducted followed by the analysis of fusion strategies to implement the fusion method in the pipeline structure g'. Following this, we also mention the feature processing steps performed over the data in the given pipeline. Having done this, we move on to discuss the results obtained while performing the prior evaluations across pipeline configurations for each of the three modalities. Given the prior evaluations and warm-starting SMAC using ψ_l and M, a set of incumbent configurations for each task over respective datasets across the 3 modalities is obtained. We study the optimization process that yields these incumbent configurations along with the associated performance and efficiency of the method by comparing the performance against a pure NAS-based method and generating incumbents that give better performance as compared to the configurations in M. Variants of the pipeline architecture are designed to process the tabular + text, text + vision, and tabular + text + vision modalities over tasks such as Image Text Matching (ITM), Visual Question Answering (VQA), classification and regression.

4.1 Experiment Design

4.1.1 Objective & Setup of the Experiments

In this subsection, we shall state the objective of designing and conducting the experiments mentioned further in this chapter. We stated the goal of our research in the form of **RQ.1** and **RQ.2**. Thus to evaluate these goals, we design and perform a set of experiments by applying the previously mentioned methodology. The goal of the experiments is stated as follows:

- Goal of Prior Evaluations: We formulated **RQ.1** as an exploratory question to explore the integration of pre-trained models in the AutoML system for performing AutoML over multimodal data. To explore this, we first design a set of experiments to evaluate pipeline architectures on different tasks and modalities. The aim is to confirm the findings stated in seminal benchmark literature and design pipeline architectures that are well-motivated to collect information regarding the prior evaluations of these frameworks on different multimodal datasets comprising different tasks. Having studied well-performing pipeline architectures comprising pre-trained models, the goal is to construct a meta-dataset M. M is used to train the meta-learner ψ_l, whose predictions are used for finding the warm-start configurations as well as the final optimal set of configurations for a given task. Furthermore, having performed these prior evaluations, performing a hyperparameter optimisation on these exact structures helps us in understanding the performance of our SMBO procedure.

- Goal of SMAC optimisation: Once M is constructed and ψ_l trained on this, the goal of the procedure is to generate well-performing and well-motivated configurations for starting the search of the Bayesian Optimisation process. The next objective is to analyse and understand the optimisation behaviour of these SMBO experiments having warm-started this search procedure. We then understand the performance score of the incumbent configurations generated by the SMAC procedure and compute the ALC metric under the given budget to evaluate the observed learning procedure.

We now explain the setup of the two kinds of experiments we perform in this research:

- Setup of Prior Evaluations: To perform prior evaluations for the 3 mentioned modalities across classification, regression, ITM and VQA tasks we build pipeline architectures as explained in Chapter 3. We use the AutoGluon *multimodal-net* for processing the tabular modality. *multimodal-net* employs a Neural Architecture Search method [10, 9]

for constructing a Multi-Layered Perceptron called *multimodal-fusion-mlp* consisting of 109 million parameters. Furthermore, to automatically assess models for mapping the multimodal universal latent representations, we use the Tabular Predictor class from AutoGluon for constructing tree-based ensemble (shallow ML) models. Each evaluation over the 18 AutoMm datasets, Flickr30k, SBU Image Captioning Data, VQA2.0 dataset, Petfinder Dataset and CD-18 dataset is performed for $\approx 5-6$ hours. Thus by setting up a greedy NAS-based search, we ensure that the evaluated pipeline configurations are well-performing, least sub-optimal or even optimal. Moreover, the pre-trained models used for obtaining the prior evaluations are initialised using their default hyperparameter choices and pre-trained weights. Table 4.1 provides an overview of some of the initialised hyperparameters such as **layer_no_e**: layer_norm_epsilon, **atdo**: attention_dropout, **hid_dim**: hidden_dimension in experiment, **lin_hid_siz**: linear_hidden_size, **nah**: number of attention heads, **nhl**: number of hidden layers, for the evaluated pipelines.

	FLAVA			Data2Vec		
	layer_no_e	atdo	hid_dim	layer_no_e	atdo	hid_dim
data: cloth						
WEnsemble_L2	$1e-6$	0.0	512	$1e-12$	0.0	512
data: wine						
ETMSE_BAG_L2	$2e-4$	0.0234	433	$1e-5$	0.0467	264

Table 4.1: Instances of pipeline hyperparameter configurations. The first instance in the table is the default configuration while the second one is the instance of an optimised pipeline configuration on a subset of hyperparameters.

- SMAC Setup: During our experimentations, we restricted the evaluation time taken by each trial to 20 minutes, taking inspiration from Liu et al. [29]. The wall clock time is set to 45 minutes. This implies that the entire optimisation may run at most up to 45 minutes and, each trial can run up to a maximum of 20 minutes. Trials that do not finish their run within this allocated budget are treated as *crash* and the objective value of such configurations is set to inf. Thus it may be completely possible that out of the set of Γ submitted trials only a subset of trials γ might successfully terminate, implying $\gamma \subseteq \Gamma$.

	Hyperparams			Hyperparams		
	layer_no_e	atdo	hid_dim	lin_hid_siz	nah	nhl
data: Fkr30k, SBU						
FLAVA	$1e-12$	0.0	768	512	12	12
data: Fkr30k, SBU						
ALBEF	$1e-10$	0.0	768	256	12	12

Table 4.2: Task: **Image Text Matching**. Default hyperparameter configurations of FLAVA and ALBEF models.

	Hyperparams			Hyperparams		
	layer_no_e	atdo	hid_dim	lin_hid_siz	nah	nhl
data: VQA2.0						
FLAVA	$1e-12$	0.0	768	512	12	12
data: VQA2.0						
ALBEF	$1e-10$	0.0	768	256	12	12

Table 4.3: task: **Visual Question Answering (VQA)**. Default hyperparameter configurations of FlAVA and ALBEF models.

4.1.2 Analysis of Available Multimodal/Unimodal Datasets

Unimodal ML systems have been studied vastly in the last decade. There is tremendous publically available unimodal tabular, text and image data. We first study different sources and their available datasets for deciding the inclusion of relevant datasets in this study. OpenML itself consists of 5092 publically available datasets [1]. Furthermore, huge corpora of image-only datasets are available like CIFAR [2] contains 80 million labelled 32×32 images across 10 classes. Also, datasets like MSCOCO [26] consist of 328K images and can be used for image segmentation and object detection unimodal tasks. MNIST [19] is another publically available dataset of handwritten digits consisting of 600K images. A huge repository of other image-only and text-only datasets can be found here [3]. Although this remains true, publically available repositories of well-benchmarked multimodal datasets

[1] https://www.openml.org/search?type=data&sort=runs&status=active
[2] https://www.cs.toronto.edu/ kriz/cifar.html
[3] https://paperswithcode.com/datasets?mod=images

are scarce. The challenge we faced was to find evidence of some multimodal benchmark datasets that have been evaluated over different pre-trained multimodal models. We present this evidence of good empirically evaluated multimodal datasets which is discussed next.

Shi et al. [10] assemble 18 multimodal datasets about the tabular + text type of modality. They claim that these datasets originate from real-world business applications and make them publicly available [4]. The purpose of these datasets is to ensure that any modelling strategy for combating the tabular + text type of modality, that performs well over all these 18 datasets will serve as a practical foundation for tabular + text multimodal AutoML. Figure 4.1 provides an overview of the 18 publically released datasets by Shi et al.

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

Figure 4.1: An overview of the 18 datasets proposed by Shi et al. [10] 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.

Furthermore, they claim *enough* diversity regarding these datasets in terms of sample size, problem types, number and the types of features. 11 of these 18 datasets contain multiple text fields with these text fields varying in terms of length and nature of these texts (for eg. short and lengthy product reviews). Furthermore, they provide each dataset with a $80-20$ hold-out split, with the 20% reserved as *test data*. Further results in their work [10] also show how the predictive signal is decomposed and divided into the text and tabular modalities. They claim that methods and systems that perform well across these diverse sets of 18 datasets are likely to provide real-world value for an important class of applications. Thus, having this class of benchmark datasets of tabular + text class of

[4] https://github.com/sxjscience/automl_multimodal_benchmark

modality augments our research in investigating pipeline architectures that perform well across these datasets. Having ML pipelines that perform well across these datasets provides a sense of generalization across different tasks.

Having studied publically available datasets for tabular + text inputs, we provide evidence of publically available datasets that have been empirically evaluated for multimodal Image + Text, Image + Tabular and Image + Text + Tabular tasks. Ferraro et al. [11] provide an extensive empirical survey on current publically available datasets that have become a benchmark in analysing the performance of current vision-language systems. The quality of a dataset is highly dependent on the sampling and scraping techniques it underwent during its construction. Datasets are usually affected by *reporting bias* and *photographer's bias*, where photographs in a domain are somewhat predictable [11]. Thus, it becomes important to include datasets in our study that have been quantitatively studied on metrics that capture these *biases*. For vision datasets, the quality of vision can be attributed to the variety of visual subjects and the provided scenarios as well as to the richness of their annotations. Language quality can be assessed using metrics like **Vocabulary Size** *(#Vocab)*, **Syntactic Complexity**, **Part-of-Speech Distribution**, **Abstract:Concrete Ratio** *(#Conc, #Abs, %Abs)*, **Average Sentence Length** (Sent Len) and **Perplexity** *(ppl)* [11].For generalising over vision-language tasks, we decided to include the VQA [1], SBU [32], Flickr30k and Flickr8k [35] datasets for Visual Question Answering and Image-Text Retrieval tasks respectively. Publicly accessible datasets that

		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

Figure 4.2: Summary statistics and quality metrics of a sample of major datasets provided by Ferraro et al. [11].

encompass the integration of tabular, text, and vision modalities are presently scarce. Furthermore, there exists a dearth of comprehensive literature that rigorously examines the efficacy and quality of such datasets empirically. This is underscored by the utilization

of AutoML frameworks like AutoGluon [9], as evidenced by its application to datasets such as PetFinder [5], wherein a confluence of tabular, text, and vision data is leveraged to enhance predictive performance dataset, which motivates us on using this data for our study. PetFinder is a public dataset intended to build *good* models to predict the *pet adoption* rate. It is a multimodal dataset that spans different modalities like image, text and tabular signals. Furthermore, to diversify the tasks for this modality we study another multimodal tabular dataset released by Salmasi et al. [48]. The CD-18 [48] [6] dataset for mobile phone price prediction consists of tabular, text and vision features that contribute towards the descriptions of a smartphone model. We integrate this task of predicting mobile phone prices based on these multimodal descriptive features for studying the performance of our process over a regression task corresponding to this modality.

4.1.3 Analysis of a Fusion Strategy

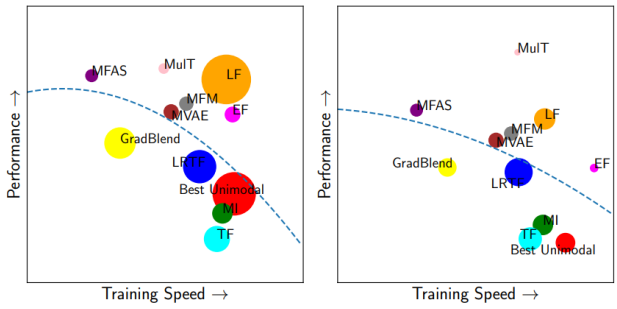
In Chapter 3 we mentioned performing a *late-fuse* on the respective high-dimensional embeddings obtained from the NAS method (if the tabular modality is present) and the pre-trained multimodal model. In this subsection, we delve into the reasons behind selecting this particular design choice for the *fusion* of multimodal information.

Learning multimodal representations is a challenging task. The research on this mainly involves methods of integrating or *fusing* information in a comprehensive and relevant manner. Liang et al. [25] state that this particular field of research has seen limited focus and challenges on generalizations across domains and modalities remain widely un-tackled. Questions on robustly handling noisy and missing modalities and the complexity during their inference and training remain open. Taking this motivation into account, Liang et al. release MULTIBENCH [25], a systematic and unified large-scale benchmark for multimodal learning that spans 15 datasets, 10 modalities, 20 prediction tasks and 6 research areas. In this work, they study various multimodal tasks across real-world datasets and provide rich insights on some common information *fusion* paradigms. Let x_1, x_2 be some input vectors for diffrent modalities, z_1, z_2 be their respective unimodal representations and z_{mm} be their multimodal representation. Liang et al. [25] study the following fusion paradigms:

1. Early Fusion : $z_{mm} = [x_1, x_2]$, concatenation of input vectors.

2. Late Fusion : $z_{mm} = [z_1, z_2]$, concatenation of respective unimodal representations.

[5] https://www.kaggle.com/competitions/petfinder-adoption-prediction/data
[6] https://github.com/AidinZe/CD18-Cellphone-Dataset-with-18-Features



(a) All datasets (b) Datasets with > 6 approaches

Figure 4.3: Tradeoff between performance and robustness observed by Liang et al. [25]. The size of the circles shows the variance in robustness across datasets. The best linear fit is denoted by the blue line. Better-performing models show better robustness.

3. Multiplicative Interactions (MI) : $z_{mm} = z_1 \mathbb{W} z_2 + z_1^T \mathbb{U} + \mathbb{V} z_2 + b$ where $\mathbb{W}, \mathbb{U}, \mathbb{Z}$ and b are trainable parameters.

4. Multimodal gated units: $z_{mm} = z_1 \odot h(z_2)$, where h represents a function with sigmoid activation and $\odot$ represents element wise product.

5. Temporal Attention Models: Alignment using Transformer models, where unimodal vectors are aligned using self-attention layers of Transformer models. $z_{mm} = [z_{1\to2}, z_{2\to1}] = [CM(z_1, z_2), CM(z_2, z_1)]$, where CM represents Cross-Modal attention.

They conduct experimentation across their 15 multimodal datasets using this information *fusion* paradigms. They further compare several other fusion methods like the multimodal model architecture search method MFAS [47], Cross-Correlation Analysis (CCA) [41], and Multimodal Transformers (MuLT). From Figure 4.3 as revealed by the experimentation of Liang et al. [25], a simple fusion paradigm like Late Fusion (LF) is an appealing choice that scores high on both the metrics (performance and robustness), especially when compared with complex yet slightly better performing methods like MFAS or MuLT. Furthermore, LF is found to be the easiest information fusion paradigm, that adapts easily to new datasets across different domains [25]. Also, from figure 4.3 LF has a positive correlation between performance and robustness implying that a model implementing LF which achieves high accuracy tends to stay above other models on the performance-imperfection curve. LF has an advantage over MFAS and CCA, as both face scalability issues. They are intrinsically designed only for 2 modalities, usually image and text [25].

Method	prod	qaq	qaa	cloth	airbnb	ae	mercari	jigsaw	imdb	fake	kick	jc	wine	pop	channel	salary	book	house	avg ↑	mrr ↑
Choosing Text-Net:	NLP Backbones and Fine-tuning Tricks (Section A.1)																			
RoBERTa	0.588	0.412	0.268	0.700	0.344	0.953	0.561	0.960	0.731	0.929	0.751	0.615	0.811	-0.000	0.301	0.396	0.151	0.821	0.572	0.07
ELECTRA	0.705	0.410	0.356	0.718	0.349	0.955	0.586	0.965	0.750	0.824	0.754	0.606	0.813	0.003	0.315	0.457	0.466	0.857	0.605	0.09
+ Exponential Decay $\tau = 0.8$	0.728	0.436	0.431	0.743	0.337	0.953	0.579	0.963	0.852	0.963	0.760	0.664	0.808	0.004	0.308	0.447	0.568	0.841	0.632	0.12
+ Average 3 ★	0.729	0.451	0.432	0.746	0.350	0.954	0.581	0.965	0.858	0.961	0.766	0.656	0.807	0.004	0.307	0.445	0.571	0.841	0.635	0.14
Choosing Multimodal-Net:	Fusion Strategy (Section A.2, Figure S2)																			
All-Text	0.907	0.454	0.419	0.746	0.366	0.957	0.599	**0.967**	0.840	0.967	**0.799**	0.645	0.810	0.013	0.480	0.465	0.585	0.892	0.662	0.28
Fuse-Early	**0.913**	0.441	0.418	0.745	0.377	0.953	0.596	**0.967**	0.843	0.960	0.770	0.653	0.806	0.013	0.474	0.458	0.548	0.901	0.658	0.21
Fuse-Late ★	0.907	0.449	**0.445**	0.747	0.395	0.958	0.603	0.966	0.857	0.961	0.773	0.639	0.812	0.015	0.481	0.468	0.571	0.907	0.664	0.22
Choosing Aggregation:	Multimodal Model Aggregation (Sections A.3 and A.4, Figure S3)																			
Pre-Embedding	0.895	0.216	0.247	0.642	0.449	0.972	0.433	0.586	0.871	0.926	0.743	0.491	0.680	0.012	0.526	0.460	0.581	0.939	0.593	0.11
Text-Embedding	0.867	0.446	0.432	0.748	0.430	0.972	0.434	0.587	0.855	0.962	0.790	0.658	0.830	0.008	0.502	0.438	0.594	0.932	0.638	0.17
Multimodal-Embedding	0.907	0.439	0.437	0.749	0.438	0.974	0.432	0.587	0.847	0.967	0.794	**0.683**	0.829	0.007	0.517	0.451	0.595	0.934	0.644	0.25
Weighted-Ensemble	0.907	0.439	0.429	0.744	0.453	0.976	0.597	0.957	0.876	0.923	0.787	0.641	0.814	0.018	0.554	0.483	0.620	0.941	0.676	0.25
Stack-Ensemble ★	0.909	**0.456**	0.438	**0.751**	0.459	0.977	**0.605**	**0.967**	**0.878**	0.964	0.797	0.624	0.836	**0.020**	**0.556**	0.496	**0.638**	**0.943**	**0.684**	**0.69**
	Tabular AutoML + Feature Engineering Baselines (Section A.3)																			
AG-Weighted	0.891	0.046	0.076	-0.002	0.426	0.841	0.098	0.587	0.845	0.686	0.668	0.004	0.173	0.016	0.549	0.226	0.222	0.934	0.405	0.08
AG-Stack	0.891	0.046	0.077	0.001	0.435	0.841	0.098	0.587	0.844	0.697	0.670	0.003	0.175	0.017	0.550	0.226	0.233	0.934	0.407	0.09
AG-Weighted+ N-Gram	0.892	0.426	0.382	0.610	0.450	0.978	0.526	0.909	0.842	0.966	0.772	0.357	0.829	0.019	0.546	0.484	0.591	0.941	0.640	0.17
AG-Stack+ N-Gram	0.895	0.414	0.383	0.654	**0.466**	**0.979**	0.569	0.915	0.850	**0.968**	0.775	0.612	**0.842**	**0.020**	0.548	0.494	0.600	0.943	0.663	0.43
H2O AutoML	0.869	0.247	0.159	0.163	0.329	0.976	0.430	0.531	0.813	0.756	0.669	0.411	0.478	0.014	0.530	0.525	0.444	0.939	0.516	0.11
H2O AutoML + Word2Vec	0.859	0.244	0.285	0.624	0.347	0.973	0.534	0.847	0.827	0.943	0.755	0.443	0.778	0.013	0.524	**0.528**	0.586	0.932	0.613	0.14
H2O AutoML + Pre-Embedding	0.846	0.227	0.312	0.644	0.367	0.969	0.282	0.572	0.874	0.893	0.738	0.549	0.571	0.007	0.483	0.483	0.523	0.933	0.572	0.09

Figure 4.4: Accuracy Score over different AutoML strategies proposed by Shi et al. [10]

The conjecture that *Late-Fuse* (LF) works best is supported by Shi et al. as well in their work [10] where they automatically synthesise end-to-end ML pipelines for tabular + text modalities. **Thus, based on this literary evidence it makes sense to adopt a *Late-Fusion* method in our pipeline architecture for aligning the information rooted in different modalities or multimodal signals**.

4.1.4 Feature Processing

Our approach involves utilizing pre-trained models with predefined feature processors that encode input data for compatibility with these models. In our implementation, we construct the pre-trained model class using PyTorch, while initializing the pre-trained weights and architectures through the Hugging Face library. This approach limits us to using uniform feature processors across different pre-trained models. The underlying assumption here is that these feature processors are designed to encode information in a manner that optimally complements the architecture of the corresponding pre-trained model. Thus for the FLAVA pre-trained model, we use only the FlavaFeatureProcessor. Similarly, for ALBEF and Data2Vec we use the AlbefFeatureProcessor and Data2VecFeatureProcessor respectively. The FLAVA processor implements a Word Level Processing tokenization strategy employed by a WordPieceProcessor for feeding the raw input to the ViT-B16 transformer. WordPieceProcessor breaks down input text into subwords, often consisting of partial words or smaller meaningful units. The tokenization process involves creating a vocabulary of subwords from the training corpus, and then mapping words in the input text to these subword tokens. For example "AI is great!" is tokenized as ["##AI", "is",

"great","!"]. Feature Processors of ALBEF and Datavec use an AutoTokenizatio strategy. AutoTokenizer, as the name suggests, automatically selects the appropriate tokenizer based on the provided pre-trained model. The tokenization strategy employed by AutoTokenizer is determined by the specific model being used. We set AutoTokenizer to make use of the Byte Pair Encoding (BPE) strategy to tokenize the input text. The main idea behind BPE is to iteratively merge the most frequent pairs of characters or character sequences in the input text until a specified vocabulary size is reached.

During the prior evaluations, we analysed the performance over an evaluation metric which can be said as a proxy for the predictive signal contained in the late-fused (or not) universal latent representation of the multimodal input signals. As a result of late-fusing high-dimensional embeddings about the tabular and vision-language modalities, some noise might occur in the final multimodal representation. Thus, we apply feature processing and generation techniques to these unified representations before feeding them to the corresponding downstream models. This processing is uniform and includes choices such as whether to keep features of 'int' and 'float' raw types. We use the feature processor implemented by AutoGluon called the AutoMLPipelineProcessor. The purpose of such a feature generator is to transform data from one form to another in a stateful manner. We first initialise the generator with various arguments that dictate the way features are generated. Then, the generator is fit through either the .fit() or .fit_transform() methods using universal latent representations typically in a pandas DataFrame format.

4.2 Results: Prior Evaluation

In this subsection, we present our analysis of the prior evaluations obtained for the tabular + text, text + vision and tabular + text + vision modality. We implement the experiments designed as per the experimentation design discussed above and implement the methodology presented in Chapter 3 for conducting these experiments. The prior evaluations are obtained over the 3 modalities for classification, regression, image-text-matching and visual question-answering tasks using the 3 pipeline architectures discussed in Chapter 3.

4.2.1 Prior Evaluation: Tabular + Text Modality

We set up the first variant of our pipeline structure g' for processing the tabular + text modality. The experiments are conducted over the 18 AutoMM datasets mentioned in our experiment design subsection. These datasets comprise binary, multi-class classification

and regression tasks. We evaluate the pipeline performance by calculating the Area Under Curve (ROC-AUC) for the classification tasks and R^2 score for the regression tasks.

FLAVA						
	Score		Pred Time(s)		Fit Time(s)	
Data	σ^2_{wc}	μ_{wc}	σ^2_{wc}	μ_{wc}	σ^2_{wc}	μ_{wc}
jigsaw	0.000044	0.314	71679.456	474.730	2627820.755	1483.255
qaq	0.000200	0.370	550000.200	165.370	8500000.200	2900.370
pop	0.000230	0.401	1065936.202	321.940	7929658.512	378.124
mercari	0.000210	0.520	550000.210	160.520	8500000.210	3800.520
wine	0.000002	0.564	2273.136	56.765	4532810.510	7801.555
book	0.000019	0.437	60945.122	244.285	1801.200	1207.643
fake job	0.000034	0.193	291444.235	659.678	3990672.259	3405.227
ae	0.000048	0.328	255260.230	480.929	4099846.759	3181.522
qaa	0.000210	0.280	450000.210	175.280	7500000.210	3500.280
california	0.000087	0.341	159827.760	252.586	3668.226	3266.053
house	0.000220	0.430	600000.220	205.430	9000000.220	3300.430
imdb	0.000011	0.323	288.593	180.533	15391.566	8110.366
kick	0.000006	0.304	580.323	168.309	6724575.418	10242.138
channel	0.000260	0.417	421024.394	16.800	9230152.588	324.848
product	0.000221	0.270	542264.896	331.691	12042405.725	15962.631
jcPenny	0.000055	0.298	52213.975	363.157	5284.575	4802.834
airbnb	0.000240	0.390	500000.240	185.390	8000000.240	2700.390
cloth	0.000173	0.591	2572167.226	153.053	2280984.365	3186.961
salary	0.000271	0.388	660804.886	187.209	8499147.032	3013.924
Average	0.000139	**0.366** ⋆	399251.579	**239.357** ⋆	6438015.599	4480.178

Table 4.4: FLAVA - Variance and Mean Across 18 AutoMM datasets. Variance (σ^2_{wc}) and mean (μ_{wc}) are shown for scores, prediction times, and fit times.

In our experiments, we convert the multi-class classification problem into a multi-label binary classification problem and compute the average ROC-AUC metric to evaluate the performance of a pipeline configuration. The experiments as mentioned before are performed using FLAVA and Data2Vec pre-trained models initialised using their default hyperparameters and pre-trained weights. We use AutoGluon's NAS-based multimodal net comprising 109 million parameters to process the tabular embeddings, which are then late-fused with the universal latent representation of the text modality, generated by a pre-trained model. Furthermore, we use Autogluon's NAS methods for automatically evaluating tree-based ensemble models for fine-tuning the multimodal representations over the input

Table 4.5: Data2Vec model evaluation results for various datasets from the AutoMM Benchmark. Variance (σ^2_{wc}) and mean (μ_{wc}) are shown for scores, prediction times, and fit times.

Data2Vec						
	Score		Pred Time(s)		Fit Time(s)	
Data	σ^2_{wc}	μ_{wc}	σ^2_{wc}	μ_{wc}	σ^2_{wc}	μ_{wc}
pop	0.0122	0.6024	40063.942	365.343	29240324.749	4422.472
channel	0.0083	0.4210	556.349	16.422	265308.755	726.756
product	0.0011	0.2950	2878.091	226.815	8726524.487	2612.783
salary	0.0013	0.4097	366.736	70.071	329222.496	1382.618
cloth	0.0002	0.5708	4092.842	149.343	802735.603	3706.722
wine	0.0013	0.3430	3581.426	167.117	1522433.868	1522.845
fake job	0.0249	665.166	3432.821	0.188	222.770	3257.646
imdb	0.0197	173.333	9105.966	0.172	210.771	12703.483
jigsaw	0.0177	474.944	1460.789	0.287	289.417	3009.011
kick	0.0123	174.962	6834.045	0.212	260.620	13461.275
ae	0.0122	457.845	3639.768	0.318	873.446	11079.678
book	0.0225	177.649	1183.363	0.384	199.376	1319.848
california	0.0203	313.869	2996.421	0.286	404.892	3310.725
jcpenny	0.0181	339.913	3064.090	0.259	422.406	4054.593
qaa	0.0086	0.236	2368.214	0.167	2456.336	2734.789
qaq	0.0092	0.278	2753.366	0.196	2567.423	2967.433
mercari	0.0067	0.125	1378.256	0.098	1234.576	1467.311
airbnb	0.0055	0.092	956.332	0.074	865.256	1108.543
house	0.0078	0.189	1789.367	0.127	1598.246	1820.744
Average	0.0118	0.2366	2384.1416	0.1574	2510.8936	**2759.9726** ⋆

Table 4.6: σ^2_{bc} for FlavaTextModel and Data2VecTextModel

Model	σ^2_{bc} in Score	σ^2_{bc} in Prediction Time (s)	σ^2_{bc} in Fit Time (s)
FLAVA	0.0016	268.57 ⋆	17507930.28 ⋆
Data2Vec	0.0011 ⋆	340.41	26278114.06

data domain. Tables 4.12, 4.11, 4.13 present evaluations over 6 of the 18 multimodal tabular + text datasets released by Shi et al. [10]. Additional results over the remaining datasets are provided in Appendix A.1. We run each NAS-based method for obtaining well-performing architectures for $\approx 5-6$ hours.

Our aim is to study how varying the textual signal generated using a pre-trained

model, present in the late-fused universal representation affects the overall performance of the pipeline. For that, we conduct experiments by performing the classification and regression tasks using pipeline configurations in g' over the 18 multimodal datasets. We study the predictive ability of the late-fused multimodal universal representations by exploring several tree-based ensemble models (as shown in fig 4.7) for the mapping of these embeddings to a target domain. The experiments are conducted by first experimenting on FLAVA and then on Data2Vec. We show our obtained results for 6 (salary, product, news, pop, cloth and wine) of the 18 datasets in this section in the form of Tables 4.12, 4.11, 4.13 and 4.7. Details of the downstream models are provided in Appendix A.3. Also, additional results for the remaining datasets in the AutoMM Benchmark can be found in Appendix A.1.

On analysing these results we find that pipeline configurations consisting of the FLAVA model initialised in the given experimental settings perform better on average (0.366) than pipeline configurations consisting of Data2Vec models (0.2366). Even though this is true, FLAVA pipelines were relatively inefficient in terms of fitting the data across these 18 datasets. The pipeline configurations consisting of Data2Vec took 1720.205 seconds less on average for the fitting of training data. If observe the difference Δ between the average performance score of the FLAVA and the Data2vec pipelines, we find $\Delta = 0.13$, indicating approximately similar performances. Moreover, the observed Δ in terms of inferencing (full-data) between FLAVA and Data2vec was -2144.7846 seconds, indicating the inferencing efficiency of FLAVA pipelines. Tables 4.4, 4.5 display these results for the reader's reference. Additionally, we compute a within-class variance score σ^2_{wc} for each of the datasets to understand the variability in performance when evaluating across different downstream models. μ_{wc} indicates the average performance evaluated across different downstream models for a dataset. Given this, we see from the average σ^2_{wc} of the Score for both FLAVA and Data2Vec and understand that the performance on average for every dataset, evaluated against several models, doesn't vary much. Also, from 4.6 we see that the variability (σ^2_{bc}) in the performance of pipeline configurations consisting of FLAVA and Data2Vec between the 18 AutoMM multimodal datasets does not very much. Finally, the average σ^2_{wc} for the prediction time and fitting time is quite large for both FLAVA and Data2vec. Thus, even though FLAVA on average turned out to be more efficient than Data2Vec, we remain unsure how *reliable* this efficient nature tends to be.

4.2.2 Prior Evaluation: Text + Vision Modality

Table 4.7: Average NAUC scores of models with FlavaTextModel and Data2VecTextModel

Model	Avg. Score (FLAVA)	Avg. Score (Data2Vec)
product + salary		
WE L2	0.258	0.3673
ET BAG L1	0.288	0.369
WE L3	0.256	0.367
RF BAG L1	0.290	0.350
LGBM BAG L2	0.281	0.321
LGBMXT BAG L2	0.260	0.349
RF BAG L2	0.258	0.345
ET BAG L2	0.257	0.357
Avg. (product + salary)	0.271	0.3566
channel + pop		
WE L3	0.442	0.437
LGBM BAG L2	0.440	0.437
RF BAG L2	0.439	0.436
WE L2	0.434	0.435
LGBMXT BAG L1	0.396	0.359
LGBM BAG L2	0.357	0.318
SE L2	0.423	0.446
WE L3	0.420	0.443
LGBMXT BAG L1	0.416	0.432
SE L2	0.384	0.406
LGBMXT BAG L2	0.377	0.927
RF BAG L1	0.362	0.927
XGBoost	0.358	0.928
LGBM BAG L2	0.372	0.927
Avg. (channel + pop)	0.4026	0.5075
cloth + wine		
LGBM BAG L2	0.599	0.562
ET BAG L1	0.566	0.569
RF BAG L1	0.577	0.590
WE L2	0.621	0.659
LGBM BAG L2	0.594	0.594
ET BAG L2	0.562	0.512
WE L3	0.566	0.531
Avg. (cloth + wine)	0.5816	0.5783

FLAVA						
	Score		Pred Time(s)		Fit Time(s)	
Data	σ^2_{wc}	μ_{wc}	σ_{wc^2}	μ_{wc}	σ^2_{wc}	μ_{wc}
Flickr30k	**0.042**	**0.205**	**3356939.854**	**3691.097**	**12386.962**	**7499.211**
SBU	**0.012**	**0.177**	**1785102.379**	**4447.424**	**5204.603**	**14814.139**
ALBEF						
	Score		Pred Time(s)		Fit Time(s)	
Data	σ^2_{wc}	μ_{wc}	σ_{wc^2}	μ_{wc}	σ^2_{wc}	μ_{wc}
Flickr30k	**0.052**	**0.216**	**3843.196**	**3.983**	**15372.218**	**5139.134**
SBU	**0.013**	**0.181**	**2268.557**	**4.265**	**14489.429**	**2886.672**

Table 4.8: FLAVA and ALBEF prior evaluation results for Flickr30k and SBU datasets for the *image text matching task* over the text + vision modality. Variance (σ^2_{wc}) and mean (μ_{wc}) are shown for scores, prediction times, and fit times.

To evaluate the text + vision modality, we set up the framework of our pipeline in the form of variants 2 and 3 discussed in Chapter 3. For this modality, we evaluate the pipeline consisting of the FLAVA and ALBEF pre-trained models over image text matching and visual question-answering tasks. The image text matching task involves performing evaluations over unlabelled data, particularly a set of texts and corresponding positive or negative images. The visual question-answering task can be distilled as a classification task. We set up the task mainly as a set of multiple objective questions corresponding to an image, comprising a corresponding binary answer.

ALBEF						
	Score		Pred Time(s)		Fit Time(s)	
Data	σ^2_{wc}	μ_{wc}	σ_{wc^2}	μ_{wc}	σ^2_{wc}	μ_{wc}
VQA2.0	**0.026**	**0.931**	**26702872.876**	**1535.239** ⋆	**5733629.292**	**4876.595**
FLAVA						
	Score		Pred Time(s)		Fit Time(s)	
Data	σ^2_{wc}	μ_{wc}	σ_{wc^2}	μ_{wc}	σ^2_{wc}	μ_{wc}
VQA2.0	**0.001**	**0.933** ⋆	**7637409.074**	**1927.186**	**3095158.288**	**4651.606** ⋆

Table 4.9: ALBEF model evaluation results for the VQA2.0. Variance (σ^2_{wc}) and mean (μ_{wc}) are shown for scores, prediction times, and fit times.

For the image text matching task, the evaluations are performed over the Flickr30k and SBU Image Captioning dataset. The visual question-answering task is solved for the

VQA2.0 dataset. As done for the first modality, we study the predictive nature of the generated universal latent representations, generated by the pre-trained models in the pipeline structure, by analysing the overall performance of the pipeline. The performance scores act as an empirical *proxy* for the richness of the predictive signal contained in the embeddings generated by the pre-trained models. From table 4.8, we observe that pipelines consisting of the ALBEF pre-trained model perform better than the ones consisting of FLAVA in retrieving relevant text captions, given an image input. This tells us that pipeline architectures that chose to fuse the image text input signals before applying cross-attention using a multimodal transformer *generalized* better over these two datasets, 0.261 vs. 0.205 performance score on Flickr30k and 0.181 vs. 0.177 on the SBU Image Captioning dataset. Looking at the average fit and prediction times of the pipelines consisting of FLAVA and ALBEF in Table 4.8, we find that ALBEF pipelines are drastically efficient in zero-shot inferencing as compared to the pipelines consisting of the FLAVA pre-trained model.

A detailed overview consisting of the FLAVA and ALBEF pre-trained models along with the respective downstream models and their corresponding scores, fit and performance time is shown in Tables 4.14, 4.15. Having analysed the image text matching task, we move on towards analyzing the overall results for the visual question-answering task. Table 4.9 lists the average performance of pipelines consisting of the FLAVA and ALBEF pre-trained models, when the downstream model component in the pipeline is varied using a NAS-based method. From the given table we observe that for the VQA task performed over the VQA 2.0 dataset, pipelines consisting of the FLAVA model as the pre-trained model component perform slightly better (0.933) as compared to pipeline configurations with ALBEF as a pre-trained model (0.931). Furthermore, we observe that the ALBEF pipelines on average are efficient ($\Delta \approx -392$ seconds) at inferencing full data for this dataset and task as compared to the pipelines consisting of FLAVA. On the other hand, the FLAVA pipelines turned out to be more effective ($\Delta \approx -225$ seconds) in terms of training the input data as compared to the ALBEF pipelines on average (4.9). More detailed evaluations are provided in tables 4.14 and 4.15.

4.2.3 Prior Evaluation: Tabular + Text + Vision Modality

FLAVA						
	Score		Pred Time(s)		Fit Time(s)	
Data	σ^2_{wc}	μ_{wc}	σ_{wc^2}	μ_{wc}	σ^2_{wc}	μ_{wc}
PetFinder	0.101	0.373	354361.021	**155.055** ⋆	661442.435	**1387.345** ⋆
CD-18	0.139	0.412	509609.809	215.790	272725.334	906.433
ALBEF						
	Score		Pred Time(s)		Fit Time(s)	
Data	σ^2_{wc}	μ_{wc}	σ_{wc^2}	μ_{wc}	σ^2_{wc}	μ_{wc}
PetFinder	0.108	**0.393** ⋆	156400.250	207.764	74043.333	1464.711
CD-18	0.122	**0.424** ⋆	97042.834	**178.171** ⋆	46760.811	**816.697** ⋆

Table 4.10: FLAVA and ALBEF model evaluation results for PetFinder and CD-18. Variance (σ^2_{wc}) and mean (μ_{wc}) are shown for scores, prediction times, and fit times.

To tackle this modality we refer to variant 1 of our pipeline structure g'. We set up Autogluon's multimodal for processing the tabular signals and used the pre-trained models to extract cross-modal interactions between the vision-language modalities. We already explained in detail our methodology and set-up of the experiment. We perform our set of experiments over a classification task solved for the Petfinder dataset and a regression task solved for the CD-18 mobile dataset. A detailed overview of our evaluations is provided in Table 4.17.

We now present our analysis for the Petfinder and CD-18 datasets over pipeline configurations consisting of FLAVA and ALBEF models set for solving the classification and regression tasks respectively. From Table 4.10 we observe that the pipeline consisting of the ALBEF models outperforms (0.393) the pipelines consisting of the FLAVA pre-trained model on average (0.373). Also, the ALBEF pipelines on average outperform (0.424) the FLAVA pipelines (0.412) for the CD-18 dataset. Furthermore, we see that the ALBEF pipelines over the regression task (CD-18) on average are more efficient than the FLAVA pipelines by taking ≈ 90 seconds less on average for fitting the training data. For the Petfinder dataset, pipeline configurations consisting of the FLAVA pre-trained models portray efficiency by taking ≈ 88 seconds to fit over the full data as compared to pipeline configurations consisting of the ALBEF model. ALBEF pipelines are more efficient in inferencing the training data for the CD-18 data as well as portraying a $\Delta \approx 38$ seconds. For the PetFinder dataset, the FLAVA pipelines infer full data by ≈ 52 seconds faster on average as compared to that of the pipelines consisting of ALBEF models 4.10.

Table 4.11: Pipeline evaluations comprising of FlavaTextModel, Data2VecTextModel and several downstream models, on the *channel* and *pop* data from the AutoMM Benchmark. score represents the NAUC score for the classification tasks and R^2 for the regression tasks.

	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**	**11467.700**
RFMSE_BAG_L1	**0.362**	**107.512**	**3844.022**	**0.927**	**590.493**	**18794.570**
XGBoost*	**0.358**	**1152.318**	**14604.284**	**0.928**	**315.603**	**10984.844**
LightGBM_BAG_L2	**0.372**	**1249.579**	**17855.008**	**0.927**	**3.134**	**1133.980**

Table 4.12: Pipeline evaluations comprising of FlavaTextModel, Data2VecTextModel and several downstream models, on the *product* and *salary* data from the AutoMM Benchmark. score represents the NAUC score for the classification tasks and R^2 for the regression tasks.

	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**	**20340.757**	**0.367**	**1061.557**	**18765.396**
RFMSE_BAG_L1	**0.290**	**266.354**	**920.093**	**0.350**	**330.742**	**881.680**
LightGBM_BAG_L2	**0.281**	**574.719**	**19221.342**	**0.321**	**617.396**	**22389.342**
XT_BAG_L2	**0.260**	**574.879**	**1266.841**	**0.349**	**620.545**	**1381.093**
RFMSE_BAG_L2	**0.258**	**847.312**	**17277.004**	**0.345**	**728.986**	**16452.985**
ETMSE_BAG_L2	**0.257**	**835.327**	**16345.931**	**0.357**	**745.634**	**18437.789**
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**

Table 4.13: Pipeline evaluations comprising of FlavaTextModel, Data2VecTextModel and several downstream models, on the *cloth* and *wine* data from the AutoMM Benchmark. score represents the AUC score for the classification tasks and R^2 for the regression tasks.

	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**

Table 4.14: Pipeline evaluations comprising of FlavaModel, AlbefModel and several downstream models, on the Flickr30k data. score represents the NAUC score for the classification tasks and R^2 for the regression tasks. Prediction and fit times are mentioned in seconds.

	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

Table 4.15: Pipeline evaluations comprising of FlavaModel, AlbefModel and several downstream models, on the SBU Image Captioning data. *score* represents the NAUC score for the classification tasks and R^2 for the regression tasks. Prediction and fit times are mentioned in seconds.

	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

Table 4.16: Pipeline evaluations comprising of FLAVA, ALBEF and several downstream models, on the VQA2.0 dataset. The score represents the NAUC score for the classification tasks and R^2 for the regression tasks.

	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

Table 4.17: Pipeline evaluations comprising of FLAVA, ALBEF and several downstream models, on the *Petfinder* and *CD*-18 data. score represents the NAUC score for the classification tasks and R^2 for the regression tasks.

	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

4.3 Results: SMAC Procedure

In this section, we discuss the results obtained for the sequential model-based algorithm configuration selection procedure given our experiment design and prior evaluations over the 3 specified modalities. We aim to provide an overview of the optimisation process and the performance scores for the generated incumbent configurations evaluated over the objective function.

For every modality, we discuss the behaviour of the optimisation process, robustness in terms of finding *better* configurations than the ones evaluated in the meta-dataset and comment regarding the efficiency of a Sequential Model-Based Bayesian Optimisation (SMBO) process, having warm-started its search for *optimal* configurations. For each evaluated dataset we further present the best incumbent configuration. Moreover, we also compare the results of our method for generating optimal pipeline configurations against AutoGluon [9], a pure NAS-based AutoML system to compare the efficiency in analyzing relevant pipeline configurations.

4.3.1 Results : Tabular + Text Modality

We present our SMAC optimisation results for the Product Sentiment Machine Hack dataset evaluated over the 18 AutoMM Benchmark datasets [10]. The Scenario $\mathcal{S}$ for the experiment is defined as: $\#D_{train}$ **:** 5019, **Trial Time :** 1200, **Wall Clock Time** T **:** 2750, $t_0 : 60$ (t_θ is the initialization parameter). We examine the learning curve and understand the behaviour of the SMAC algorithm under a 100% train size for the product sentiment machine hack data. The time on the x-axis of fig 4.5 is transformed using a log transformation function :

$$\tilde{t} = \frac{log(1 + \frac{t}{t_\theta})}{log(1 + \frac{T}{t_\theta})} \tag{4.1}$$

Transformation $\tilde{t}$ is performed to examine the behaviour of the SMAC algorithm at the start of the optimisation process.

Data: Product Sentiment Machine Hack:

Task: classification

Figure 4.5 offers a comprehensive view of the learning process executed by the SMAC algorithm when crafting end-to-end pipeline configurations for the product sentiment machine hack dataset. The dataset encompasses a total of $5,091$ $\#Train$ data points. As previously mentioned, the initial incumbent configuration is derived from the warm-started configuration predicted by the Expected Improvement (EI) acquisition function, acting as the launchpad for the SMAC search procedure.

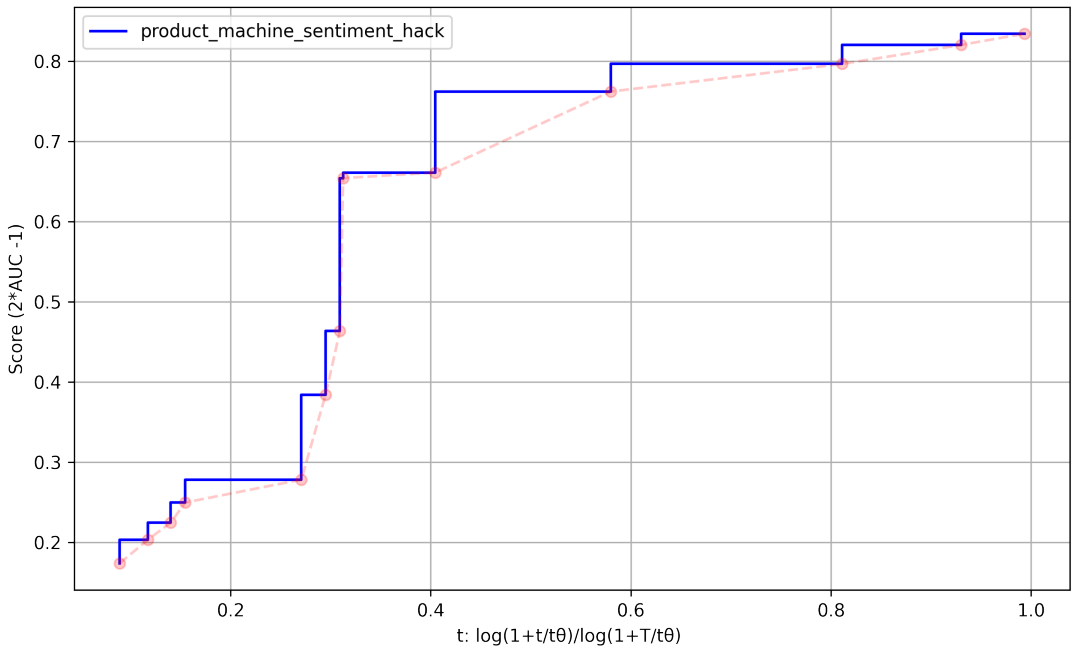


Figure 4.5: The learning curve depicting the obtained NAUC as a function of time across different pipeline configurations evaluated during the SMAC optimisation process for the product sentiment hack dataset. **The Area Under Learning Curve (ALC)** is **0.8190**

Consequently, for each evaluation, the performance score of the first configuration corresponds to the assessment conducted on θ_w as forecasted by the acquisition function, utilizing the predicted mean and uncertainty furnished by ψ_l. Upon observing Figure 4.5, it becomes evident that the assessment conducted on θ_w yields an NAUC score of 0.1034, which equates to an AUC score of 0.5517. To contextualize this result, a glance at Tables 4.5 and 4.4 reveals that data2vec pipelines for this dataset achieved an average NAUC score of approximately 0.29, while FLAVA pipelines secured an average NAUC score of approximately 0.27. Interestingly, the NAUC score obtained on θ_w closely aligns with the prior NAS-based evaluations. This suggests that SMAC commences its configuration exploration with a θ_w configuration that leads to an AUC score of 0.5517.

As SMAC diligently follows its systematic process to determine θ_{next} and intensify θ_{next} when $f_{\text{next}} > f_{\text{incumbent}}$, supported by the Intensifier $\mathcal{I}$, it converges within the prescribed 45-minute timeframe. Consequently, SMAC foresees the ultimate optimal incumbent pipeline configuration, referred to as θ_{optimal}, for the Product Sentiment Machine Hack Dataset. Impressively, the NAUC score assessed for θ_{optimal}, as predicted by SMAC, attains a remarkable value of 0.8342, corresponding to an AUC score of 0.9171. This outcome underscores the prowess of SMAC in navigating the intricate landscape of pipeline configurations. In comparison, Shi et al. outlined the outcomes of their benchmark evaluations of their NAS-based technique aimed at addressing the tabular + text modality in Figure 4.4. Their most notable achievement for this dataset

manifested as an AUC score of 0.913. This context serves to formalize the discussion surrounding the performance of our approach to that of the NAS-based methods evaluated by Shi et al.

Our proposed method demonstrates a performance that is either slightly superior or approximately equivalent to that of Shi et al. [10] on this particular dataset. While this observation holds, it is worth noting that Shi et al. detail in their work [10] that their models were trained individually for each of the 18 datasets for an extensive duration of around 200 hours. In contrast, our optimization approach, which leverages a metadata M, a meta-learner ψ_l, and an acquisition function $a_{\mathcal{M}l}$ to generate pipeline configurations comprising pre-trained vision-language models, operates in a significantly more efficient manner. It achieves a comparable level of performance while requiring only 0.375% of the time consumed by their proposed NAS-based technique for searching optimal configurations. This efficiency gain underscores the efficacy of our optimization approach and highlights its potential to expedite the model development process in multimodal tabular data analysis. We provide the details of θ_{optimal} in the form of Table 4.18. SMAC evaluated

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
Hiddem 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 4.18: Best incumbent for the product machine data

32 configurations, of which 13 trials resulted as the incumbents and 6 trials were treated as $-inf$ or CRASH. **The Area Under Learning Curve (ALC)** value obtained for the product sentiment machine hack data is **0.8190**.

Data: cloth

Task: regression

Now, we delve into the results attained for a regression task utilizing the *cloth* dataset, shifting our focus from classification to regression analysis. The primary objective of this task is to predict

women's clothing review scores based on both tabular and text features. The evaluation metric employed in this context is the *root mean squared error* (RMSE) score, where lower values indicate better model performance. The optimization scenario $\mathcal{S}$ for this task is defined as follows: $\#D_{train}$ **:** $18,788$, **Trial Time :** $1,200$, **Wall Clock Time** T **:** $2,750$, and $t_0 : 60$ (t_θ serves as the initialization parameter). We are now poised to explore the learning curve and gain insights into the SMAC algorithm's behaviour when confronted with the entire training data (*100%* train size) for the *cloth* dataset.

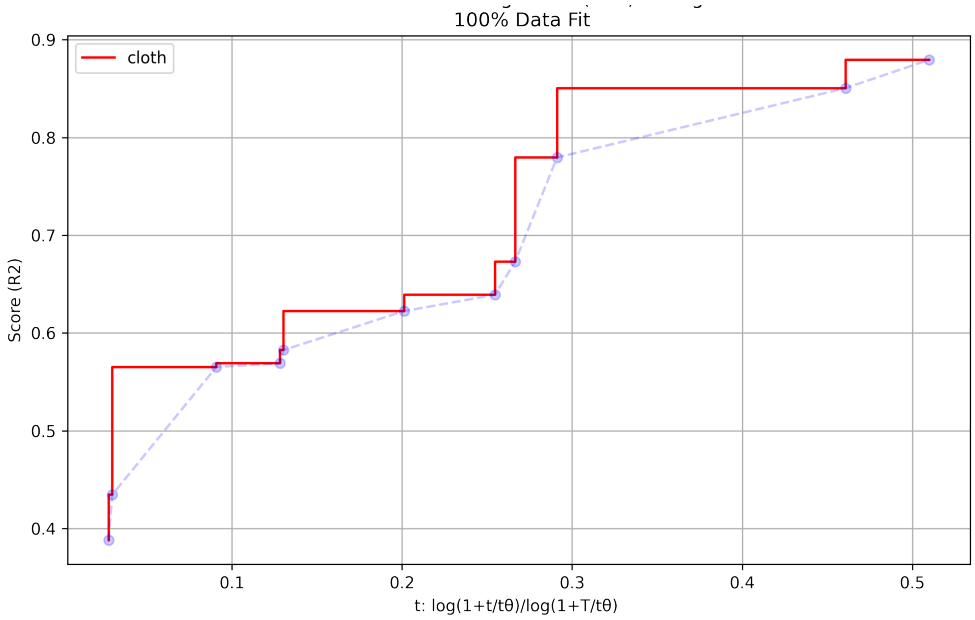


Figure 4.6: The learning curve illustrating the NAUC values obtained as a function of time during the SMAC optimization process for the *cloth* dataset. **The Area Under Learning Curve (ALC)** is **0.7405**.

Upon examining Figure 4.6, the initial observation is the SMAC optimization process's ability to converge within the stipulated maximum wall clock time of 45 minutes. This robust convergence showcases the effectiveness of the SMAC algorithm under the provided budget constraints on the *cloth* dataset. Regarding the initial configuration θ_w, the R^2 score obtained from the first trial amounted to 0.3388. Referring to the prior evaluations in Table 4.4, we note that the average R^2 score for FLAVA pipelines was 0.591, while for Data2Vec pipelines in Table 4.5, the average R^2 score was 0.570. The R^2 score achieved by the initial configuration is approximately 20% lower than the average of the NAS-based evaluations. As the search commences with the evaluation of θ_w, the algorithm proceeds to generate incumbent configurations 3, 4, and 5, showcasing marginal increments in the R^2 value. This behaviour can be attributed to an Aggressive Racing strategy, where the SMAC algorithm intensifies around prior incumbents by sampling

configurations within their local neighbourhoods. Additionally, the Expected Improvement (EI) acquisition function in SMAC balances exploration and exploitation by selecting configurations with high predictive means and uncertainties. This high uncertainty leads to the exploration of local regions. This random and local search tendency of the acquisition maximizer is evident in the graph, where incumbent trials 3, 4, and 5 are clustered nearby.

Ultimately, SMAC identifies the best incumbent configuration θ_{optimal}, yielding an R^2 value of 0.8795 on the *cloth* dataset. Referencing Table 4.19 provides a comprehensive insight into the components and corresponding hyperparameter values associated with the obtained θ_{optimal} for the *cloth* dataset. This comprehensive analysis demonstrates the effectiveness of SMAC in refining pipeline configurations for regression tasks, resulting in improved R^2 scores while adhering to given budget constraints. If we compare our results to the AutoMM benchmark depicted in

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
Hiddem 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 4.19: Best incumbent for the cloth data

figure 4.4, we see that the best value obtained for the cloth dataset by Shi et al using an expensive NAS method is 0.751. The R^2 value obtained by evaluating θ_{optimal} is 0.8795, suggesting that our predicted pipeline configuration can better explain the variability present in this multimodal dataset. Furthermore, our search for θ_{optimal} converges in 45 minutes as compared to 200 hours of training NAS-based methods for this multimodal dataset. Our method outperforms Autogluon on this tabular + text dataset in terms of both performance and efficiency.

In this experiment, SMAC submitted 35 trials for optimisation across 5 optimization loops. Of these $25, 15$ trials were evaluated, out of which 12 trials ran successfully, and the remaining 3 trials were treated as $-inf$ or CRASH. **The Area Under Learning Curve (ALC)** that we

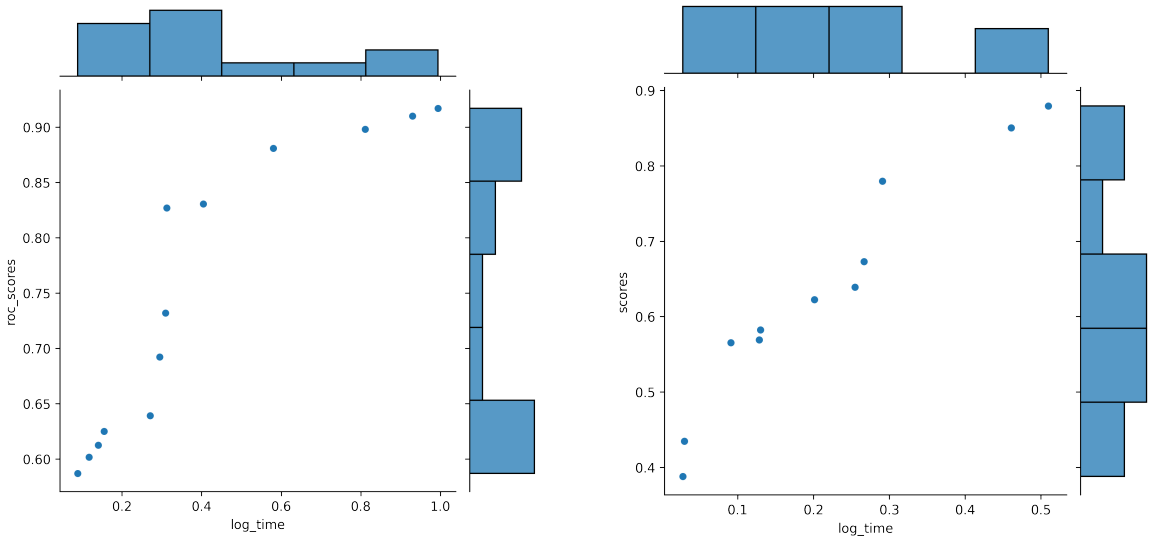


Figure 4.7: The joint probability plot for the ROC AUC scores and transformed log time $\tilde{t}$ along with the marginal probability distributions for the product sentiment hack and cloth dataset respectively.

observed for the cloth dataset for our SMBO process is **0.7405**.

Overall Analysis

We ran the SMAC algorithm for automatically generating end-to-end ML pipelines to tackle the tabular + text modality on 13 out of the 18 datasets released by Shi et al. [39], consisting of 6 classification (binary as well as multiclass) and 7 regression datasets in our experimentation. We present the obtained ALC scores over each of these 13 datasets and the NAUC scores obtained by the best incumbent pipeline configurations. Analyzing Figures 4.9 and 4.8, it is evident that the SMAC algorithm has demonstrated the capability to generate numerous "optimal" or well-performing end-to-end pipeline configurations that adapt effectively to the unique characteristics of their respective datasets. This study aimed to devise a method for achieving robust optimization within the constraints of a limited training budget. Notably, for datasets where the full training data was employed, the SMAC algorithm exhibited the capacity to identify configurations yielding high NAUC scores consistently.

The elevated ALC values of 0.8190, 0.8443, and 0.9024 for the "products," "salary," and "jc penny" datasets, respectively, provide compelling evidence that initiating the Bayesian Optimization process with warm-starting aids in achieving faster convergence. Moreover, for datasets like "ae price prediction," where the training size samples were deliberately restricted (approximately 44.94% for the "ae price prediction" dataset), it is observed that the SMAC algorithm generates pipeline configurations with strong performance; however, the frequency of occurrence for such configurations is comparatively lower. The relatively modest ALC score of

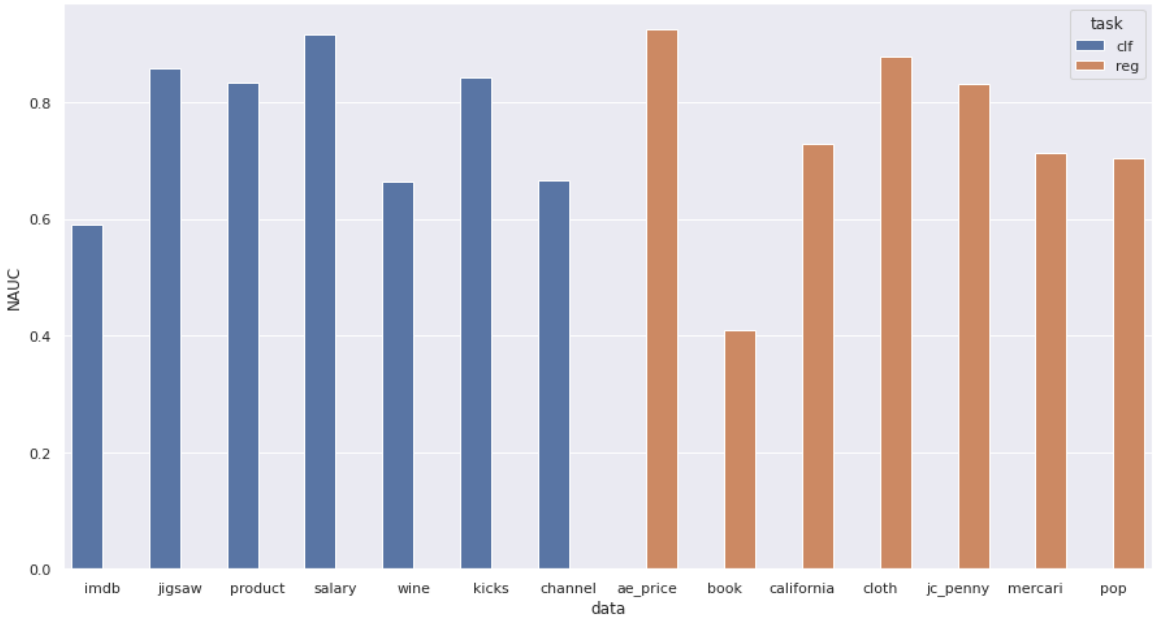


Figure 4.8: Bar Plot representing the obtained NAUC values by the best selected incumbent pipeline configurations generated by the SMAC algorithm

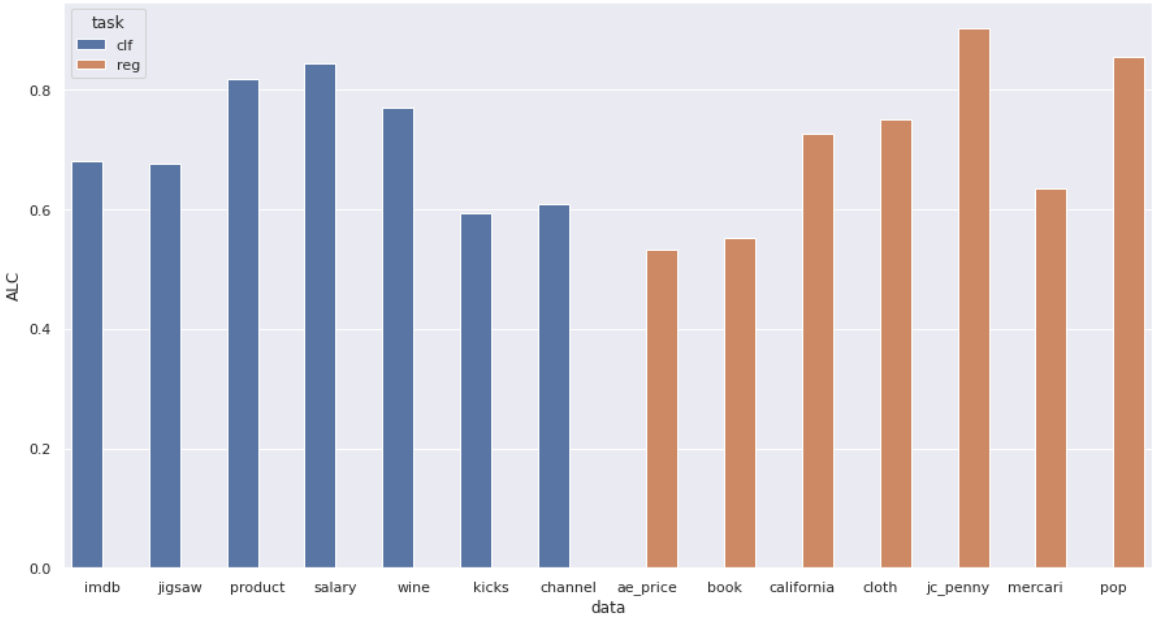


Figure 4.9: Bar Plot representing the ALC values, representing the learning capability of the SMAC algorithm under the budget T for the 13 datasets mentioned above

0.5342 for this scenario captures the characteristic behaviour exhibited by the AutoML system in this context.

We show the **best** performance (NAUC scores) of the NAS-based pipeline architectures proposed by Shi et al. in their work [10] in the form of figure 4.10. Figures 4.8 and 4.10 provide a

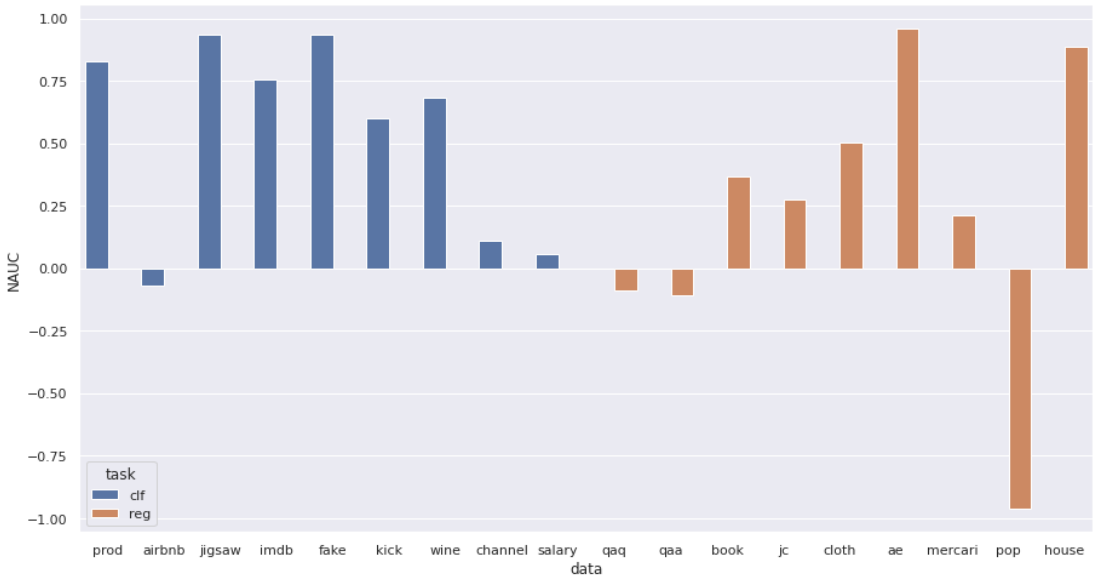


Figure 4.10: Benchmark NAUC values obtained by Shi et al. [10] for the 18 datasets released in the AutoMM Benchmark.

clear visual representation of our method's performance when compared to the costly NAS-based methods examined by Shi et al. across a set of 18 diverse multimodal datasets. Notably, our approach consistently achieves performance scores that are either on par with or slightly *better* than those attained by the aforementioned resource-intensive methods. By obtaining *optimal* NAUC scores within a remarkably short timeframe of 45 minutes for each of the 18 datasets, we effectively showcase the efficiency gained through the incorporation of a warm-starting mechanism in the SMAC search procedure. This mechanism involves the pre-evaluation of pipeline configurations that incorporate a range of pre-trained multimodal models. The outcome is a rapid convergence to promising solutions, underscoring the potency of our approach in optimizing multimodal pipelines with a markedly reduced computational burden.

These findings underscore our approach's efficacy in delivering competitive results while significantly expediting the optimization process. The ability to efficiently identify effective configurations for a diverse array of datasets underscores the practical viability and scalability of our methodology, offering a valuable contribution to the realm of multimodal optimization.

4.3.2 Results : Text + Vision Modality

Given our analysis for the tabular + text modality, we now move on to discuss the results for the text + vision modality over the Visual Question Answering (VQA) and Image Text Matching (ITM) Tasks. Having discussed the experiment design and the methodology we now discuss the results of our SMAC optimization process over the VQA2.0, Flickr30k and SBU Image Captioning Dataset.

Data: VQA2.0

Task: Visual Question Answering

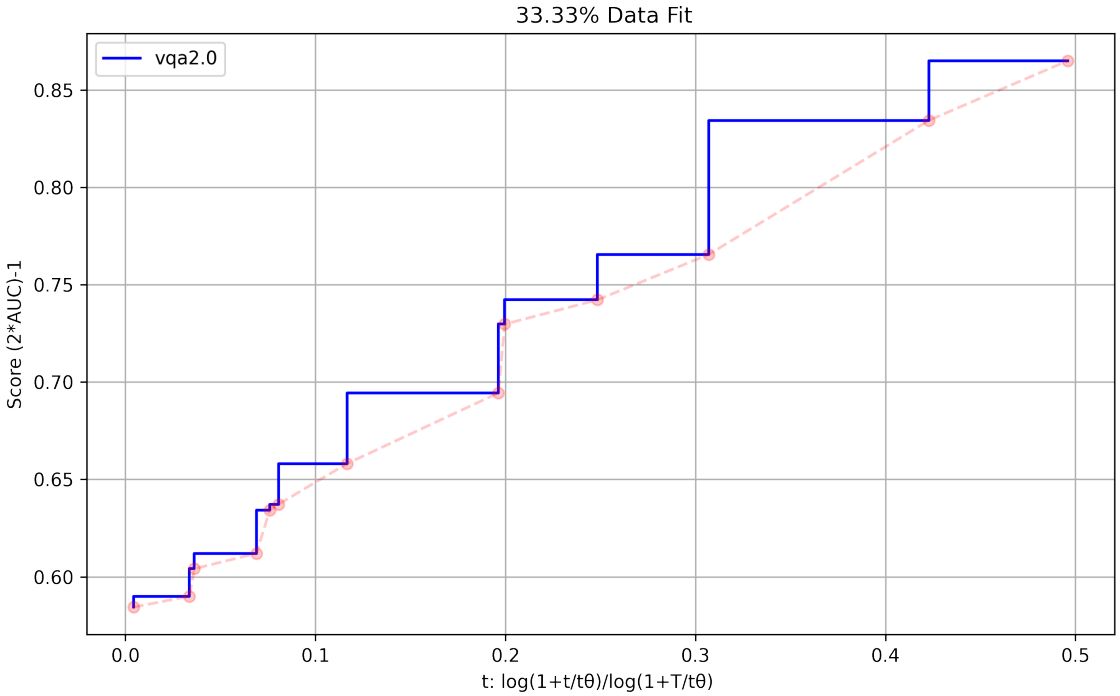


Figure 4.11: The learning curve depicting the obtained NAUC as a function of time across different pipeline configurations evaluated during the SMAC optimisation process for the VQA2.0 dataset. **The Area Under Learning Curve (ALC)** is **0.8112**

We setup the optimisation scenario for the VQA2.0 dataset as follows:
Data : VQA2.0, **No of incumbents** : 14, **No of trials submitted** : 25, **train size** : 33.33%, **train samples** : 10,000, **trial time**: 1200, **wall clock time** : 2750. Upon observing Figure 4.11, we can deduce that the NAUC score achieved through evaluation over θ_w amounts to 0.5844. When we direct our attention to the performance outcomes derived from prior evaluations on VQA datasets, specifically focusing on pipeline configurations employing FLAVA and ALBEF models, a pattern emerges. The average performance score for FLAVA pipelines hovers at 0.933, while the average NAUC for ALBEF pipelines stands at 0.931 (as indicated in Table 4.9). Notably, the NAUC associated with θ_w registers a dip of roughly 30% compared to the average NAUC from

these earlier assessments.

However, a more nuanced interpretation reveals that the process of finding the optimal pipeline commences with an initial configuration yielding a score of 0.5844. As the SMAC algorithm takes effect, it refines this configuration, resulting in a set of optimal pipeline components and hyperparameter settings denoted as θ_{optimal}. Remarkably, the NAUC score attributed to θ_{optimal} reaches 86.5%. This value is notably close to the NAUC score derived from evaluating models through a NAS-based method, a process that entails approximately 5 to 6 hours of computation. Although there is no benchmark equivalent to AutoMM for this specific modality, we undertake a comparison of our method against a pure-NAS approach implemented by Autogluon, under the same temporal and data constraints. This comparison involves the VQA 2.0 dataset, and AutoGluon constructs a multimodal-fusion MLP incorporating a staggering 119 million parameters. The resultant AUC score is measured at 0.7685 when subject to a data allocation of 33.33% and an optimization time frame of 45 minutes 4.16.

This head-to-head evaluation demonstrates the efficiency of pipeline architectures founded on pre-trained models in addressing vision-language inputs. Table 4.20 presents the pipeline components and corresponding hyperparameter configurations about θ_{optimal}. Moreover, our approach's efficiency finds further reinforcement in the capacity of meta-learning strategies and warm-starting the SMAC search procedure. These facets collectively facilitate faster convergence, validating the effectiveness of our methodology across the spectrum of multimodal optimization. During this optimisation process, the intensifier submitted 27 trials for optimisation, of which 14 trials were deemed as valid incumbents.

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
Hiddem 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 4.20: Best incumbent for the VQA2.0 data

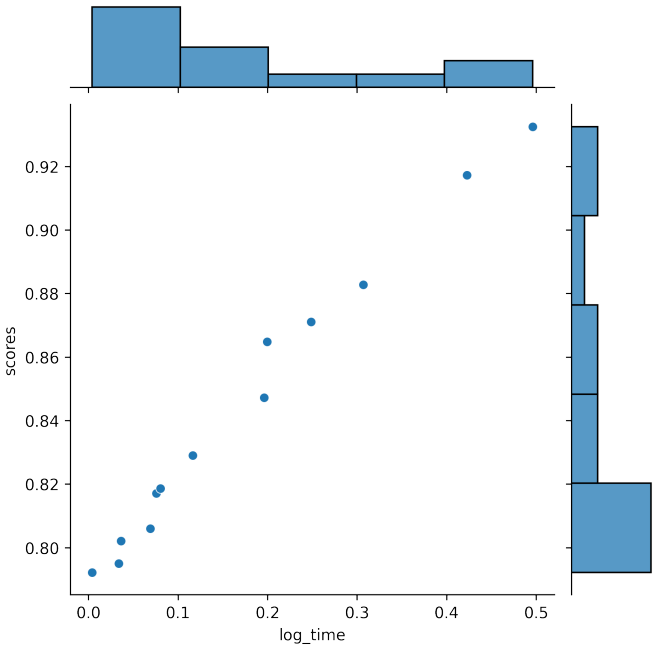


Figure 4.12: The joint probability plot for the ROC AUC scores and transformed log time $\tilde{t}$ along with the marginal probability distributions for the VQA2.0 dataset respectively.

Data: Flickr30k and SBU

Task: ITM For the Image Text Matching task, we set up the optimisation Scenario S as follows:

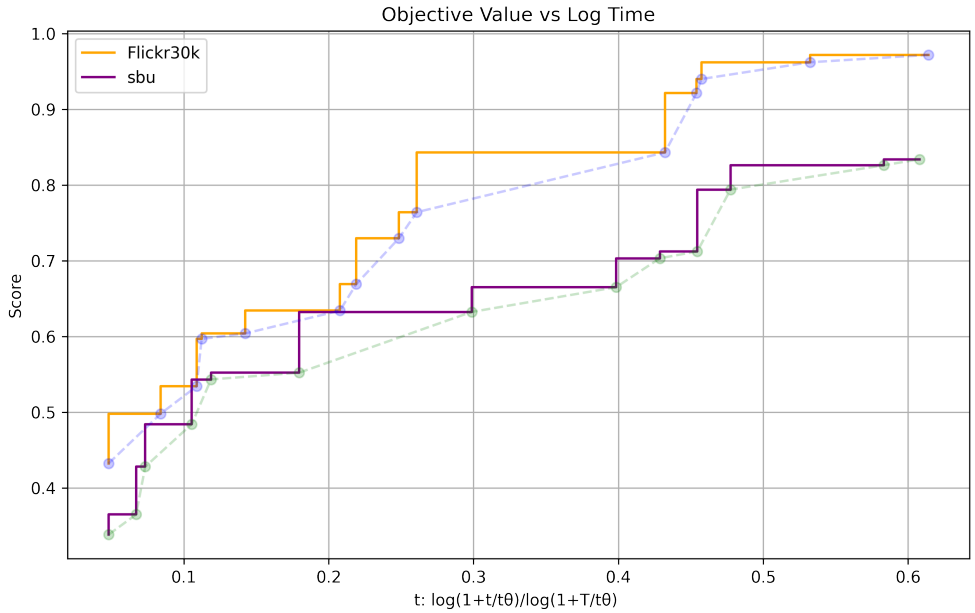


Figure 4.13: The learning curve depicting the obtained NAUC as a function of time across different pipeline configurations evaluated during the SMAC optimisation process on the Flickr30k and SBU dataset for the ITM task. **The Area Under Learning Curve (ALC)** values for the Flickr30k and SBU Image Captioning dataset are **0.8243** & **0.6852** respectively.

Data : Flickr30k, **No of successful trials** : 14, **No of trials submitted** : 20, **train size** : 33.33%, **train samples**: 10,000, **trial time**: 1200, **wall clock time**: 2750.
Data : SBU Image Captioning, **No of successful trials** : 13, **No of trials submitted** : 17, **train size** : 33.33%, **train samples** : 30,000, **trial time**: 1200, **wall clock time** : 2750.

Examining Figure 4.13, the initial configuration θ_w, as predicted by the EI acquisition function, yields a NAUC score of 0.388 for the SBU dataset and 0.424 for the Flickr30k dataset. Upon referencing the prior evaluations conducted on both datasets in Table 4.8, it is evident that the average NAUC score for FLAVA pipelines was 0.205, while for ALBEF pipelines, it was 0.206 for full data. To assess the efficiency of our approach, we conducted an additional experiment for 33% of the data with a 45-minute optimization budget. In this experiment, we ran a full-NAS-based method using AutoGluon. Under these constraints, the multimodal fusion MLP constructed by AutoGluon achieved NAUC scores of 0.340 and 0.389 for the Flickr30k and SBU Image Captioning Dataset, respectively. This score was approximately 63% lower than the score obtained for θ_{optimal}

using our method for the Flickr30k dataset. Specifically, θ_{optimal} resulted in a NAUC score of 0.9442 for the Flickr30k dataset and 0.6682 for the SBU Image Captioning Dataset under the given constraints.

Moreover, Figure 4.14 provides a visual representation of the obtained NAUC values by the best-selected incumbent pipeline configurations generated by the SMAC algorithm. Similarly, Figure 4.15 showcases the ALC values, highlighting the learning capacity of the SMAC algorithm under the budget T for the three datasets mentioned above. The juxtaposition of these two figures, 4.14 and 4.15, reveals the generalization capabilities of the AutoML system across the three datasets. The elevated ALC scores indicate that SMAC consistently generates incumbents that frequently perform well throughout the optimization run. This behaviour holds for all three text + vision datasets. These results underscore the advantages of having unified multimodal representations of the data and utilizing a warm-starting strategy, which contributes to a more rapid convergence.

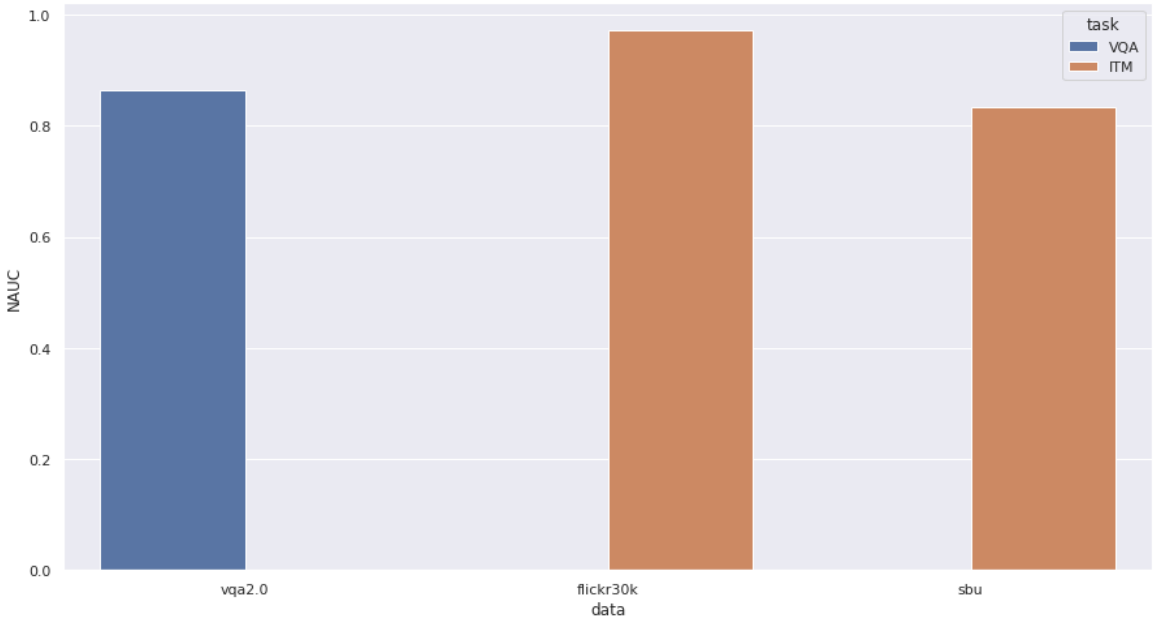


Figure 4.14: NAUC scores obtained using our AutoML method for datasets about the text + vision modality

In conclusion, our proposed AutoML method demonstrates its efficacy across various text + vision datasets, enabling efficient convergence and consistently generating well-performing incumbent pipeline configurations. Tables 4.21 and 4.22 list the configurations θ_{optimal} for the ITM task over the Flickr30k and SBU datasets respectively.

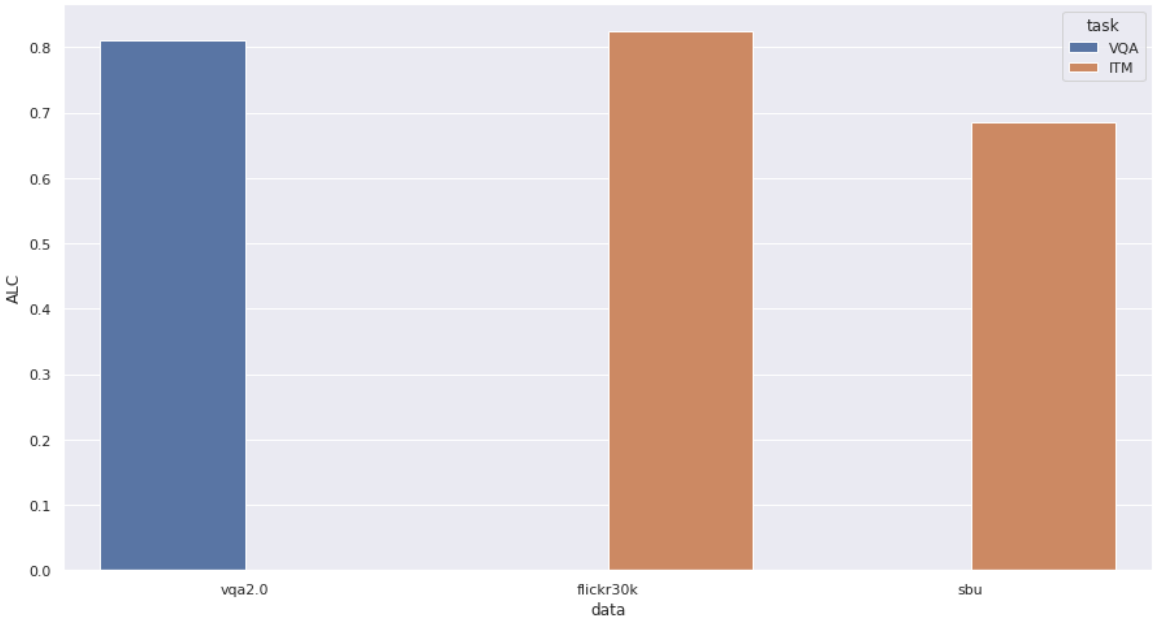


Figure 4.15: ALC scores of our SMAC optimization process for datasets about the text + vision modality

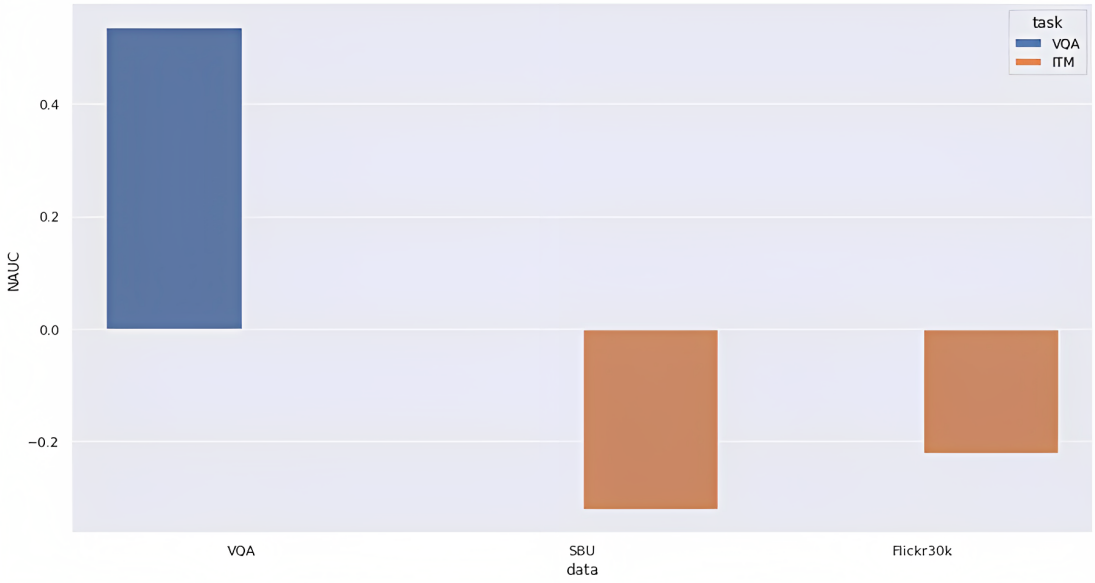


Figure 4.16: Obtained NAUC score by AutoGluon under the contained data and trial time setting for the VQA, SBU and Flickr30k datasets over VQA and ITM tasks respectively.

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
Hiddem 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 4.21: 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	$1.2268910169581294e-06$
Attention Dropout	0.32449508667965
Hiddem 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 4.22: Best incumbent for the SBU Image Captioning data

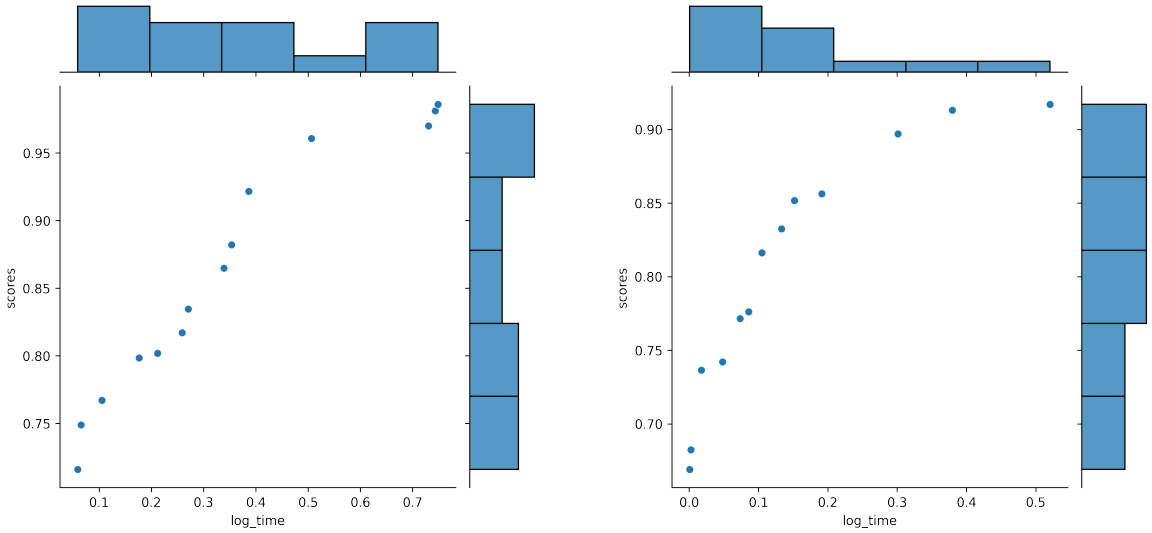


Figure 4.17: The joint probability plot for the ROC AUC scores and transformed log time $\tilde{t}$ along with the marginal probability distributions for the Flickr30k hack and SBU Image Captioning dataset respectively.

4.3.3 Results Tabular + Text + Vision Modality

Data: Petfinder and CD-18 datasets

Task: classification and regression We setup the Scenario $\mathcal{S}$ for the optimisation processor the PetFinder and CD-18 datasets as follows:

textbfData : PetFinder, **No of successful trials** : 12, **No of trials submitted** : 16, **train size** : 100%, **train samples** : 15,000, **trial time**: 1200, **wall clock time** : 2750.

Data : Cd-18, **No of successful trials** : 11, **No of trials submitted** : 18, **train size** : 100%, **train samples** : 14,993, **trial time**: 1200, **wall clock time** : 2750. From fig 4.18 we understand the behaviour of our proposed AutoML system on the PetFinder and CD-18 datasets for the classification and regression tasks respectively. The details of the Scenario defined for the SMAC optimization are as follows :

Figure 4.18 provides insights into the behaviour of the SMAC algorithm with the initial configuration θ_w, resulting in NAUC scores of 0.187 and 0.235 for the CD-18 and Petfinder datasets, respectively. Comparing these scores with the average performance scores (NAUC) obtained from FLAVA pipelines on the PetFinder and CD-18 datasets, which were 0.373 and 0.412 respectively, we observe that θ_w achieves NAUC scores similar to the NAS-based average scores. Additionally, for ALBEF pipelines, the average NAUC scores for the PetFinder and CD-18 datasets were 0.393 and 0.424 respectively.

Starting with θ_w as the initial configuration, the SMAC optimization process converges

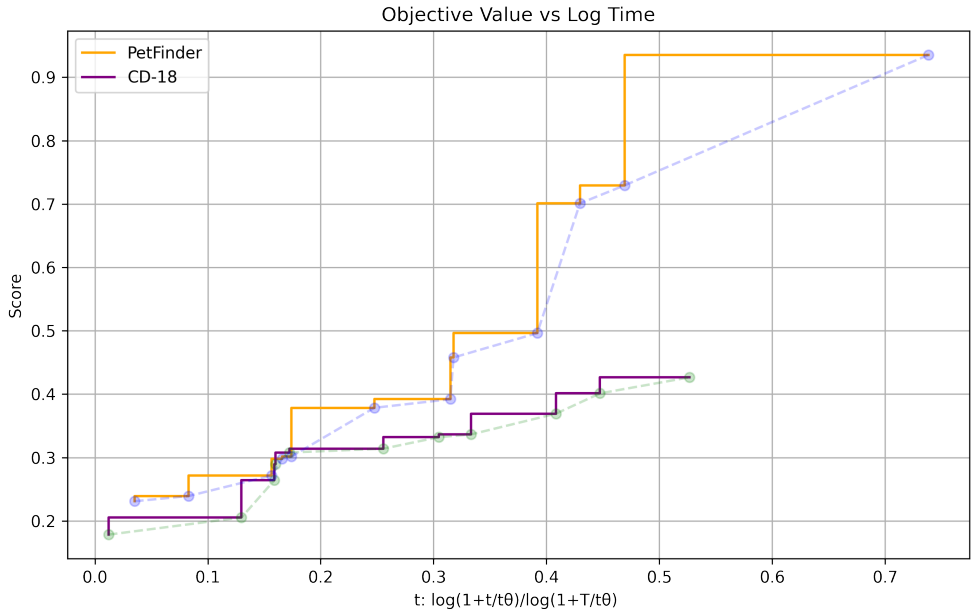


Figure 4.18: The learning curve depicting the obtained NAUC as a function of time across different pipeline configurations evaluated during the SMAC optimisation process on the PetFinder and CD-18 dataset for the classification and regression tasks respectively. **The Area Under Learning Curve (ALC)** values for the Flickr30k and SBU Image Captioning dataset are **0.7647** & **0.5047** respectively.

within 45 minutes for both the PetFinder and CD-18 datasets. The NAUC scores evaluated for the corresponding θ_{optimal} of the classification task performed on the PetFinder dataset is 0.9264, while for the regression task on the CD-18 dataset, it is 0.4127. The hyperparameters for the θ_{optimal} configurations are listed in Tables 4.23 and 4.24 for the PetFinder and CD-18 datasets, respectively. Furthermore, we compare our results with those obtained from Autogluon by running a NAS-based search for 45 minutes for the classification and regression tasks on the respective datasets. This NAS-based method using AutoGluon yields a NAUC score of 0.6844 for the Petfinder dataset and an NAUC score of 0.3603 for the CD-18 dataset, as demonstrated in Figure 4.20.

In conclusion, our proposed AutoML method showcases its effectiveness in achieving competitive performance across both classification and regression tasks for various datasets, as highlighted by the NAUC scores obtained from the θ_{optimal} configurations. These scores **outperform or match the results** from a NAS-based approach using AutoGluon **effectively** given some time constraints.

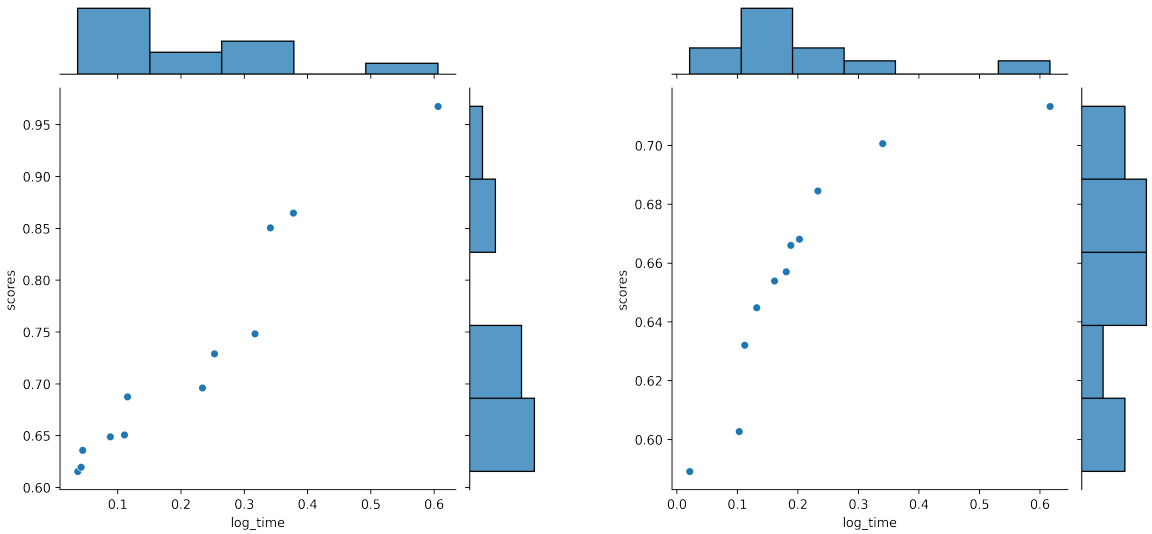


Figure 4.19: The joint probability plot for the ROC AUC scores and transformed log time $\tilde{t}$ along with the marginal probability distributions for the PetFinder hack and CD-18 dataset respectively.

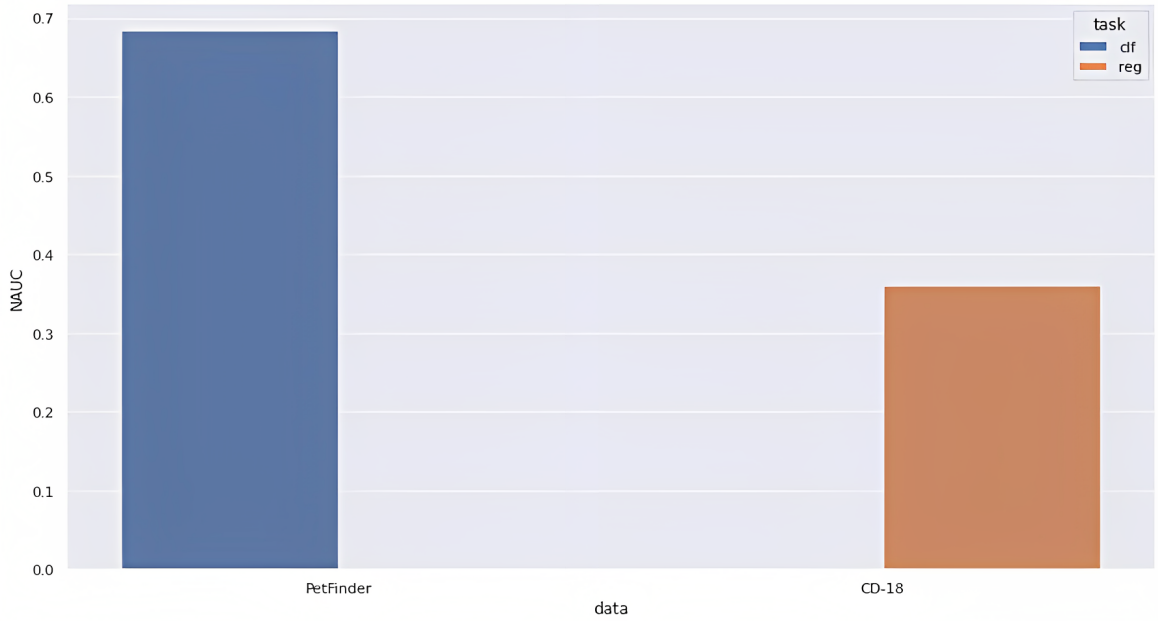


Figure 4.20: NAUC scores of the multimodal-fusion MLP constructed by AutoGluon to tackle the classification and regression tasks over the Petfinder and CD-18 multimodal datasets.

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 4.23: Best incumbent for the PetFinder 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

Table 4.24: Best incumbent for the CD-18 data

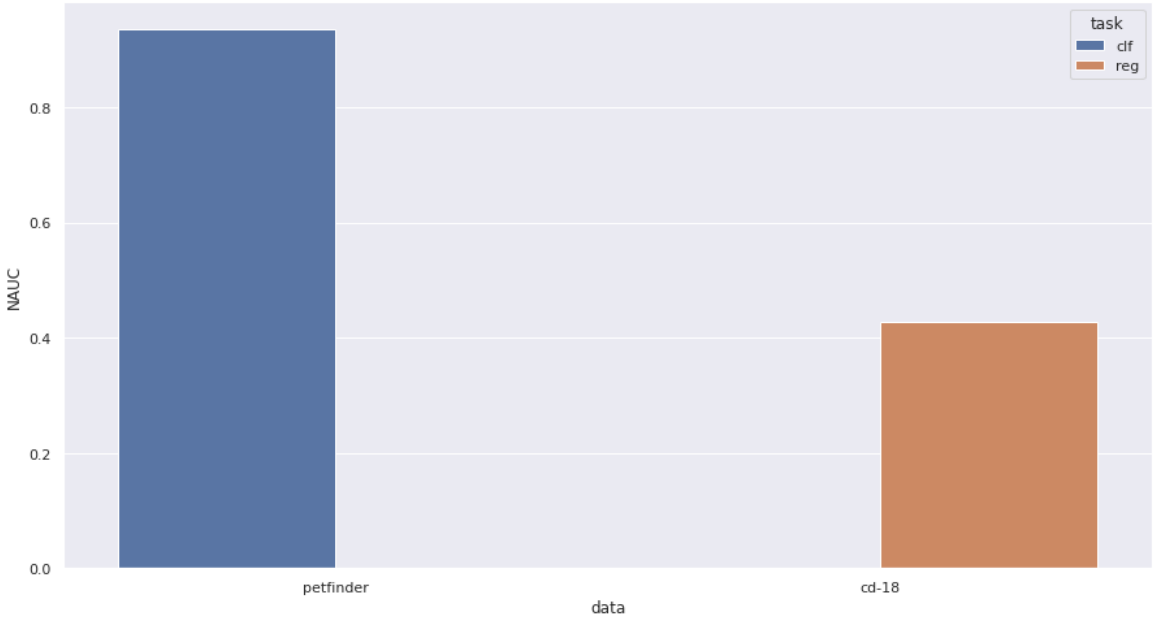


Figure 4.21: Bar Plot representing the obtained NAUC values by the best selected incumbent pipeline configurations generated by the SMAC algorithm

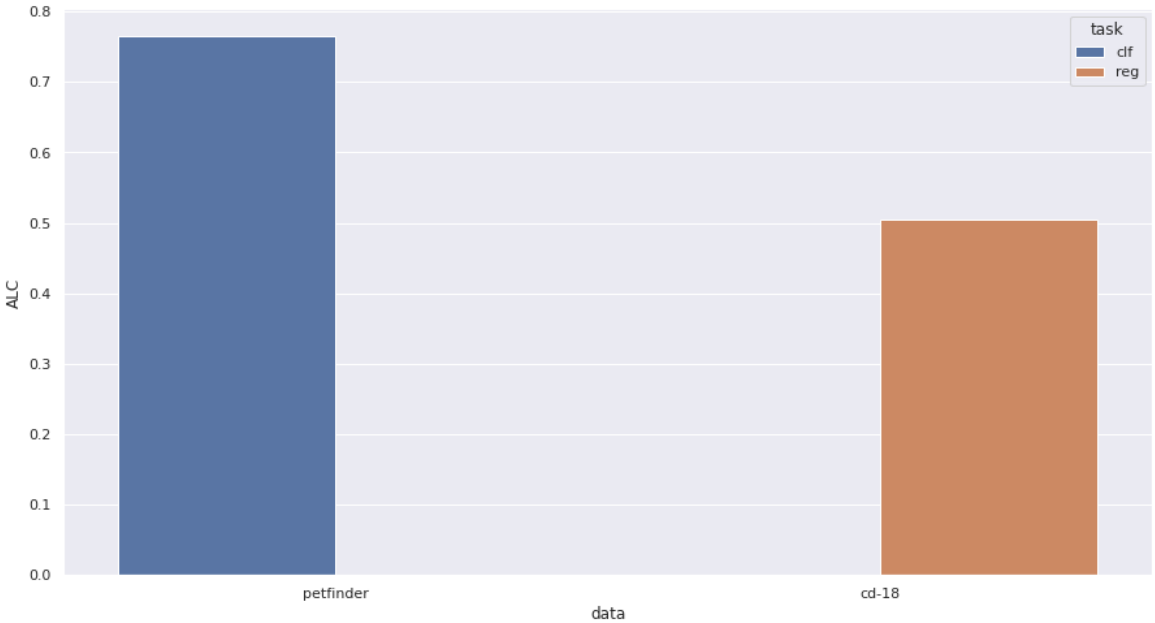


Figure 4.22: Bar Plot representing the ALC values, representing the learning capability of the SMAC algorithm under the budget T for the 2 datasets mentioned above

Chapter 5

Conclusions

The central objective of this research has been to tackle the intricate challenges of automating the process of machine learning model selection, hyperparameter tuning, and pipeline configuration for multimodal data analysis. Specifically, we have aimed to develop a sophisticated AutoML framework that is tailored to handle inputs combining visual, textual and tabular information. This overarching goal is driven by the increasing complexity and diversity of modern datasets, where the fusion of different data modalities demands advanced optimization techniques to extract meaningful insights.

Our proposed approach is grounded in a multidimensional framework that synergizes the strengths of pre-trained multimodal models, meta-learning strategies, and predictive optimization. This methodological synthesis seeks to streamline the convoluted process of configuring intricate pipelines that integrate diverse models and their associated hyperparameters. By orchestrating an ensemble of techniques, including predictive modelling, acquisition functions, and intelligent initialization, our method endeavours to achieve both efficient exploration of the configuration space and the effective exploitation of promising solutions.

1. **Experiments and Findings**: In this study, we embarked on a comprehensive exploration of the efficiency and effectiveness of our proposed AutoML approach for multimodal data, specifically targeting vision-language as well as Tabular inputs. Through rigorous experimentation and analysis, we have illuminated several key insights and achievements.

2. **Convergence Robustness**: Our approach demonstrates remarkable convergence

robustness within prescribed computational budgets. Across diverse datasets and tasks, such as classification and regression, visual question answering, and image-text matching, our AutoML framework consistently achieved convergence within a limited period. This capability underscores the adaptability and efficiency of our method in navigating the multimodal configuration space to identify optimal pipeline configurations.

3. **Accelerated Convergence with Warm-Starting**: By warm-starting the optimization process with an initial configuration predicted by the acquisition function, using a meta-dataset based on prior evaluations, our method showcased accelerated convergence. This was evident across a range of datasets and tasks, where the initial configurations consistently exhibited performance scores on par with or closely resembling the average NAS-based prior evaluations. This validates the efficacy of our approach in efficiently guiding the search towards promising areas of the configuration space.

4. **Competitive Performance with Incumbent Configurations**: The obtained incumbent pipeline configurations, denoted as θ_{optimal}, consistently yielded competitive performance scores. These scores quantified through metrics like NAUC and R^2, surpassed or closely matched results from benchmark NAS-based methods, even when considering the stringent time constraints imposed on the optimization process. This demonstration of comparable performance in significantly reduced time intervals reaffirms the utility and efficiency of our proposed AutoML method.

5. **Efficiency in Real-World Scenarios**: Moreover, our method's success is particularly pronounced in cases where full-NAS approaches would be computationally infeasible. This is evident in scenarios where our method's θ_{optimal} configurations outperformed NAS-based approaches even when evaluated within the same constraints. This showcases the robustness of our method in effectively utilizing both available data and computational resources.

In conclusion, this study introduces an innovative AutoML approach tailored to multimodal data, with a focus on pre-trained Transformer models. Through a combination of these multimodal pre-trained models, meta-learning, and a sophisticated optimization process, our approach showcases accelerated convergence, competitive performance, enhanced efficiency, and the possibility of synthesizing end-to-end ML pipeline configurations for a given task and multimodal data. These achievements not only contribute to the realm of AutoML

but also emphasize the potency of multimodal pipelines in efficiently handling complex data and tasks. Overall, our findings mark a significant step towards making sophisticated multimodal data analysis accessible and effective in various domains.

By aligning our proposed methodology with these empirical insights, we have not only realized the objectives of this research but have also paved the way for a more efficient, effective, and accessible future of multimodal data analysis through advanced AutoML techniques.

Bibliography

[1] Aishwarya Agrawal, Jiasen Lu, Stanislaw Antol, Margaret Mitchell, C. Lawrence Zitnick, Dhruv Batra, and Devi Parikh. Vqa: Visual question answering, 2015. 56

[2] Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. data2vec: A general framework for self-supervised learning in speech, vision and language, 2022. 2, 10, 11, 13, 17, 18, 30, 38, 39

[3] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers, 2021.

[4] Hangbo Bao, Wenhui Wang, Li Dong, Qiang Liu, Owais Khan Mohammed, Kriti Aggarwal, Subhojit Som, and Furu Wei. Vlmo: Unified vision-language pre-training with mixture-of-modality-experts, 2021.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

[7] Yifan Du, Zikang Liu, Junyi Li, and Wayne Xin Zhao. A survey of vision-language pre-trained models, 2022.

[8] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. 2018.

[9] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. Autogluon-tabular: Robust and accurate automl for structured data, 2020.

[10] Nick Erickson, Xingjian Shi, James Sharpnack, and Alexander Smola. Multimodal automl for image, text and tabular data. In *Proceedings of the 28th ACM SIGKDD*

Conference on Knowledge Discovery and Data Mining, KDD '22, page 4786–4787, New York, NY, USA, 2022. Association for Computing Machinery.

[11] Francis Ferraro, Nasrin Mostafazadeh, Ting-Hao (Kenneth) Huang, Lucy Vanderwende, Jacob Devlin, Michel Galley, and Margaret Mitchell. A survey of current datasets for vision and language research. In Lluís Màrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 207–213. The Association for Computational Linguistics, 2015.

[12] Matthias Feurer. Scalable meta-learning for bayesian optimization using ranking-weighted gaussian process ensembles. 2018.

[13] Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. Auto-sklearn 2.0: Hands-free automl via meta-learning. 2020.

[14] Matthias Feurer and Frank Hutter. Chapter 1 hyperparameter optimization. 2018.

[15] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey, 2020.

[16] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated Machine Learning: Methods, Systems, Challenges*. Springer Publishing Company, Incorporated, 1st edition, 2019.

[17] Salman H. Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *CoRR*, abs/2101.01169, 2021.

[18] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations, 2019.

[19] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998.

[20] Junnan Li, Ramprasaath R. Selvaraju, Akhilesh Deepak Gotmare, Shafiq Joty, Caiming Xiong, and Steven Hoi. Align before fuse: Vision and language representation learning with momentum distillation, 2021.

[21] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. Visualbert: A simple and performant baseline for vision and language, 2019.

[22] Wei Li, Can Gao, Guocheng Niu, Xinyan Xiao, Hao Liu, Jiachen Liu, Hua Wu, and Haifeng Wang. Unimo: Towards unified-modal understanding and generation via cross-modal contrastive learning, 2020.

[23] Wei Li, Can Gao, Guocheng Niu, Xinyan Xiao, Hao Liu, Jiachen Liu, Hua Wu, and Haifeng Wang. Unimo-2: End-to-end unified vision-language grounded learning, 2022.

[24] Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, Yejin Choi, and Jianfeng Gao. Oscar: Object-semantics aligned pre-training for vision-language tasks, 2020.

[25] Paul Pu Liang, Yiwei Lyu, Xiang Fan, Zetian Wu, Yun Cheng, Jason Wu, Leslie Chen, Peter Wu, Michelle A. Lee, Yuke Zhu, Ruslan Salakhutdinov, and Louis-Philippe Morency. Multibench: Multiscale benchmarks for multimodal representation learning. In Joaquin Vanschoren and Sai-Kit Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021.

[26] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, volume 8693 of *Lecture Notes in Computer Science*, pages 740–755. Springer, 2014.

[27] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search, 2018.

[28] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[29] Zhengying Liu, Adrien Pavao, Zhen Xu, Sergio Escalera, Fabio Ferreira, Isabelle Guyon, Sirui Hong, Frank Hutter, Rongrong Ji, Júlio C. S. Jacques Júnior, Ge Li, Marius Lindauer, Zhipeng Luo, Meysam Madadi, Thomas Nierhoff, Kangning Niu, Chunguang Pan, Danny Stoll, Sébastien Treguer, Jin Wang, Peng Wang, Chenglin Wu, Youcheng Xiong, Arber Zela, and Yang Zhang. Winning solutions and post-challenge analyses of the chalearn autodl challenge 2019. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(9):3108–3125, 2021.

[30] Felix Mohr, Marcel Wever, and Eyke Hüllermeier. ML-Plan: Automated machine learning via hierarchical planning. *Machine Learning*, 107(8):1495–1515, September 2018.

[31] P. Nguyen, Melanie Hilario, and Alexandros Kalousis. Using meta-mining to support data mining workflow planning and optimization. *J. Artif. Intell. Res.*, 51:605–644, 2014.

[32] Vicente Ordonez, Girish Kulkarni, and Tamara L. Berg. Im2text: Describing images

using 1 million captioned photographs. In John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger, editors, *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*, pages 1143–1151, 2011.

[33] Ekrem Öztürk, Fabio Ferreira, Hadi Jomaa, Lars Schmidt-Thieme, Josif Grabocka, and Frank Hutter. Zero-shot AutoML with pretrained models. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 17138–17155. PMLR, 17–23 Jul 2022.

[34] Letitia Parcalabescu, Michele Cafagna, Lilitta Muradjan, Anette Frank, Iacer Calixto, and Albert Gatt. Valse: A task-independent benchmark for vision and language models centered on linguistic phenomena, 2021.

[35] Bryan A. Plummer, Liwei Wang, Chris M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models, 2015.

[36] Juan-Manuel Pérez-Rúa, Valentin Vielzeuf, Stéphane Pateux, Moez Baccouche, and Frédéric Jurie. Mfas: Multimodal fusion architecture search, 2019.

[37] XiPeng Qiu, TianXiang Sun, YiGe Xu, YunFan Shao, Ning Dai, and XuanJing Huang. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10):1872–1897, sep 2020.

[38] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[39] Xingjian Shi, Jonas Mueller, Nick Erickson, Mu Li, and Alexander J. Smola. Benchmarking multimodal automl for tabular data with text fields, 2021.

[40] Amanpreet Singh, Ronghang Hu, Vedanuj Goswami, Guillaume Couairon, Wojciech Galuba, Marcus Rohrbach, and Douwe Kiela. Flava: A foundational language and vision alignment model, 2021.

[41] Krishna Somandepalli, Naveen Kumar, Ruchir Travadi, and Shrikanth Narayanan. Multimodal representation learning using deep multiset canonical correlation, 2019.

[42] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. Vl-bert: Pre-training of generic visual-linguistic representations, 2019.

[43] Hao Tan and Mohit Bansal. Lxmert: Learning cross-modality encoder representations from transformers, 2019.

[44] Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms, 2012.

[45] Markus Johannes Van Ackeren, Francesca M Barbero, Stefania Mattioni, Roberto Bottini, and Olivier Collignon. Neuronal populations in the occipital cortex of the blind synchronize to the temporal dynamics of speech. *eLife*, 7:e31640, jan 2018.

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[47] Zhen Xu, David R. So, and Andrew M. Dai. Mufasa: Multimodal fusion architecture search for electronic health records, 2021.

[48] Aidin Zehtab-Salmasi, Ali-Reza Feizi-Derakhshi, Narjes Nikzad-Khasmakhi, Meysam Asgari-Chenaghlu, and Saeideh Nabipour. Multimodal price prediction. *Annals of Data Science*, April 2021.

[49] Marc-André Zöller and Marco F. Huber. Benchmark and survey of automated machine learning frameworks. 2019.

Appendix A

Appendix

A.1 Prior Evaluations

In this chapter, we shall list the Baseline Evaluations obtained on the remaining datasets of the AutoMM Benchmark. Apart from that, after each baseline evaluation result, we shall publish the visualisation of the learning curve during the process of synthesizing multimodal end-to-end pipelines for the multimodal tabular + text datasets. We used these prior evaluations $P \bigcup P^*$ to construct our meta-dataset M

Table A.1: Pipeline evaluations comprising of FlavaTextModel, Data2VecTextModel and several downstream models, on the fake job and imdb data. *score* represents the NAUC score for the classification tasks and R^2 for the regression tasks. Prediction and fit times are mentioned in seconds.

	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

Table A.2: Pipeline evaluations comprising of FlavaTextModel, Data2VecTextModel and several downstream models, on the jigsaw and kick data. *score* represents the NAUC score for the classification tasks and R^2 for the regression tasks. Prediction and fit times are mentioned in seconds.

	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

Table A.3: Pipeline evaluations comprising of FlavaTextModel, Data2VecTextModel and several downstream models, on the *ae* and *book* data from the AutoMM Benchmark. score represents the NAUC score for the classification tasks and R^2 for the regression tasks.

	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

Table A.4: Pipeline evaluations comprising of FlavaTextModel, Data2VecTextModel and several downstream models, on the *california* and *jcpenny* data from the AutoMM Benchmark. score represents the NAUC score for the classification tasks and R^2 for the regression tasks.

	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

A.2 Additional SMAC Results

We will present results on the remaining datasets from the AutoMM Benchmark.

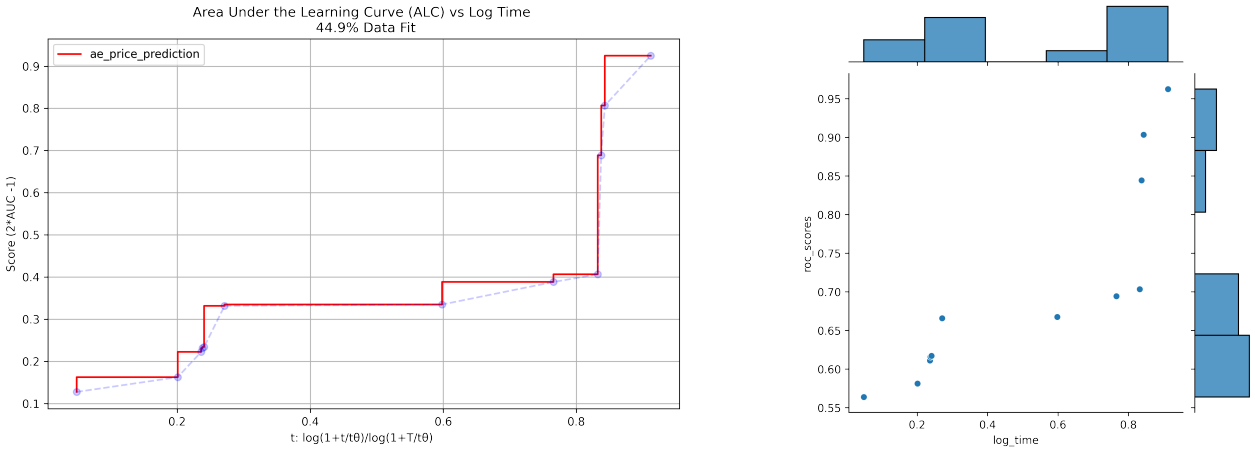


Figure A.1: Obtained Results for the ae price prediction dataset.

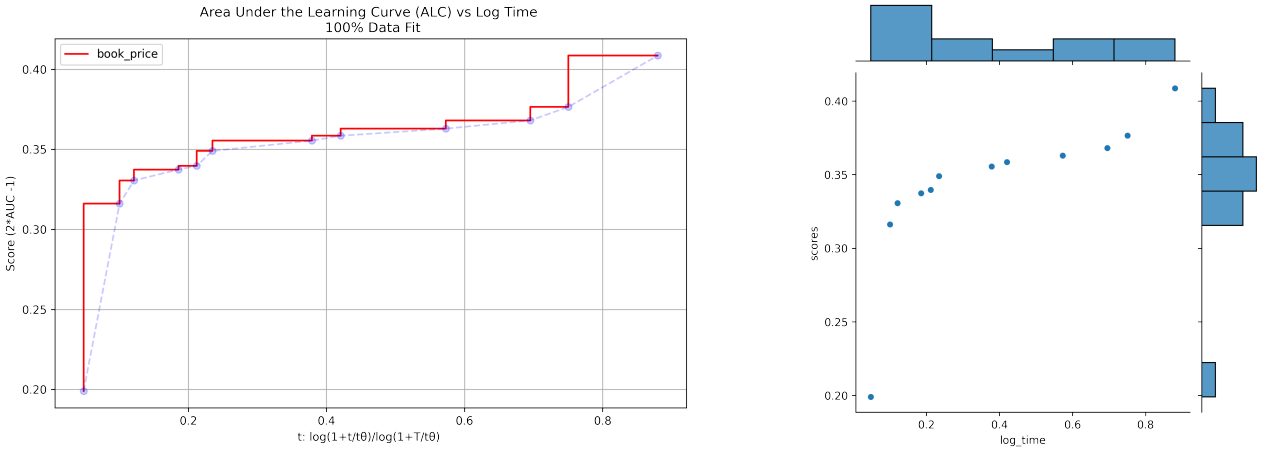


Figure A.2: Obtained Results for the book price dataset.

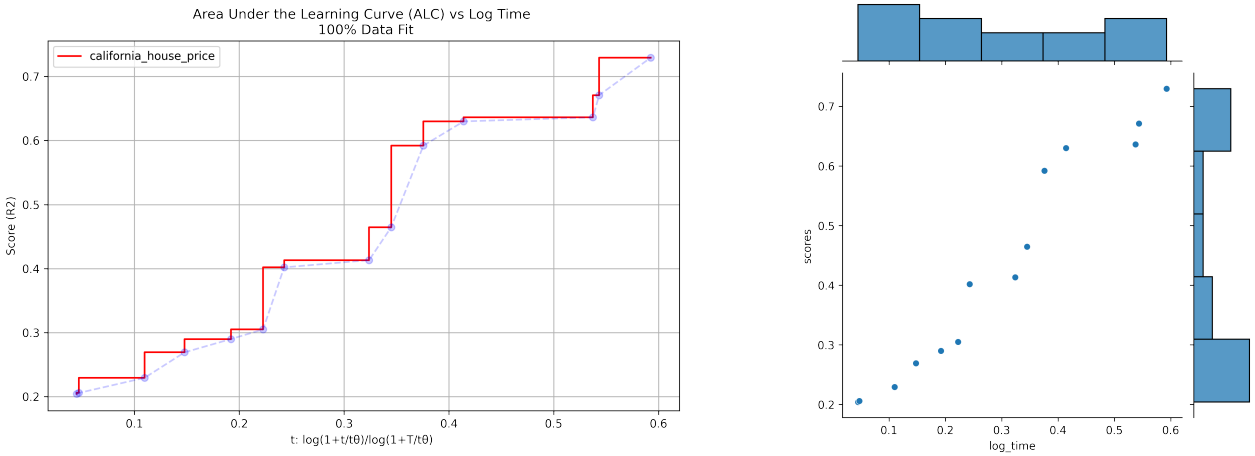


Figure A.3: Obtained Results for the california house price prediction dataset.

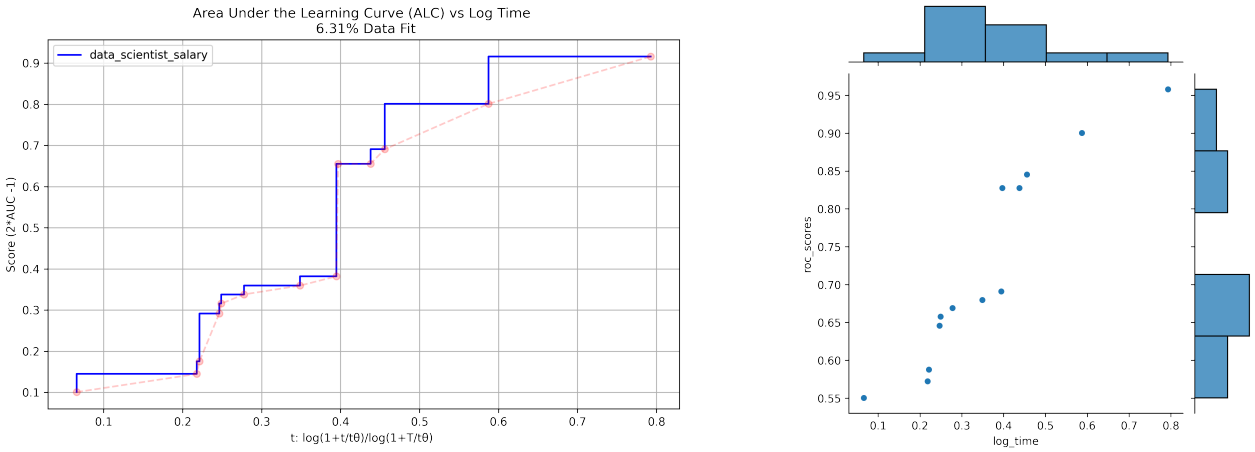


Figure A.4: Obtained Results for the salary price dataset.

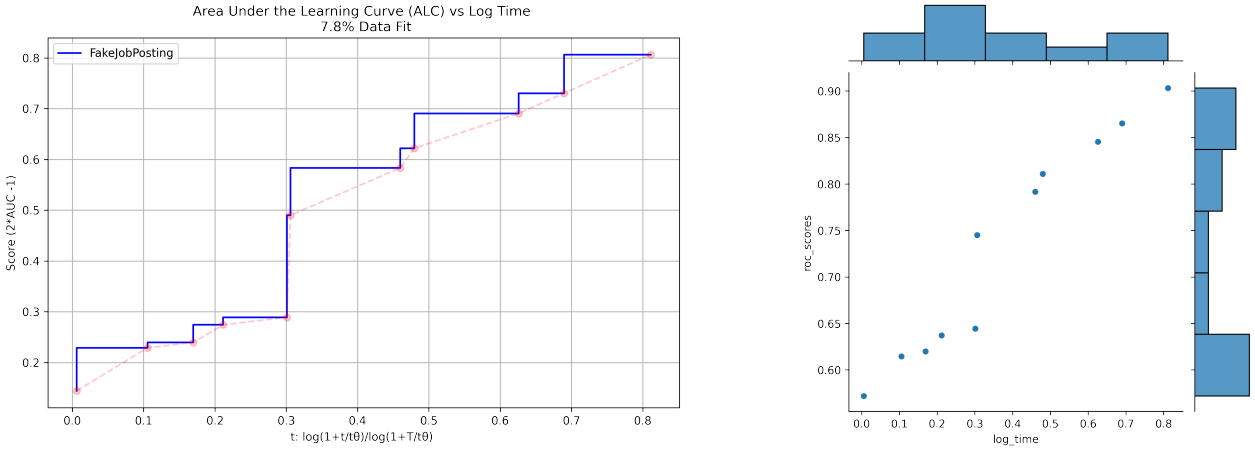


Figure A.5: Obtained Results for the fake job posting dataset.

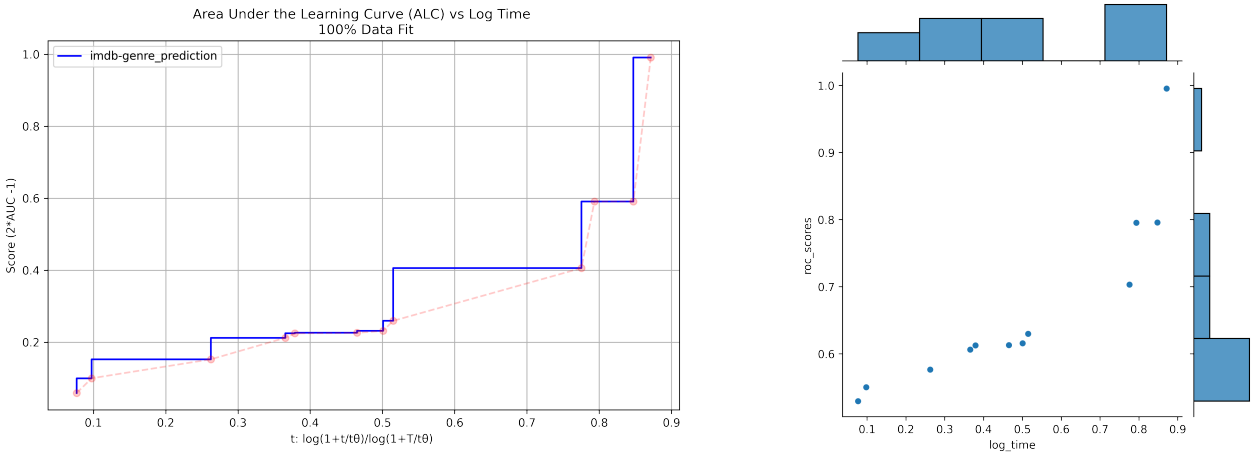


Figure A.6: Obtained Results for the imdb dataset.

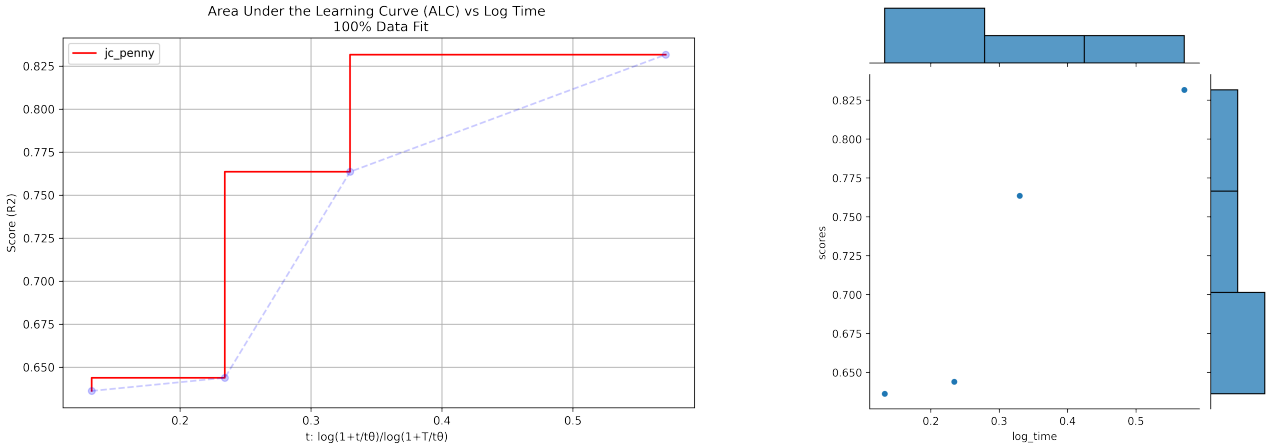


Figure A.7: Obtained Results for the jcpenny dataset.

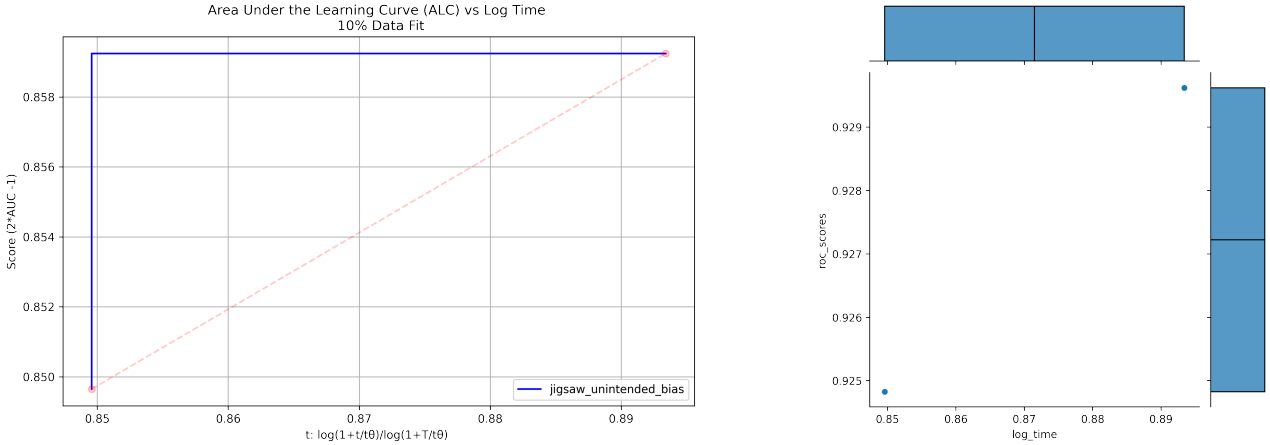


Figure A.8: Obtained Results for the jigsaw dataset.

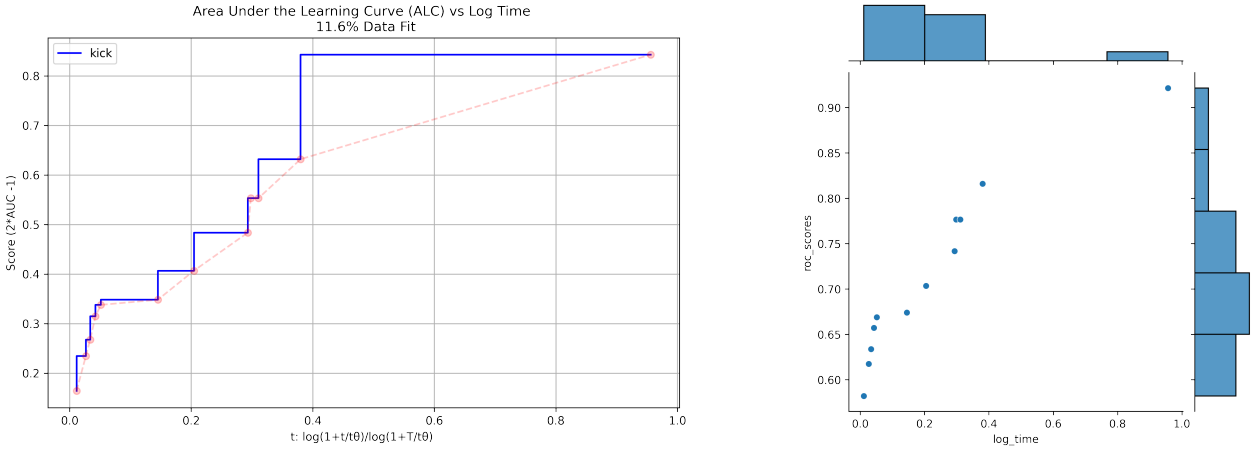


Figure A.9: Obtained Results for the kickstarter funding dataset.

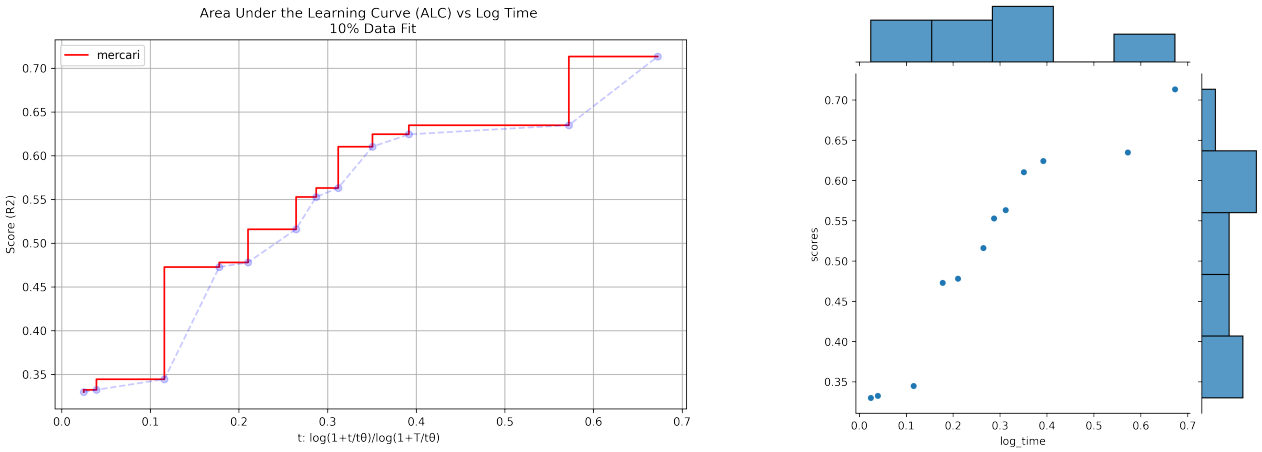


Figure A.10: Obtained Results for the mercari price prediction dataset.

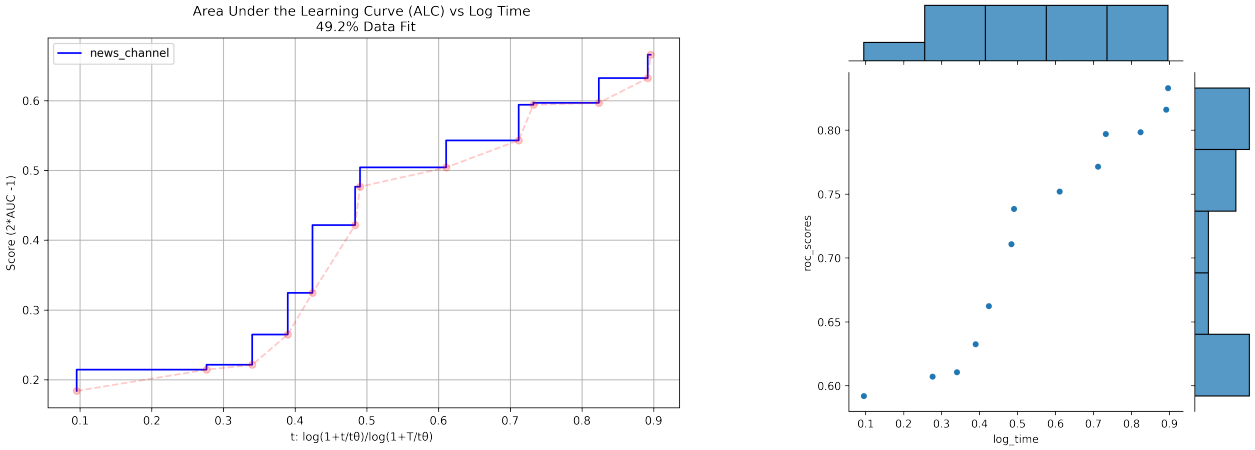


Figure A.11: Obtained Results for the news channel dataset.

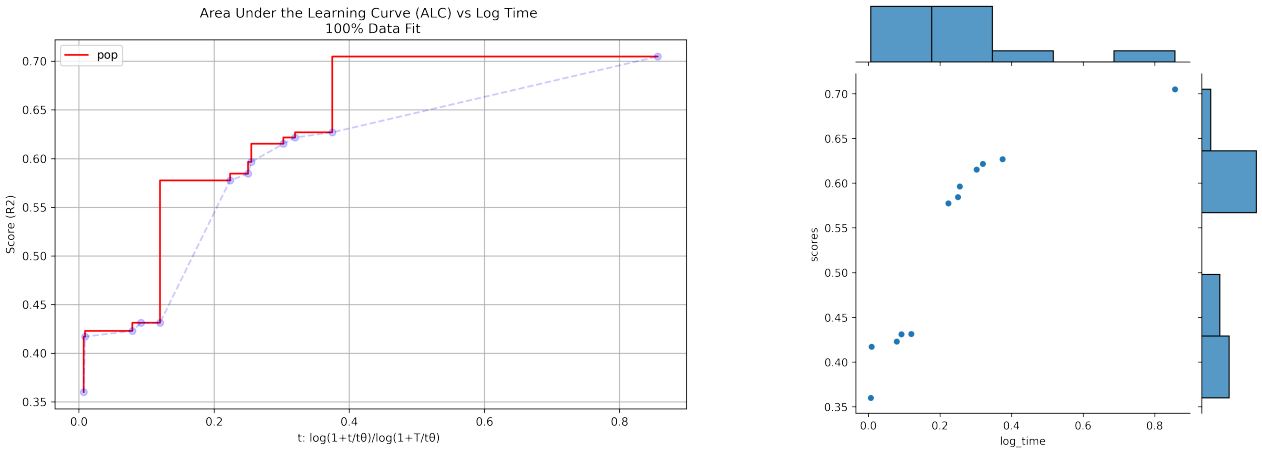


Figure A.12: Obtained Results for the popularity dataset.

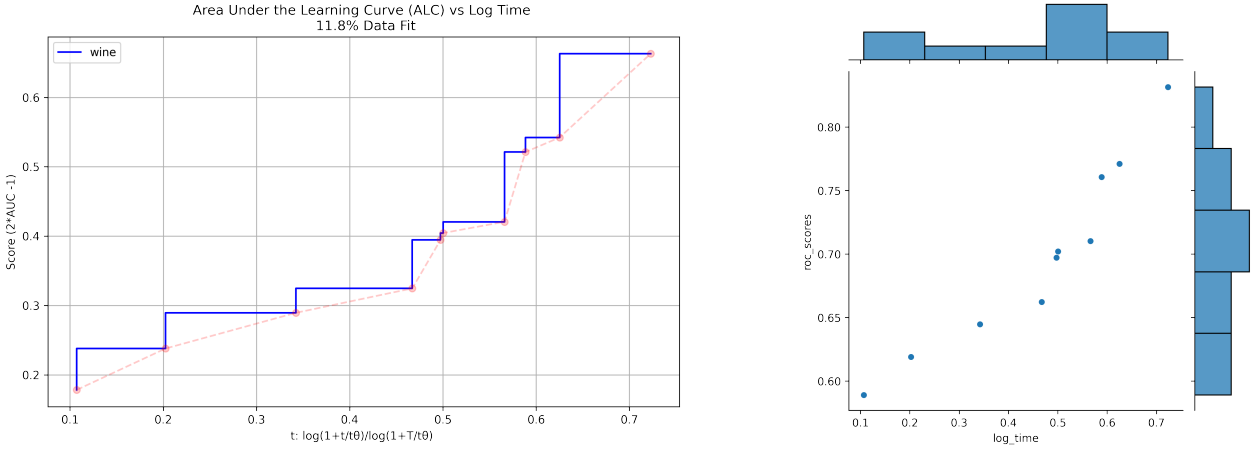


Figure A.13: Obtained Results for the wine dataset.

A.3 Downstream Models

We have mentioned several downstream models throughout this research. We included around 40 downstream models into our configuration space Θ, mainly:

```
downstrem_choices = ['CatBoostClassifier',
                     'CatBoostRegressor',
                     'XGBoostClassifier',
                     'XGBoostRegressor',
                     'stacked_ensemble_RF_L1',
                     'stacked_ensemble_XT_L1_1',
                     'stacked_ensemble_XT_L1_2',
                     'stacked_ensemble_LGB_L1_1',
                     'stacked_ensemble_LGB_L1_2',
                     'stacked_ensemble_LGB_L1_3',
                     'stacked_ensemble_CAT_L1_1',
                     'stacked_ensemble_CAT_L1_2',
                     'weighted_ensemble_RF_L1',
                     'weighted_ensemble_XT_L1_1',
                     'weighted_ensemble_XT_L1_2',
                     'weighted_ensemble_LGB_L1_1',
                     'weighted_ensemble_LGB_L1_2',
                     'weighted_ensemble_LGB_L1_3',
                     'weighted_ensemble_CAT_L1_1',
                     'weighted_ensemble_CAT_L1_2',
                     'stacked_ensemble_r__RF_L1',
                     'stacked_ensemble_r_XT_L1_1',
                     'stacked_ensemble_r_XT_L1_2',
                     'stacked_ensemble_r_LGB_L1_1',
                     'stacked_ensemble_r_LGB_L1_2',
                     'stacked_ensemble_r_LGB_L1_3',
                     'stacked_ensemble_r_CAT_L1_1',
                     'stacked_ensemble_r_CAT_L1_2',
                     'weighted_ensemble_r__RF_L1',
                     'weighted_ensemble_r_XT_L1_1',
                     'weighted_ensemble_r_XT_L1_2',
                     'weighted_ensemble_r_LGB_L1_1',
                     'weighted_ensemble_r_LGB_L1_2',
                     'weighted_ensemble_r_LGB_L1_3',
                     'weighted_ensemble_r_CAT_L1_1',
                     'weighted_ensemble_r_CAT_L1_2']
```

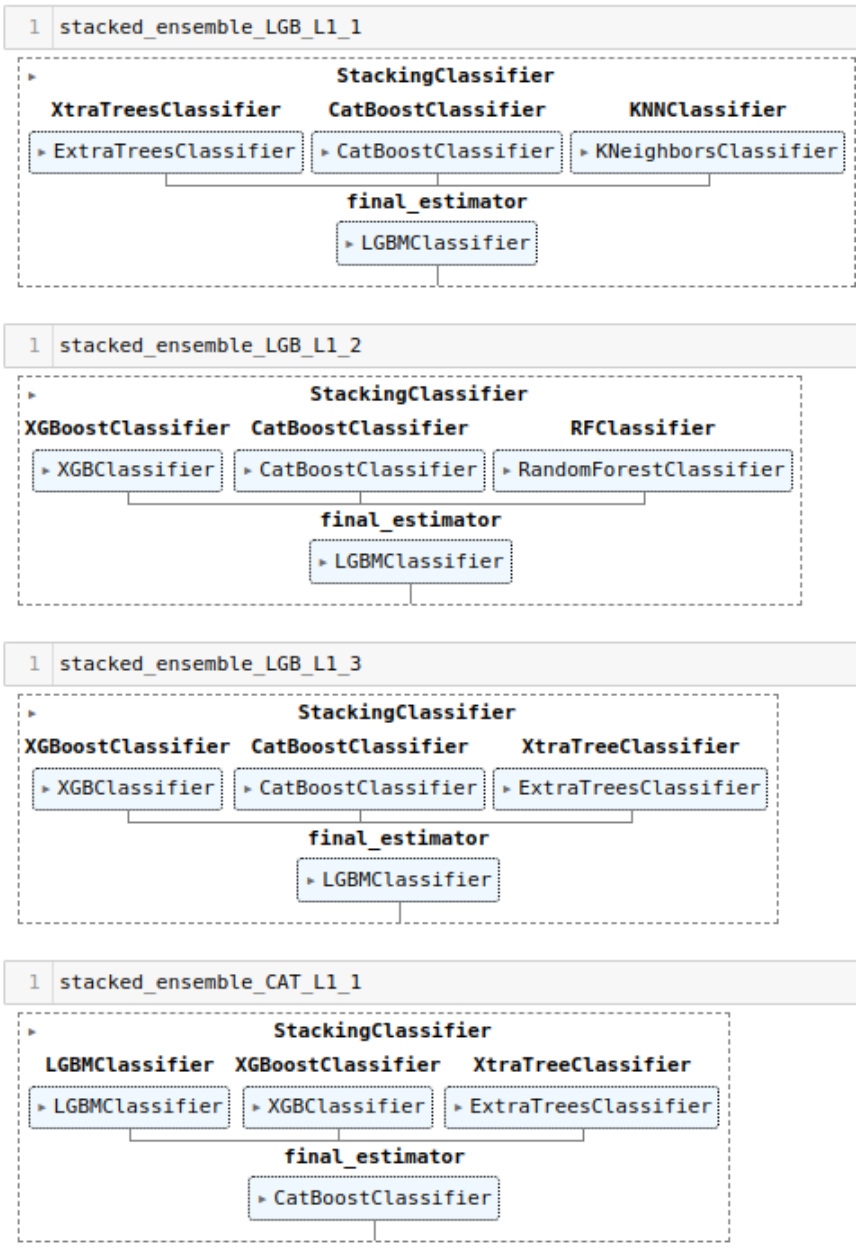


Figure A.14: Some of the stack ensemble models

Figure A.15: Some of the stack ensemble models

Figure A.16: Some of the weighted ensemble models

A.4 Further Studies on Pretrained Models

	public data		Multimodal Tasks VQAv2	SNLI-VE	HM	Language Tasks CoLA	SST-2	RTE	MRPC	QQP	MNLI	QNLI	STS-B	ImageNet linear eval
1	✓	$\text{BERT}_{\text{base}}$ [28]	–	–	–	54.6	92.5	62.5	81.9/87.6	90.6/87.4	84.4	91.0	88.1	–
2	✗	CLIP-ViT-B/16 [83]	55.3	74.0	63.4	25.4	88.2	55.2	74.9/65.0	76.8/53.9	33.5	50.5	16.0	80.2
3	✗	$\text{SimVLM}_{\text{base}}$ [109]	77.9	84.2	–	46.7	90.9	63.9	75.2/84.4	90.4/87.2	83.4	88.6	–	80.6
4	✓	VisualBERT [63]	70.8	77.3†	74.1‡	38.6	89.4	56.6	71.9/82.1	89.4/86.0	**81.6**	87.0	81.8	–
5	✓	$\text{UNITER}_{\text{base}}$ [16]	72.7	78.3	–	37.4	89.7	55.6	69.3/80.3	89.2/85.7	80.9	86.0	75.3	–
6	✓	$\text{VL-BERT}_{\text{base}}$ [101]	71.2	–	–	38.7	89.8	55.7	70.6/81.8	89.0/85.4	81.2	86.3	82.9	–
7	✓	ViLBERT [70]	70.6	75.7†	74.1‡	36.1	90.4	53.7	69.0/79.4	88.6/85.0	79.9	83.8	77.9	–
8	✓	LXMERT [102]	72.4	–	–	39.0	90.2	57.2	69.7/80.4	75.3/75.3	80.4	84.2	75.3	–
9	✓	UniT [43]	67.0	73.1	–	–	89.3	–	–	90.6/ –	81.5	**88.0**	–	–
10	✓	CLIP-ViT-B/16 (PMD)	59.8	73.5	56.6	11.0	83.5	53.1	63.5/68.7	75.4/43.0	32.9	49.5	13.7	73.0
11	✓	FLAVA (ours)	**72.8**	**79.0**	**76.7**	**50.7**	**90.9**	**57.8**	**81.4/86.9**	**90.4/87.2**	80.3	87.3	**85.7**	**75.5**

Table 5. **Comparing FLAVA (Table 4 column 6) with previous models on multimodal tasks, language tasks, and ImageNet linear evaluation.** We report results on development sets of the GLUE benchmark [108]. We report Matthew's Correlation for CoLA; accuracy/F1 for MRPC and QQP; the Pearson/Spearman correlation for STS-B; average of mismatched and matched accuracy for MNLI; AUROC for Hateful Memes; test-dev VQA score for VQAv2 and accuracy for all other tasks. The results for BERT and other VLP methods on GLUE benchmark are obtained from [47]. The results on V&L tasks are from original papers. For UniT, we use "shared, (COCO init.)" version. Note that SimVLM is pretrained on an order of magnitude more data than FLAVA (1.8B vs 70M). †: taken from [95]; ‡: taken from [53]. The overall best result among the multimodal approaches is underlined while **bold** signifies the best model trained on public data.

Figure A.17: Comparing FLAVA with different pre-trained models across different vision-language tasks [40].

		MIM 1	MLM 2	FLAVA_{C} 3	FLAVA_{MM} 4	FLAVA w/o init 5	FLAVA 6	CLIP 7	CLIP 8
Datasets	Eval method	PMD	PMD	PMD	PMD	(PMD+IN-1k+CCNews+BC)		PMD	400M [83]
MNLI [111]	fine-tuning	–	73.23	70.99	76.82	78.06	**80.33**	32.85	33.52
CoLA [110]	fine-tuning	–	39.55	17.58	38.97	44.22	**50.65**	11.02	25.37
MRPC [29]	fine-tuning	–	73.24	76.31	79.14	78.91	**84.16**	68.74	69.91
QQP [49]	fine-tuning	–	86.68	85.94	88.49	98.61	**88.74**	59.17	65.33
SST-2 [97]	fine-tuning	–	87.96	86.47	89.33	90.14	**90.94**	83.49	88.19
QNLI [88]	fine-tuning	–	82.32	71.85	84.77	86.40	**87.31**	49.46	50.54
RTE [7, 25, 36, 40]	fine-tuning	–	50.54	51.99	51.99	54.87	**57.76**	53.07	55.23
STS-B [1]	fine-tuning	–	78.89	57.28	84.29	83.21	**85.67**	13.70	15.98
NLP Avg.		–	71.55	64.80	74.22	75.55	**78.19**	46.44	50.50
ImageNet [90]	linear eval	41.79	–	74.09	74.34	73.49	**75.54**	72.95	80.20
Food101 [11]	linear eval	53.30	–	87.77	87.53	87.39	**88.51**	85.49	91.56
CIFAR10 [58]	linear eval	76.20	–	**93.44**	92.37	92.63	92.87	91.25	94.93
CIFAR100 [58]	linear eval	55.57	–	**78.37**	78.01	76.49	77.68	74.40	81.10
Cars [56]	linear eval	14.71	–	**72.12**	72.07	66.81	70.87	62.84	85.92
Aircraft [74]	linear eval	13.83	–	**49.74**	48.90	44.73	47.31	40.02	51.40
DTD [20]	linear eval	55.53	–	76.86	76.91	75.80	**77.29**	73.40	78.46
Pets [79]	linear eval	34.48	–	**84.98**	84.93	82.77	84.82	79.61	91.66
Caltech101 [32]	linear eval	67.36	–	94.91	95.32	94.95	**95.74**	93.76	95.51
Flowers102 [76]	linear eval	67.23	–	96.36	**96.39**	95.58	96.37	94.94	97.12
MNIST [60]	linear eval	96.40	–	98.39	98.58	**98.70**	98.42	97.38	99.01
STL10 [21]	linear eval	80.12	–	98.06	98.31	98.32	**98.89**	97.29	99.09
EuroSAT [41]	linear eval	95.48	–	97.00	96.98	97.04	**97.26**	95.70	95.38
GTSRB [100]	linear eval	63.14	–	78.92	77.93	77.71	**79.46**	76.34	88.61
KITTI [35]	linear eval	86.03	–	87.83	88.84	88.70	**89.04**	84.89	86.56
PCAM [106]	linear eval	85.10	–	85.02	85.51	**85.72**	85.31	83.99	83.72
UCF101 [98]	linear eval	46.34	–	82.69	82.90	81.42	**83.32**	77.85	85.17
CLEVR [52]	linear eval	61.51	–	79.35	**81.66**	80.62	79.66	73.64	75.89
FER 2013 [38]	linear eval	50.98	–	59.96	60.87	58.99	**61.12**	57.04	68.36
SUN397 [113]	linear eval	52.45	–	81.27	81.41	81.05	**82.17**	79.96	82.05
SST [83]	linear eval	57.77	–	56.67	**59.25**	56.40	57.11	56.84	74.68
Country211 [83]	linear eval	8.87	–	27.27	26.75	27.01	**28.92**	25.12	30.10
Vision Avg.		57.46	–	79.14	79.35	78.29	**79.44**	76.12	82.57
VQAv2 [39]	fine-tuning	–	–	67.13	71.69	71.29	**72.49**	59.81	54.83
SNLI-VE [114]	fine-tuning	–	–	73.27	78.36	78.14	**78.89**	73.53	74.27
Hateful Memes [53]	fine-tuning	–	–	55.58	70.72	**77.45**	76.09	56.59	63.93
Flickr30K [81] TR R@1	zero-shot	–	–	68.30	**69.30**	64.50	67.70	60.90	82.20
Flickr30K [81] TR R@5	zero-shot	–	–	93.50	92.90	90.30	**94.00**	88.90	96.60
Flickr30K [81] IR R@1	zero-shot	–	–	60.56	63.16	60.04	**65.22**	56.48	62.08
Flickr30K [81] IR R@5	zero-shot	–	–	86.68	87.70	86.46	**89.38**	83.60	85.68
COCO [66] TR R@1	zero-shot	–	–	43.08	**43.48**	39.88	42.74	37.12	52.48
COCO [66] TR R@5	zero-shot	–	–	75.82	**76.76**	72.84	**76.76**	69.48	76.68
COCO [66] IR R@1	zero-shot	–	–	37.59	**38.46**	34.95	38.38	33.29	33.07
COCO [66] IR R@5	zero-shot	–	–	67.28	**67.68**	64.63	67.47	62.47	58.37
Multimodal Avg.		–	–	66.25	69.11	67.32	**69.92**	62.02	67.29
Macro Avg.		19.15	23.85	70.06	74.23	73.72	**75.85**	61.52	66.78

Table 4. **Comparing our full FLAVA pretraining with other settings**, where FLAVA gets the highest macro average score. MNLI numbers are average of MNLI-m and MNLI-mm. MRPC and QQP numbers are average of accuracy and F1. We report PCC for CoLA, MCC for STS-B, and AUROC for Hateful Memes, respectively. We perform zero-shot text retrieval and image retrieval (TR and IR) on Flickr30K and COCO based on their matching scores from the contrastive loss and report top-1 and top-5 recall. For all other tasks we report accuracy. Column 8 is the best released model in [83] based on ViT-B/16 pretrained on 400M image-text pairs. The overall best result is underlined while **bold** signifies the best on public data (PMD and unimodal).

Figure A.18: Comparison of FLAVA against different pre-trained vision-language Transformer based models across various tasks and objectives [40].

Metric	Model	Existence	Plurality	Counting			Sp.rel.‡	Action		Coreference		Foil-it!	Avg.
		quantifiers	number	balanced	sns.†	adv.†	relations	repl.†	actant swap	standard	clean		
	Random	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0
acc_r	GPT1*	61.8	53.1	51.2	48.7	69.5	**77.2**	65.4	72.2	45.6	45.2	77.5	60.7
	GPT2*	58.0	51.9	51.6	49.8	45.3	75.0	66.8	**76.9**	54.5	50.0	80.7	60.1
	CLIP	66.9	56.2	62.1	62.5	57.5	64.3	**75.6**	68.6	52.1	49.7	**88.8**	64.0
	LXMERT	78.6	64.4	62.2	69.2	42.6	60.2	54.8	45.8	46.8	44.2	87.1	59.6
	ViLBERT	65.5	61.2	58.6	62.9	73.7	57.2	70.7	68.3	47.2	48.1	86.9	63.7
	12-in-1	**95.6**	**72.4**	**76.7**	**80.2**	**77.3**	67.7	65.9	58.9	**75.7**	**69.2**	86.9	**75.1**
	VisualBERT	39.7	45.7	48.2	48.2	50.0	39.7	49.2	44.4	49.5	47.6	48.5	46.4
acc	LXMERT	55.8	55.1	52.0	55.4	49.9	50.8	51.1	48.5	49.8	49.0	70.8	53.5
	ViLBERT	2.4	50.3	50.7	50.6	51.8	49.9	52.6	50.4	50.0	50.0	55.9	51.3
	12-in-1	**89.0**	**62.0**	**64.9**	**69.2**	**66.7**	**53.4**	**57.3**	**52.2**	**54.4**	**54.3**	**71.5**	**63.2**
	VisualBERT	49.3	46.5	48.3	47.8	50.0	49.3	48.8	49.7	50.0	50.0	46.6	48.8
$\min(p_c, p_f)$	LXMERT	41.6	**42.2**	50.9	50.0	37.3	**28.4**	35.8	36.8	**18.4**	**17.3**	**69.3**	38.9
	ViLBERT	47.9	2.1	24.4	24.7	17.5	1.5	11.9	7.1	1.3	1.9	12.9	13.9
	12-in-1	**85.0**	33.4	**64.3**	**61.7**	**59.5**	13.3	**47.8**	**37.6**	15.8	13.5	48.8	**43.7**
	VisualBERT	1.3	0.3	0.0	0.0	0.0	1.3	0.0	0.0	0.0	0.0	0.2	0.3
$AUROC$ $\times 100$	LXMERT	60.5	57.3	53.8	57.7	50.5	51.9	52.1	47.6	49.8	49.5	76.9	55.2
	ViLBERT	52.5	54.1	50.8	51.6	53.5	51.2	57.2	**57.8**	49.9	49.9	75.2	54.9
	12-in-1	**96.3**	**67.4**	**72.0**	**77.8**	**75.1**	**55.8**	**61.3**	55.0	**59.8**	**59.6**	**81.0**	**69.2**
	VisualBERT	28.9	29.0	24.5	16.5	20.9	45.2	17.7	36.3	45.3	46.3	28.5	30.8

Table 2: Performance of unimodal and multimodal models on the VALSE benchmark according to different metrics. We bold-face the best overall result per metric, and underscore all results below (or at) the random baseline. acc_r is a pairwise ranking accuracy where a prediction is considered correct if $p(caption, img) > p(foil, img)$. Precision p_c and foil precision p_f are *competing* metrics where naïvely increasing one can decrease the other: therefore *looking at the smaller number among the two gives a good intuition of how informed is a model prediction.* †**sns.** Counting small numbers. **adv.** Counting adversarial. **repl.** Action replacement. ‡ **Sp.rel.** Spatial relations. *Unimodal text-only models that do not use images as input. CLIP is only tested in pairwise ranking mode (fn. 6).

Figure A.19: Comparison of different vision-language Transformer architectures on the VALSE benchmark [34]

Method	Multimodal Pretraining data			Pretraining Objectives				Target Modalities			
	public	dataset(s)	size	Contr.	ITM	Masking	Unimodal	V	CV&L	MV&L	L
CLIP [83]	✗	WebImageText	400M	✓	–	–	–	✓	✓	–	–
ALIGN [50]	✗	JFT	1.8B	✓	–	–	–	✓	✓	–	–
SimVLM [109]	✗	JFT	1.8B	–	–	PrefixLM	CLM	*	✓	✓	✓
UniT [43]	–	None	–	–	–	–	–	*	–	✓	✓
VinVL [118]	✓	Combination	9M	✓	–	MLM	–	–	✓	✓	–
ViLT [54]	✓	Combination	10M	–	✓	MLM	–	–	✓	✓	–
ALBEF [62]	✓	Combination	5M	✓	✓	MLM	–	–	✓	✓	–
FLAVA (ours)	✓	PMD (Tbl. 2)	70M	✓	✓	MMM	MLM+MIM	✓	✓	✓	✓

Table 1. Comparison of recent models in different modalities. CV&L and MV&L stands for cross-modal and multi-modal vision-and-language. * means the modality is partially targeted (SimVLM [109] and UniT [43] include ImageNet and object detection, respectively).

Figure A.20: This figure compares model flexibility across different input modalities [40].

Model	Flickr30k-IR R@1 / R@5 / R@10	Flickr30k-TR R@1 / R@5 / R@10	SNLI-VE Val / Test	VQA test-dev / -std	CoCo Caption BLUE4 / CIDEr
ViLBERT-base	58.20 / 84.90 / 91.52	-	-	70.55 / 70.92	-
VLP-base	-	-	-	70.5 / 70.7	36.5 / 116.9
UNITER-base	72.52 / 92.36 / **96.08**	85.90 / 97.10 / 98.80	78.59 / 78.28	72.70 / 72.91	-
Oscar-base	-	-	-	73.16 / 73.44	36.5 / 123.7
Villa-base	**74.74** / 92.86 / 95.82	86.60 / 97.90 / **99.20**	79.47 / 79.03	73.59 / 73.67	-
Ernie-ViL-base	74.44 / 92.72 / 95.94	86.70 / 97.80 / 99.00	-	72.62 / 72.85	-
UNIMO-base	74.66 / **93.40** / **96.08**	**89.70** / **98.40** / 99.10	**80.00** / **79.10**	**73.79** / **74.02**	**38.8** / **124.4**
UNITER-large	75.56 / 94.08 / 96.76	87.30 / 98.00 / 99.20	79.39 / 79.38	73.82 / 74.02	-
Oscar-large	-	-	-	73.61 / 73.82	37.4 / **127.8**
Villa-large	76.26 / **94.24** / 96.84	87.90 / 97.50 / 98.80	80.18 / 80.02	74.69 / 74.87	-
ERNIE-ViL-large	76.70 / 93.58 / 96.44	88.10 / 98.00 / 99.20	-	74.75 / 74.93	-
UNIMO-large	**78.04** / **94.24** / **97.12**	**89.40** / **98.90** / **99.80**	**81.11** / **80.63**	**75.06** / **75.27**	**39.6** / 127.7

Table 1: Evaluation results on the multi-modal downstream tasks.

Model	SST-2 Acc	MNLI Acc-(m/mm)	CoLA Mat	STS-B Per	CoQA Acc	SQuAD-QG B4/ME/R-L	CNNDM R-1/2/L	Gigaword R-1/2/L
BERT-base	92.7	84.4 / -	-	-	-	-	-	-
RoBERTa-base	94.8	-	63.6	-	77.4	22.15/24.58/51.12	42.31/20.04/39.49	38.65/19.66/36.04
UNIMO-base	**95.1**	**86.8/86.7**	**65.4**	**91.0**	**80.2**	**22.78/25.24/51.34**	**42.42/20.12/39.61**	**38.80/19.99/36.27**
w/o single-modal	82.0	59.9/64.9	15.0	88.8	67.1	17.09/21.04/46.47	41.06/19.01/38.23	38.06/18.91/35.41
BERT-large	93.2	86.6/-	60.6	90.0	-	-	-	-
RoBERTa-large	96.4	**90.2/90.2**	68.0	92.4	**85.1**	23.39/25.73/52.11	43.10/20.29/40.24	39.32/20.01/36.58
XLNet-large	95.6	89.8/-	63.6	91.8	-	-	-	-
UniLM-large	94.5	87.0/85.9	61.1	87.7	82.5	22.12/25.06/51.07	43.33/20.21/40.51	38.45/19.45/35.75
UNIMO-large	**96.8**	89.8/89.5	**68.5**	**92.6**	84.9	**24.59/26.39/52.47**	**43.51/20.65/40.63**	**39.71/20.37/36.88**

Table 2: Comparison on the single-modal downstream tasks. R-1, R-2 and R-L denote ROUGE-1, ROUGE-2 and ROUGE-L, respectively. Mat, Per, B4 and ME denote Matthews correlation coefficient, Pearson correlation coefficient, BLUE4 and METEOR (Lavie and Agarwal, 2007), respectively. "w/o single-modal" denotes removing the single-modal learning process on the single-modal data from UNIMO, which is similar to UNITER-base (Chen et al., 2020b). The results on SST-2, MNLI, CoLA, STS-B and CoQA are evaluated on the dev set. The results of RoBERTa on the generation tasks CoQA, SQuAD-QG, CNNDM and Gigaword are evaluated by utilizing the UNIMO architecture initialized with pre-trained parameters of RoBERTa.

Figure A.21: Comparison of UNIMO against different Transformer based vision-language models across different datasets.