A Two-Phase Recall-and-Select Framework for Fast Model Selection

Jianwei Cui, Wenhang Shi, Honglin Tao, Wei Lu, Xiaoyong Du
Renmin University of China, Beijing, China
{cuijianwei, wenhangshi, honglintao, lu-wei, duyong}@ruc.edu.cn

***Abstract*—As the ubiquity of deep learning in various machine learning applications has amplified, a proliferation of neural network models has been trained and shared on public model repositories. In the context of a targeted machine learning assignment, utilizing an apt source model as a starting point typically outperforms the strategy of training from scratch, particularly with limited training data. Despite the investigation and development of numerous model selection strategies in prior work, the process remains time-consuming, especially given the ever-increasing scale of model repositories. In this paper, we propose a two-phase (coarse-recall and fine-selection) model selection framework, aiming to enhance the efficiency of selecting a robust model by leveraging the models' training performances on benchmark datasets. Specifically, the coarse-recall phase clusters models showcasing similar training performances on benchmark datasets in an offline manner. A light-weight proxy score is subsequently computed between this model cluster and the target dataset, which serves to recall a significantly smaller subset of potential candidate models in a swift manner. In the following fine-selection phase, the final model is chosen by fine-tuning the recalled models on the target dataset with successive halving. To accelerate the process, the final fine-tuning performance of each potential model is predicted by mining the model's convergence trend on the benchmark datasets, which aids in filtering lower performance models more earlier during fine-tuning. Through extensive experimentation on tasks covering natural language processing and computer vision, it has been demonstrated that the proposed methodology facilitates the selection of a high-performing model at a rate about 3x times faster than conventional baseline methods. Our code is available at https://github.com/plasware/two-phase-selection.** R2.O3

***Index Terms*—model selection, model clustering**

I. Introduction

Nowadays, a plethora of neural networks, meticulously trained in diverse fields such as natural language processing and computer vision, are readily available. These models are commonly hosted on public repositories or model hubs like HuggingFace [1]. Given the wide range of these models' training data, it is plausible that for any specific downstream task, there exists a trained model whose domain distribution of the training dataset is well-transferable for the target task. Employing such a pre-trained model for parameter initialization, followed by fine-tuning on the target dataset often leads to an enhanced performance. This is attributable to the effective transfer and adaptation of the knowledge garnered from the original model to the target task. Therefore, how to select the optimal pre-trained model from a vast collection is crucial for achieving superior training results in a new task, especially when the training data is limited [2] [3].

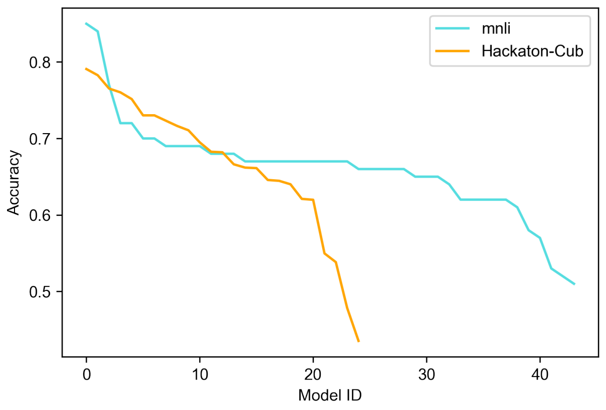


Fig. 1. Fine-tuning performance of 44 and 25 pretrained models on NLP task MNLI [4] and CV task CC6204-Hackaton-Cu [5]. The x and y-axis show pre-trained models' ID and their performances on the dataset, respectively. It's noted that for each dataset, the ids are sorted by the model accuracy desc.

The principal objective of model selection is to efficiently identify a well-suited pre-trained model from a repository for a novel machine learning task. However, the growing repository, though providing potential for improved task initialization, yet escalates the challenge of pinpointing the optimal model. Fig. 1 illustrates the fine-tuning results of different models on two distinct machine learning tasks. R6.O1 Although the model pool contains a few models that exhibit commendable performance on the target task, they are markedly outnumbered by models that perform poorly. This discrepancy underscores the increasing complexity of model selection as the volume of available models surges.

The current body of research on this challenge can be bifurcated into two main categories. The first category is centered on the development of lightweight proxy tasks to predict the post-fine-tuning performance of pre-trained models. While the methods offer computational efficiency by obviating the need for direct fine-tuning, they tend to be more prone to selecting sub-optimal models [6]. Conversely, the second category of methods employ a model selection strategy during the fine-tuning process on the target dataset, utilizing a success-halving approach to retain only high-performance models at each training iteration [2] [3]. However, as illustrated in Fig. 1, only a minor fraction of models in the repository are appropriate for a specific downstream task. Therefore, even with the successive halving strategy, it is computationally inefficient to fine-tune all models in the repository, given that each model needs to be loaded and trained for at least one iteration. Additionally, the efficacy and efficiency of these existing methods decline as the number of pre-trained models continues to expand.

Hence, we propose a two-phase model selection framework, a hybrid approach that amalgamates the advantages of both aforementioned categories, enabling the efficient selection of a suitable pre-trained model for a novel task. We split the model selection to two phases: the first phase, referred as the coarse-recall phase, is designed to identify a handful of promising model candidates based on lightweight proxy tasks. Following this, the second fine-selection phase only necessitates the fine-tuning of models recalled from the first phase to identify the most optimal model. This method significantly improves the efficiency of selecting a suitable model from a large repository, as fine-tuning is only carried out on a substantially reduced subset of models.

The coarse-call phase computes a light proxy task for each model on the target dataset, keeping only the models with high scores for fast filtering. Although the use of proxy tasks avoids fine-tuning, computing a score for each model still makes model loading and inference necessary and inefficient, especially when the model number increases. To speed up, we propose to cluster similar models based on their performances on a set of benchmark datasets. This is inspired by the fact that there is overlap in the training data of public models, so that models that perform similarly on the standard dataset will perform similarly on the new dataset. Specifically, we construct a performance matrix by training each model offline on all benchmark datasets and saving the corresponding performances. Then we cluster the models based on their performance vectors, and each time a new task arrives, we only compute scores for the clusters' representative model. By mining the similarity among models' training, we avoid repeated online computation of proxy scores for similar models and make the model selection more efficient.

As fine-tuning is more time-consuming, the second fine-selection phase needs to apply more efficient filter strategy to avoid wasting time training low-performance models at early training steps. This is motivated by the model performance consistency at the beginning and end of the training [7]. In this paper, we also apply the successive halving algorithm to filter at least half number of models with lower performance at each training iteration. Meanwhile, as illustrated in Fig. 2(b), it is plausible to filter more than half number of models if we can predict the final training performance. Again, we resort to mine the convergence processes between a pre-trained model and benchmark datasets as illustrated in Fig. 2(b). Specifically, for every recalled model, we collect the training processes on benchmark datasets, and cluster training processes with similar validation accuracy to form a convergence trend which could predict final test performance range at each iteration step. Then, after the model is fine-tuned on the target dataset for a few steps, we can assign a convergence trend to the model if the current training performance is closed to the training performance of the convergence trend at current step. By this way, the final training performance of a pre-trained model on the target dataset could be predicted more accurate, which could helps to filter more models at early steps.

To this end, we summarize the contributions of this paper as follows:

- We propose a two-phase model selection framework. The first coarse-recall phase employs a lightweight proxy score to identify a considerably smaller set of promising model candidates. Subsequently, the second fine-selection phase exclusively fine-tunes and filters models from this refined set.
- To further accelerate the process, we propose mining the training performances of pre-trained models on a collection of benchmark datasets, subsequently clustering similar models. Through clustering, we circumvent computing proxy scores online for each model in the first phase and enhance the accuracy of performance predictions for the left models in the second phase, thereby achieving more efficient and precise selection.
- We conduct extensive experiments on a substantial variety of training models, encompassing both natural language processing and computer vision domains. The results demonstrate that our proposed framework can effectively identify superior performing models with increasing efficiency, about 3x compared to successive halving and 5x compared to brute force methods.

II. The Framework

A. Preliminaries

Model Repository. The model repository is a set of pre-trained models, denoted as $M = \{m_1, m_2, ..., m_n\}$. Here, a pre-trained model m_j is a neural network model already trained on an upstream dataset with different learning methods, such as masked language model in natual language processing [8] or image classification for computer vision.

Benchmark Datasets. The benchmark datasets comprise representative datasets from the respective domain, such as the GLUE [4] for natural language process and various subsets of ImageNet [9] for computer vision. We use $D = \{d_1, d_2, ..., d_m\}$ to denote the benchmark datasets.

Performance Matrix. The performance matrix records the test results of pre-trained models finetuned on benchmark datasets, denoted as $Matrix(D, M)$ with the value $Matrix(D, M)[i][j]$ is the training performance of the pre-trained model m_j on the benchmark dataset d_i, also denoted as $p(d_i|m_j)$. The training performance could be measured through different metrics for different task, like accuracy for classification tasks.

Model Cluster. A model cluster contains a group of models having similar training performances on benchmark datasets. Fig. 2(b) illustrates two model clusters, where C_1 contains three models m_i, m_j, m_k and C_2 contains two models m_x, m_y respectively.

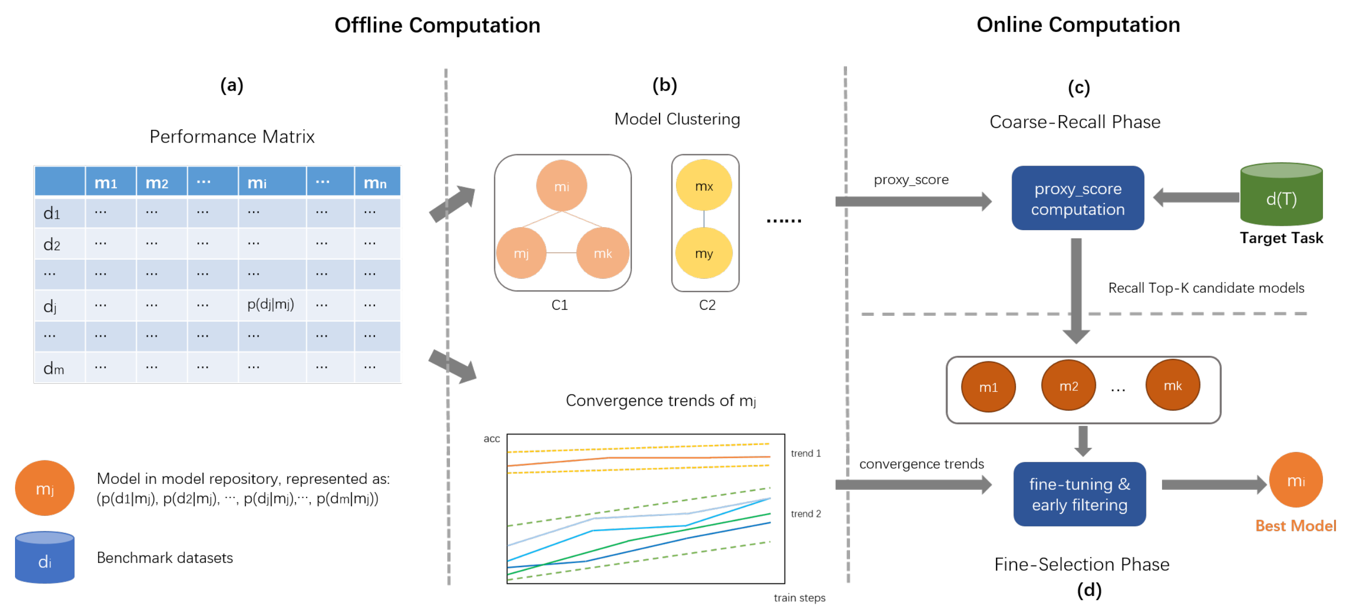


Fig. 2. The framework of two-phase model selection: (a) performance matrix, (b) model clustering based on the performance matrix, and convergence trends mining by clustering convergence processes of a pre-trained model on benchmark datasets, (c) coarse-recall phase running recall strategy based on the proxy score computation between a model cluster and the target dataset, and (d) fine-selection phase fine-tunes the recalled models and filters poorly-performance models according to convergence trend. Both (a) and (b) are maintained offline and could be used for any new task.

Convergence Trend. The convergence trend clusters datasets into different classes on which the model has a similar training process. We use $CT(m_j)_t$ to denote the class of convergence trend to which the dataset belongs at the t validation for model m_j. Fig. 2(b) illustrates two convergence trends for m_j. The first convergence trend $CT(m_j)_t[0]$ represents a convergence process which could achieve relative higher final training performance, and the second convergence trend $CT(m_j)_t[1]$ achieves lower final training performance. Based on the mined convergence trend, the final performance after training could be predicted more accurate at early training steps.

Proxy Score. The proxy score is computed based on the proxy task which predicts the training performance $p(d_i|m_j)$ without actually fine-tuning m_j on d_i, denoted as $proxy\_score(d_i|m_j)$. Several proxy tasks have been developed to predict $p(d_i|m_j)$, such as LEEP [10], KNN classification [11] etc. In this paper, we apply the LEEP score as the $proxy\_score$, which is the average log-likelihood of the expected empirical predictor result of a source model on the target dataset. The advantage of the LEEP score lies in two aspects. Firstly, LEEP could be applied to heterogeneous target tasks that have different label spaces compared with the pre-trained models. Secondly, the computation of the LEEP score does not need extra training, which is more efficient compared
R6.O3 with the KNN method.

Target Task. We use T to denote the target task and $d(T)$ to denote the training dataset of T. The proxy score for model m_j on the target dataset is denoted as $proxy\_score(d(T)|m_j)$, or $proxy\_score(T|m_j)$.

B. Framework Overview

Fig. 2 illustrates the whole process of the two-phase framework containing two computation parts:

Offline. The construction of the performance matrix constitutes the core offline process, wherein we finetune each model in M on the benchmark datasets and record the validation and test results throughout the training process. Subsequently, we execute model clustering based on this matrix, resulting in $MC = \{C_1, C_2, ..., C_p\}$. Although this offline computation is time consuming, this part could be only computed once and then the generated model clusters and intermediate results could be used in the online computation directly.

Online. This part implements the two-phase model selection computation for a new task T. The coarse-recall phase, denoted as $CR = corase\_recall(M|MC, d(T))$, returns K candidate models from model sets M, prone to achieve high training performance on $d(T)$ by computing the proxy score for model clusters MC on $d(T)$. Then, the fine-selection phase, denoted as $fine\_selection(CR|CT, d(T))$, returns the final selected model from the recalled models CR with convergence trend CT.

III. Coarse Recall

The coarse-recall phase aims to efficiently identify a much smaller number of candidate models which tend to achieve good result on the target task. Fig. 2(c) illustrates the overall steps of coarse-recall phase. We firstly present the model clustering process, and then introduce proxy score computation for model clusters on target dataset to return the recalled models.

A. Model Clustering

The computation of $proxy\_score(T|m_j)$ needs to load the model into memory and do inference computation on the target dataset. The load and inference step may consume dozens of seconds for a pre-trained model with millions of parameters and a target dataset with hundreds of data items, and consequently is still time consuming. To accelerate the coarse-recall phase, a natural way is to group pre-trained models into clusters by measuring the similarity between pre-trained models, so that the proxy score only needs to be computed for the representative model in a cluster. It reduces the time complexity of coarse-recall phase from $O(|M|)$ to $O(|MC|)$.

The models' similarity measures how two pre-trained models tend to have similar training performance on a target dataset. The training performance could be related to various factors, such as training data domain and quality, model architecture, and parameter size, etc. As these factors are heterogeneous and could not always be available, it maybe unfeasible to combine these factors explicitly to compute the model similarity. In our work, we propose to measure the similarity between models through a data-driven way motivated by the phenomena that models having similar training performances on benchmark datasets also tend to have similar training performance on a new task.

Fig. 2 (a) and (b) illustrate the model clustering process. Based on the performance matrix $Matrix(D, M)$, each model m_j could be represented as a $|D|$-dimensional vector $vec(m_j) = (p(d_1|m_j), p(d_2|m_j), ..., p(d_m|m_j))$. And model clustering could be conducted by any clustering algorithm based on the models' distance $dis(m_{j1}, m_{j2})$ measured based on $vec(m_{j1})$ and $vec(m_{j2})$. As the benchmark datasets cover a group of representative tasks for a machine learning application, the training performances on such datasets could measure both the feature extraction capability and domain characteristics of a pre-trained model. Therefore, for a target task T, it is possible that there are benchmark tasks which share similar feature extraction or domain of the training dataset. So the models, which are similar measured by $vec(m_{j1})$ and $vec(m_{j2})$ on benchmark datasets, are also tend to have similar performance in the new task T. Here, we measure the model similarity through the average accuracy differences on k benchmark datasets where two models have maximum accuracy differences : R6.O2

$$sim(m_{j1}, m_{j2}) = 1 - avg(top_k|vec[m_{j1}] - vec[m_{j2}]|) \quad (1)$$

For the performance matrix $Matrix(D, M)$, although there are $m \cdot n$ elements which need $m \cdot n$ training, it is not necessary to train a pre-trained model on every benchmark dataset , since only top accuracy differences will be used to measure the model similarity. Actually, the training performance on a subset of training data with relative small size could be enough.

Based on the above model similarity measurement, we can adopt state-of-the-art clustering algorithms to group models in model repository, such K-means [12], hierarchical clustering [13], etc. After model clustering, for a cluster C_i, the model belongs to C_i and has the maximum average training performance on benchmark datasets is selected as the representative model, denoted as $m(C_i)$. Then, the proxy score between the target dataset $d(T)$ and model cluster C_i could be calculated as $proxy\_score(T|m(C_i))$ which could avoid online computing the proxy score for all the models on $d(T)$ in the coarse-recall phase.

B. Model Recall

Based on the training performance matrix and model clustering result, a $recall\_score$ is computed as Eq. 2, where $acc(m_j)$ denotes the average accuracy of m_j on benchmark datasets D. We can see the $recall\_score(T|m_j)$ contains two parts. The $acc(m_j)$ counts the prior capacity for a model m_j to any new task, and the $proxy\_score(T|m_j)$ represents how the model m_j match the specific task T. In this paper, we adopt LEEP [10] to compute the $proxy\_score$ and normalize score between [0, 1]. Combining these two scores, we can sort the models in descedent order, and reserve top K models as the result of the coarse-recall phase.

$$recall\_score(T|m_j) = acc(m_j) \cdot proxy\_score(T|m_j) \quad (2)$$

As introduced in previous section, to speed up the computation of coarse-recall phase, we only compute the $proxy\_score$ between the target dataset and the representative model of a cluster, therefore, $proxy\_score(T|m_i)$ could be rewritten as $proxy\_score(T|m(c(m_j))$ where $c(m_j)$ denotes the cluster of model m_j belonging to. Meanwhile, as there may be a number of singleton model clusters ($|Ci| = 1$) after model clustering, the $proxy\_score$ is only computed between the target target T and non-singleton clusters for efficiency consideration. Therefore, for models in non-singleton clusters, the $recall\_score$ is computed as:

$$\begin{aligned} recall\_score(T|m_j) = acc(m_j) \cdot \\ proxy\_score(T|m(c(m_j))) \ for \ |c(m_j)| > 1 \end{aligned} \quad (3)$$

As we do not compute the $proxy\_score$ directly for singleton model clusters, the $recall\_score$ for models in singleton clusters is computed as Eq. 4 where the $proxy\_score$ is propagated from the representative models of non-singleton clusters (denoted as C_{non}) and decayed by the model similarity $sim(m_j, m(C_k))$:

$$\begin{aligned} recall\_score(T|m_i) = acc(m_i) \cdot \frac{1}{|C_{non}|} \cdot \\ \sum_{k=1}^{|C_{non}|} (sim(m_j, m(C_k)) \cdot proxy\_score(T|m(C_k))) \end{aligned} \quad (4)$$

Combining Eq. 3 and Eq. 4, we can compute the $recall\_score$ for all the models in M and return the top K models to the fine-selection phase.

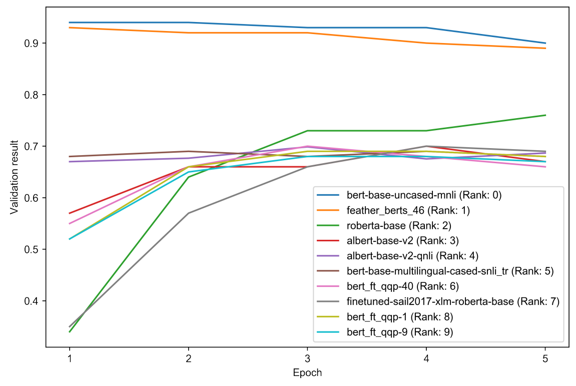


Fig. 3. Top-10 models validation and test results on MNLI dataset. Model names ignore the repository name they belong to, the full names can be found in Table VIII in Appendix B.

IV. Fine-Selection

After the initial coarse recall phase, we reduced the number of candidate pre-trained models from $O|M|$ to $O|MC|$. Next, based on successive halving, we utilize the convergence information from fine-tuning the models on benchmark datasets to select a good model more quickly and accurately.

A. Early Stopping

Fine-tuning models is a time-consuming process. Key to improve efficiency is the ability to filter out poorly-performing models at an earlier training steps. Therefore, we're interested in understanding whether there is a strong and prevalent correlation between the initial validation performance and the final test performance during the fine-tuning process of pre-trained models. This correlation could potentially allow us to filter out more models at an earlier stage of training. For each target dataset, we plot the validation performance changes during the fine-tuning process for models that pass the initial screening. Fig. 3 illustrates the performance changes of 10 models screened for the MNLI dataset. It can be observed that the models performing well on the test set also exhibit better validation performance in the early stages of training, and it seems only two models out of the total ten models achieve much higher performance at the first training epoch. Therefore, we do not need to fine-tune all models to the point of convergence. Instead, we can filter out under-performing models at an earlier stage, leading to a more efficient model selection process. Fig. 8 in Appendix A provides model's performances under another set of hyperparameters, showing the sensitivity of the training process to hyperparameters and the robustness of our method.

B. Successive Halving

Based on the observation that models which perform well in early training iterations are likely to maintain superior performance when trained to full convergence, successive halving is the state-of-the-art method applied to speed up the model selection process [2] [3]. The successive halving algorithm operates iteratively in stages, and halves the pool of considered models at each stage. Specifically, each model is trained for a fixed number of iterations at each stage and then its performance is validated. Models with poor performance are discarded, while the better-performing ones proceed to the next round of more intensive training. This process continues until the top models are identified. Assuming that the initial model number is $|M|$ and the training step during each halving is s batches, the total budget for identifying the highest performance model would typically be approximately $|M| \cdot s \cdot log_2(|M|)$ steps.

C. Fine-Selection Algorithm

While successive halving ensures that computational resources are largely devoted to the most promising models, the practice of only filtering out half of the models in each round limits the further improvement of the selection efficiency. Therefore, we propose our refinement method called 'fine-selection', which, built on successive halving, further leverages the fine-tuning information of the model on benchmark datasets. After observing the fine-tuning performance of the model on various datasets, we found that the performance changes of the same model across different datasets can be categorized into distinct clusters. As illustrated in Fig. 4, the fine-tuning performance of the BERT_base model on some benchmark datasets can be divided into four groups. Similar phenomena were observed across different models. Therefore, we propose to mining the convergence trend of models from the fine-tuning performance on benchmark datasets to predict the model's final performance on the target dataset. R6.O4

For a target dataset $d(T)$ and a given pre-trained model m_j, after training m_j on $d(T)$ for every s steps, we can compute the validation accuracy $val(T|m_j)_t$ at stage t and predict the final training performance as follows. Firstly, we generate a group of convergence trends for m_j, denoted as $\{CT(m_j)_t\}$. Specifically, we cluster the benchmark datasets into c clusters based on the validate accuracy of m on these datasets. Then, a convergence trend $CT(m_j)_t[x] = (\overline{val_x}, \overline{test_x})$ is computed as $\overline{val_x}$ and $\overline{test_x}$ are the average validate and test accuracy of m_j on the datasets in cluster x respectively. Then, based on the generated convergence trends $CT(m_j)_t$, we can assign the best matched convergence trend for m_j trained on $d(T)$ after s steps as Eq. 5. The final training performance could be predicted as the test accuracy of the matched convergence trend as in Eq. 6.

$$matched(val(T|m_j)_t) = \arg\min_x(|\overline{val_x} - val(T|m_j)_t|) \tag{5}$$

$$pred(T|m_j)_t = CT(m_j)_t[matched(val(T|m_j)_t)] \tag{6}$$

Based on convergence trend mining, Algorithm 1 describes the proposed fine-selection algorithm. Like successive halving, when each remaining model has been fine-tuned for s steps,

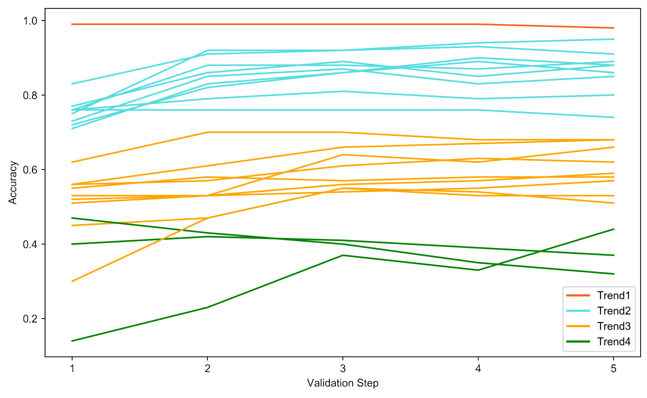


Fig. 4. Validation&Test performance of the DoyyingFace/bert-asian-hate-tweets-asian-unclean-freeze-4 model on 30 datasets.

if the number of remaining models is not 1, fine-selection is performed. The specific process is as follows:

- Obtaining: Obtain the model's every stage validation and final test results on all benchmark datasets.
- Convergence Trend Mining and Match: Perform clustering on validation results of current stage, and get the matched convergence trend by Eq. 5.
- Predict: Use the mean final test performance of the matched convergence trend as the predicted result by Eq. 6.
- Fine-Filter: Among the remaining models, starting from the model with the worst validation performance, if there exists a model with better validation performance and whose predicted performance is also better by a certain threshold, we remove this model.
- Halving: If the number of remaining models is more than half of the number of models at the beginning of the selection, we directly eliminate the model with the worst validation result until the number is reduced by half.

In summary, our fine-selection method ultimately yields a single model fully trained on the target dataset. It ensures that at least half of the models are filtered out at each step, thereby resulting in a selection efficiency significantly higher than that of successive halving.

Algorithm 1: Fine-Selection

```
Input: Recalled models $M_0$;
       Validation and Test result of $M$ on $D$, $val$ and $test$;
       Total training steps $T$, validation interval $s$
Output: Final Trained Model
1:  for $t = 0$ to $\lfloor T/s \rfloor - 1$ do
2:      $M' \leftarrow$ Train each model in $M_t$
3:      if $|M'| \neq 1$ then
4:          $v_j \leftarrow$ Validate each model
5:          $x_j \leftarrow$ Match convergence trends by Eq. 5
6:          $pred_{j]} \leftarrow$ Preddict final performance by Eq. 6
7:          $M' \leftarrow$ Remove models with lower $v$ and $pred$,
            $pred$'s difference should larger than the threshold
8:          while $|M'| > \lfloor |M_t|/2 \rfloor$ do
9:              $M' \leftarrow$ Remove models in $M'$ with lowest v
10:         end while
11:     end if
12:     $M_{t+1} \leftarrow M'$
13: end for
14: return $M_{T/s}[0]$
```

V. Experiments

In this section, we study the model selection experiment results on natural language process and computer vision tasks, and demonstrate the effectiveness and efficiency of the proposed methods.

A. Experiment Setup

Pre-trained Models. The models are collected from HuggingFace [1]. For natural language tasks, we select 40 models as model repositories, which both contain the state-of-the-art pre-trained models (such as BERT [8], Roberta [14], etc) and the models fine-tuned on some downstream tasks. For computer vision tasks, we select 30 models as repositories. The structure of the models includes ViT [15], BEiT [16], DEiT [17], poolformer [18], dinat [19], and Visual Attention Network [20]. Similar to NLP models, these models contain state-of-the-art models and their fine-tuned versions. A complete list of models are given in Appendix A.

Datasets. The experiments are conducted on NLP and CV datasets which are divided into benchmark datasets and target datasets. The benchmark datasets are used to construct the performance matrix for model clustering. The target datasets are used to evaluate the proposed two-phase model selection method. The datasets outputs are in the form of classification and are totally different in two parts, which means they are variant in the sub-tasks, label numbers, and label distributions. It is worth noting that our datasets contain both common datasets like GLUE [4] and cifar10 [21] as well as domain-specific datasets such as finance and medical science. These datasets are available on HuggingFace and have been split into training and testing sets by their contributors. The datasets used for performance matrix construction and method evaluation are given below: R6.O5 R6.O8 R6.O7 R6.O6

- Benchmark Datasets: The benchmark datasets for NLP are mainly from GLUE [4] and SuperGLUE [22]. They are COLA, MRPC, QNLI, QQP, RTE, SST2, STSB and WNLI in GLUE, and CB, COPA, WIC in SuperGLUE. And we use some domain-specific tasks popular in HuggingFace, which are: imdb [23], yelp_review_full [24], yahoo_answer_topic, dbpedia_14 [24], xnli [25], anli [26], app_reviews [27], trec [28], [29], sick [30] and financial_phrasebank [31]. For CV benchmark, we use datasets of image classification in different domains: food101 [32], CC6204-Hackaton-Cub-Dataset [5], cats_vs_dogs [33], cifar10 [21] and MNIST [34].

- Target Datasets: For NLP evaluation, four tasks are selected: tweet_eval [35] collected from Tweeter reviews, MNLI [36] from the GLUE benchmark, MultiRC [37], and Boolq [38] from superGLUE benchmark. For CV evaluation, four image classification datasets with different domains are selected: chest-xray-classification [39], MedMNIST [40], oxford-flowers [41], and beans [42].

R6.O6 R6.O7 R6.O8 We give further description of datasets in Appendix C.

Performance Matrix. We build a performance matrix by fine-tuning all the pre-trained models on the benchmark datasets, which contains 40×24 trains for natural language processing and 30×10 trains for computer vision respectively. Each training is conducted with 5 epochs and 4 epochs for natural language processing and computer vision respectively, which is enough to compare the relative accuracy between different trains and could support convergence trend mining for the fine-selection phase.

TABLE I
CLUSTERING METHODS COMPARISON

Model Similarity	Hierarchical cluster		K-means	
	NLP	CV	NLP	CV
performance-based	**0.505**	**0.806**	0.466	0.702
text-based	0.476	0.696	0.453	0.732

B. Experiment for Coarse-Recall Phase

For coarse-recall phase, we first study the experiment results for model clustering and then study the results for model recall.

R2.O4 R6.O2 **Model Clustering.** We study the model clustering results by comparing different model similarity measurements and clustering algorithms, and the clustering results are measured in terms of silhouette coefficient [43]. For model similarity measurement comparison, the performance-based similarity is calculated by Eq. 1 and we conduct an experiment in Appendix. D to determine the parameter k. The text-based similarity is calculated from the text of the corresponding model card (Appendix E. shows an example of a model card). We adopt SBERT [44] to encode the text into an vector so that cosine similarity could be computed. For the clustering algorithms, we compare two state-of-the-art clustering algorithms, K-means and hierarchical clustering.

Table I shows the comparison results. We can see the performance-based similarity achieves higher silhouette coefficient compared to text-based similarity, demonstrating the former could generate a better clustering structure. For clustering algorithm comparison, the hierarchical clustering outperforms the K-means clustering in a clear gap based on performance-based similarity, showing that clusters generated by hierarchical clustering have smaller similarity between and more connection within. To take a step further, it is worth noting that clusters generated by hierarchical clustering are more reasonable than those generated by K-means clustering as discussed below. Therefore, we will conduct the following experiments based on the results of hierarchical clustering.

Table II illustrates our clustering results of non-singleton clusters for natural language processing and computer vision tasks. For natural language processing models, there are 7 non-singleton clusters which contain 30 models out of a total of 40 models. For computer vision tasks, there are 6 non-singleton models that contain almost all the models. We study the details for some clusters. For natural language process models, as the introduction information is not available for many models, we infer the model training process from the model name. For C_1 and C_2 clusters, we can see they mainly contain pre-trained models fine-tuned on qqp [45] and cola [46] datasets respectively, demonstrating the models fine-tuned on the same downstream tasks could be grouped together by model clustering. On the other hand, we can see there are also models with names containing qqp that are not clustered into C_1, such as C_6, demonstrating the performance of models with similar model names may also vary, which may be caused by different training setups. For C_3 cluster, we find that it groups models with names containing mnli and feather_berts together, and it is reasonable since we can find from [47] that the feather_berts models are also BERT models fine-tuned on the MNLI datasets. This also demonstrates the effectiveness of the model clustering. The results of computer vision clusters exhibit a similar phenomenon. The C_1 cluster mainly contains base-size deit models [17], small-size vit models using dino [48], and vit models using msn [49]. Looking further into the models, we discover that these three kinds of models are all pre-trained or fine-tuned on dataset Imagenet-1k [50]. The C_3 cluster mainly contains base-size vit models using dino, vit base models, and beit base models. Similar to C_1 cluster, these models, except dino-vit, are all pre-trained or fine-tuned with dataset Imagenet-21k [51]. On the other hand, we can also see models that are not grouped in one cluster even though they share similar names or training datasets, like dino vit models in C_1 and C_3, which also demonstrates models with similar model names may have different performance. We put the result of the K-means clustering method in Appendix F. Generally speaking, some clusters contain models of different structures and/or training datasets. R2.O4

We also study the performance of models in non-singleton clusters in Table III. We can see the average accuracy of models in non-singleton clusters is significantly higher than that of models in singleton clusters for both natural language and computer vision tasks. Meanwhile, we also compute the count of models that achieve maximum accuracy for a benchmark dataset, and it exhibits a similar result that models in non-singleton clusters almost contribute to all the best models for benchmark datasets. The result of Table II demonstrates that models in the non-singleton clusters tend to achieve higher training performance. This could be explained by the fact that the high-quality model may achieve similar performance bounded by the state-of-the-art model, while on the other hand, the performance of poorly-performed models may differ a lot on different datasets. This phenomenon supports the $proxy\_score$ computation strategy in the coarse-recall phase that we only compute the $proxy\_score$ between the target

TABLE II
MODEL CLUSTERING RESULTS

Model Clusters of Natural Language Processing		
Cluster	**Size**	**Pre-trained Models**
C_1	5	Jeevesh8/bert_ft_qqp-68, Jeevesh8/bert_ft_qqp-9, Jeevesh8/bert_ft_qqp-40, connectivity/bert_ft_qqp-1, connectivity/bert_ft_qqp-7
C_2	7	Jeevesh8/512seq_len_6ep_bert_ft_cola-91, anirudh21/bert-base-uncased-finetuned-qnli, Jeevesh8/bert_ft_cola-88, manueltonneau/bert-twitter-en-is-hired, bert-base-uncased, aditeyabaral/finetuned-sail2017-xlm-roberta-base, DoyyingFace/bert-asian-hate-tweets-asian-unclean-freeze-4
C_3	5	Jeevesh8/feather_berts_46, ishan/bert-base-uncased-mnli roberta-base, Alireza1044/albert-base-v2-qnli, albert-base-v2
C_4	2	CAMeL-Lab/bert-base-arabic-camelbert-mix-did-nadi, aliosm/sha3bor-metre-detector-arabertv2-base
C_4	2	Splend1dchan/bert-base-uncased-slue-goldtrascription-e3-lr1e-4, aychang/bert-base-cased-trec-coarse
C_5	3	aviator-neural–bert-base-uncased-sst2, distilbert-base-uncased, 18811449050–bert_finetuning_test
C_6	4	Jeevesh8/init_bert_ft_qqp-33, Jeevesh8/init_bert_ft_qqp-24, connectivity/bert_ft_qqp-17, connectivity/bert_ft_qqp-96
C_7	2	XSY/albert-base-v2-imdb-calssification, emrecan/bert-base-multilingual-cased-snli_tr
Model Clusters of Computer Vision		
Cluster	**Size**	**Pre-trained Models**
C_1	6	facebook/deit-base-patch16-224, facebook/deit-base-patch16-384, facebook/dino-vits16, facebook/vit-msn-base, facebook/vit-msn-small, Visual-Attention-Network/van-large
C_2	2	facebook/deit-small-patch16-224, Visual-Attention-Network/van-base
C_3	11	facebook/dino-vitb16, facebook/dino-vitb8, google/vit-base-patch16-224, google/vit-base-patch16-384, lixiqi/beit-base-patch16-224-pt22k-ft22k-finetuned-FER2013-6e-05, lixiqi/beit-base-patch16-224-pt22k-ft22k-finetuned-FER2013-7e-05, lixiqi/beit-base-patch16-224-pt22k-ft22k-finetuned-FER-5e-05-3, microsoft/beit-base-patch16-224, microsoft/beit-base-patch16-224-pt22k-ft22k, microsoft/beit-base-patch16-384, nateraw/vit-age-classifier
C_4	2	shi-labs/dinat-large-in22k-in1k-224, shi-labs/dinat-large-in22k-in1k-384
C_5	2	sail/poolformer-m36, sail/poolformer-m48
C_6	2	shi-labs/dinat-base-in1k-224, microsoft/beit-large-patch16-224-pt22k

dataset and the representative models of non-singleton clusters, and propagate the score to models in singleton clusters.

TABLE III
PERFORMANCE COMPARISON OF MODELS IN SINGLETON AND NON-SINGLETON CLUSTERS

Task Type	**Cluster Type**	**Avg(Acc)**	**No. Maximum(Acc)**
NLP	Non-Singleton	0.67	22
	Singleton	0.61	2
CV	Non-Singleton	0.84	10
	Singleton	0.73	0

Model Recall. To evaluate the effectiveness of coarse-recall phase, we fine-tuning all the models on corresponding target datasets to get the actual training performance. Then, we compare the average training accuracy on the target datasets of top K recalled models. Fig. IV compares the average accuracy between coarse-recall and random-recall. Here, random-recall represents randomly return K models from model repository. We can see coarse-recall achieves higher accuracy compared with random-recall on all the four target datasets. Meanwhile, for coarse-recall, the average accuracy of smaller K values is bigger than that of bigger K values, demonstrating the top models recalled by coarse-recall tend to achieve higher training performance compared with models with lower recall score. Meanwhile, we also find that the top 5 models recalled by coarse_recall has contained the model achieving maximum training performance; and for tweet, beans and MedMNIST datasets, the number of recalled models to contain the best model is 10, 10, 15 respectively. In the following experiments, we empirically set the number of recalled models as 15 and 10 for natural language processing and computer vision tasks respectively, which accounts for about 30% of corresponding total models.

C. Experiments for Fine-Selection Phase

Convergence Trend. In this section, we demonstrate the effectiveness of mined convergence trend based on a model's validation performance on benchmark datasets. We apply clustering methods during the fine-selection phase, and usually, a single filtering is sufficient to select the final model. Therefore, we only consider the performance of clustering based on the first validation results. Firstly, we directly measure the effect of clustering through the silhouette coefficient [43], where a higher value indicates a better clustering outcome. As shown in the blue part of Fig. 6, across all models, clustering based on validation significantly outperforms random clustering, demonstrating the effectiveness of validation results for clustering. In addition, we consider each benchmark dataset as the target dataset, assess the feasibility of predicting the final test performance using the mean test performance of models within the same cluster as the current model belongs to. We compare this with predicting the test performance based on the mean of all benchmark dataset test performances. The final metric is the absolute difference between the predicted and actual test performance divided by the actual test performance, averaged across all datasets. The red part of Fig. 6 demonstrates that

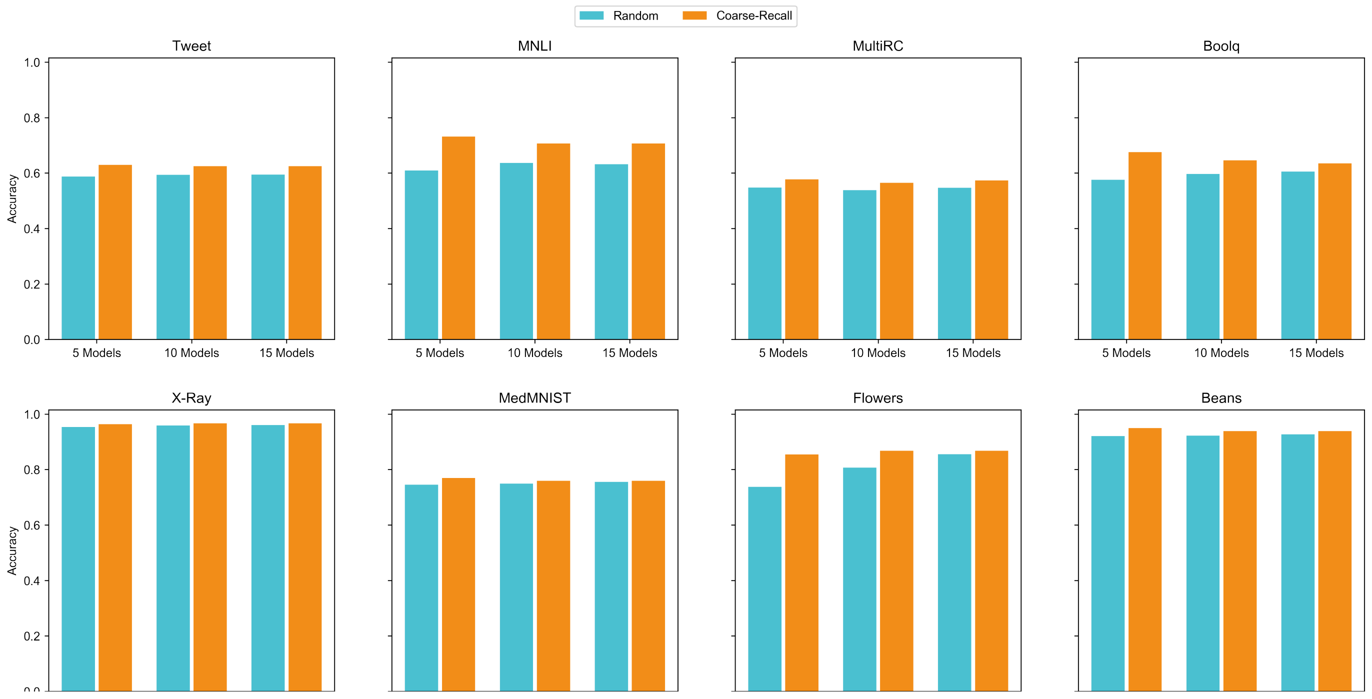


Fig. 5. The average accuracy comparison of recalled models

our method predicts final test performance more accurately. This not only validates the feasibility of using the mean test performance within the cluster as a prediction but also demonstrates the effectiveness of clustering. In summary, it is feasible to select models by mining convergence trend based on validation performance and further predicting the model's final test performance.

R8.O2 **Filtering Threshold** Given the validation fluctuations during the model training process and the potential significant discrepancies between benchmark and target datasets, utilizing convergence trends to filter models might eliminate models with good performance. Therefore, we propose introducing a threshold, stipulating that a model is only filtered out when there is another model with better validation and a predicted performance improvement exceeding this threshold. The threshold is a proportion of the difference between the predicted performances. Table IV illustrates the performance of fine-tuning methods under various threshold conditions. It can be observed that the threshold ensures better-performing models are filtered out later, albeit at the expense of efficiency. In order to more intuitively represent the efficiency of our method and the performance of the selected models, we uniformly use a 0% threshold in subsequent experiments.

Method Comparison. After validating the effectiveness of convergence trend mining from the model training performance on benchmark datasets, we further verify its value in model selection methods.

1) Performance: Fig. 7 compares the final performances of the last model selected by the successive halving and our proposed fine-selection method among the top 10 performing

TABLE IV
ACCURACY AND TIME COMPARISON AMONG DIFFERENT FILTERING THRESHOLD SETTINGS IN FINE-SELECTION. 0% IS THE ORIGINAL SETTING.

Models		0%	1%	5%	10%
MNLI	Accuracy	0.85	0.85	0.85	0.85
	RunTime	14	14	15	16
MultiRC	Accuracy	0.63	0.63	0.63	0.63
	RunTime	16	16	19	19
Flowers	Accuracy	0.985	0.985	0.986	0.986
	RunTime	15	15	18	18
X-Ray	Accuracy	0.966	0.966	0.969	0.969
	RunTime	14	15	18	18

and all of the models. At the same time, we provide the best and worst performances among the top 10 models for NLP and CV tasks. We can find that the fine-selection (FS) method is always able to pick out the optimal or near-optimal model. However, the traditional successive halving method may not necessarily select the best model.

2) Time: In addition to the performance of the selected model, the speed of model selection is equally important. In Table VI, we compare the speed improvement of the two methods based on brute-force search (BS), that is, fine-tuning all models for a fixed number of epochs. Given the consistency of training settings and hardware environments, we use the total number of fine-tuning epochs across all models to represent the time for model selection. We can observe that our method has a noticeable efficiency improvement compared to successive halving in all datasets and different model numbers. Furthermore, it's worth noting that the time taken per epoch is considerable, thus the time saved in model selection by our

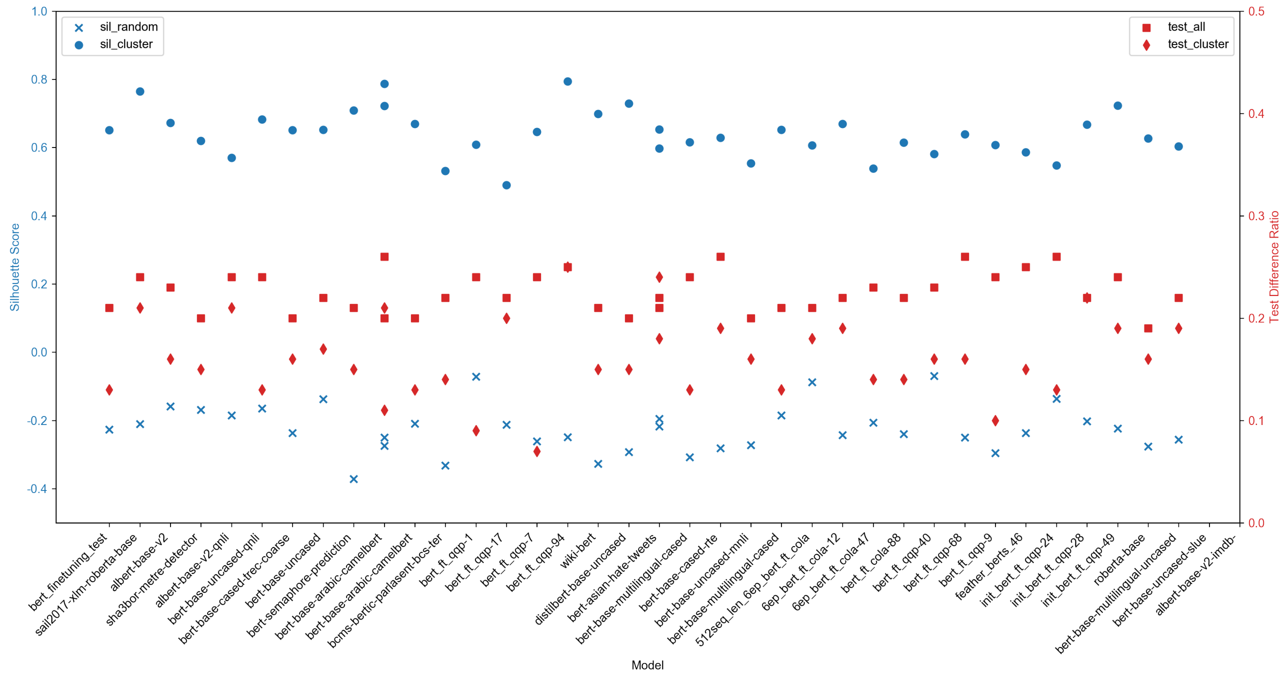


Fig. 6. Clustering Performance based on the first validation results. The blue color represents the comparison between random clustering and clustering based on validation performance, with silhouette score as the selected metric. The higher the silhouette score, the better the clustering effect. The red color represents the comparison between the prediction of test performance using the clustering method and the mean of all historical test performances. The metric used here is the absolute difference between the predicted and actual test performance divided by the actual test performance. We calculate the average with each dataset as the target dataset; the smaller the value, the more accurate the prediction performance. Full model name could be found in Table VIII in Appendix.B.

TABLE V
TIME COMPARISONS AMONG DIFFERENT MODEL SELECTION METHODS. NLP TASKS ARE TRAINED FOR 5 EPOCHS AND CV TASKS ARE TRAINED FOR 4 EPOCHS. RUNTIME IS TOTAL TRAINING EPOCH NUMBER. SELECTIONS OF TOP 10 MODELS AND ALL OF THE MODELS, WHOSE NUMBER IS 40 AND 30 FOR NLP AND CV, IS BOTH PROVIDED.

Models		10		40(NLP)/30(CV)	
		Runtime (epoch)	Speedup (vs. BF)	Runtime (epoch)	Speedup (vs. BF)
NLP	BF	50		200	
	SH	19	2.63x	77	2.60x
Tweet	FS	14	3.57x	44	4.55x
MNLI	FS	14	3.57x	44	4.55x
MultiRC	FS	15	3.33x	46	4.35x
Boolq	FS	16	3.13x	48	4.17x
CV	BF	40		120	
	SH	18	2.22x	55	2.18x
X-Ray	FS	13	3.08x	38	3.16x
MedMNIST	FS	15	2.67x	37	3.24x
Flowers	FS	15	2.67x	36	3.33x
Beans	FS	17	2.35x	41	2.93x

method is significant.

3) Scaling to more models: In addition to the 10 models obtained based on coarse-recall, we further explored the performance of our method when the number of models increased. In Fig. 7 and Table III, we provided performance and time comparisons for 40 models of natural language process and 30 models of computer vision. We found that as the number of models increases, our method not only maintains the quality of model selection but also significantly saves time. This indicates that the fine-selection method is not only suitable for fine-grained selection after coarse-grained model recall, but it is also effective for a larger number of models. Therefore, our method can be expanded to accommodate more models, making it capable of coping with the increasing emergence of pre-trained models.

D. Overall Performance

We study the end-to-end model selection effectiveness and efficiency in this section. The comparison methods are also brute-force search (BF) and successive halving (SH) in last section. The number of total models are 40 and 30 for NLP and CV, and recalled model is 10. Table VI shows accuracy and efficiency comparison among different methods where CR+FS stands for the two-phase framework (coarse-recall and fine-selection) in this paper. We can see both SH and two-phase model selection methods proposed in this paper both achieve near accuracy compared with BF. Table VI also exhibits the time consumed for the three methods in terms of training epochs. For CR+FS, as $proxy\_score$ needs to be computed in coarse-recall phase which needs to do inference on the target task, we count the computation time as $0.5 \cdot |MC|$ epochs because the inference do not need to compute the gradients to do back-propagation. From Table VI, We can see the two-phase model selection methods achieve about 2x to 3x times faster compared with SH and about 5x to 8x times faster compared with BF. The speed up comes

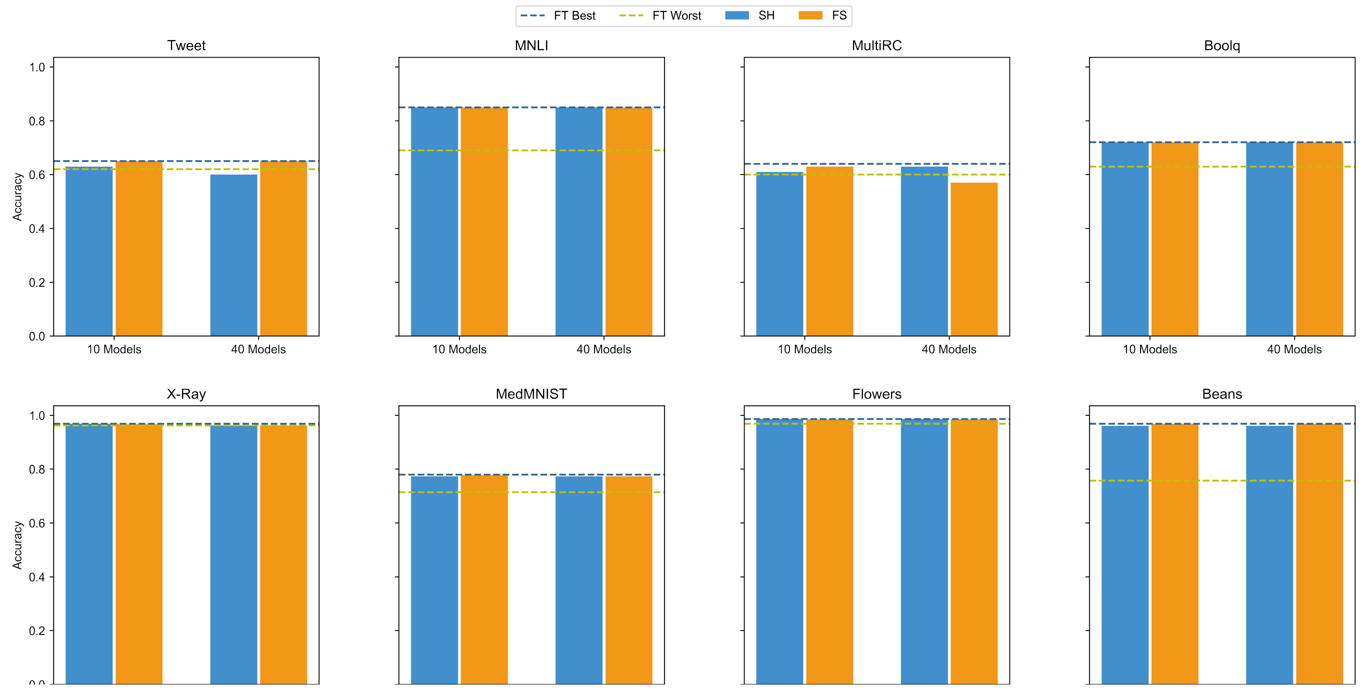


Fig. 7. Selected model's performance comparison between successive halving(SH) and our fine-selection(FS) method on NLP datasets: MNLI, Tweet, Boolq, MultiRC and CV dataset: Beans, MedMNIST, Flowers and X-ray. 10 models and 30 models represent initial model number for filtering. The best and worst model performances in the top 10 models for each dataset are also provided.

TABLE VI
END-TO-END COMPARISONS ON TIME AND PERFORMANCE AMONG DIFFERENT MODEL SELECTION METHODS. 2PH, BF, SH IS SHORT FOR 2 PHASE (COARSE RECALL +FINE SELECTION), BRUTE FORCING, SECCESSIVE HALVING, RESPECTIVELY. ACC IS ACCURACY METRIC.

	Runtime	Speedup		Acc		
	2PH	(vs.BF)	(vs.SH)	BF	SH	2PH
Tweet	19	10.53x	4.05x	0.650	0.60	0.650
MNLI	19	10.53x	4.05x	0.850	0.850	0.850
MultiRC	20	10.00x	3.85x	0.640	0.630	0.630
Boolq	21	9.52x	3.67x	0.720	0.720	0.720
X-Ray	18	6.67x	3.06x	0.969	0.962	0.962
MedMNIST	20	6.00x	2.75x	0.779	0.773	0.773
Flowers	20	6.00x	2.75x	0.986	0.986	0.985
Beans	22	5.45x	2.50x	0.968	0.961	0.961

TABLE VII
CASE STUDY OF THE FINAL SELECTED MODEL AFTER COARSE-RECALL AND FINE-SELECTION ON FOUR DATASETS. ACC AND R ARE SHORT FOR ACCURACY AND RANK. THE RANK FOR CR ARE OBTAINED BY SORTING THE RECALLED MODELS BY PROXY SCORE. AVG_ACC IS OBTAINED BY COMPUTING THE AVERAGE ACCURACIES OF THE RECALLED MODELS.

Dataset	Best_model	Acc	R@CR	Avg_Acc
MultiRC	albert-base-v2	**0.630**	5	0.574
Boolq	bert-base-uncased-mnli	**0.720**	0	0.635
MedMNIST	vit-base-patch16-384	**0.773**	1	0.768
Flowers	vit-base-patch16-224	**0.985**	9	0.891

from both phases where the coarse-recall phase largely reduce the number of models need to be fine-tuned and the fine-selection phase further reduce computation time of fine-tuning by exploiting the convergence trend. The results demonstrate the model selection method proposed in this paper could achieve high training performance with significantly lower compute time, hence are more suitable to address large scale model repository.

E. Generalization Study

The generalization capability of the proposed method for new target tasks lies in three aspects. Firstly, we can address new tasks with different domain distribution and task types compared with benchmark datasets. This is because the benchmark datasets are only used to measure the model similarity offline and will be used neither in corase-recall nor fine-selection phase. Secondly, the LEEP score in the coarse-recall phase proves to be able to measure the transferability between heterogeneous tasks. And thirdly, the fine-selection phase selects model directly based on the fine-tuning results at different validation intervals, which is more accurate to evaluate the final transferability of a source model. R2.O1 R8.O2

Table VII illustrates the best selected models for four target tasks which have different domain distribution and task type compared with benchmark datasets. The best models are ranked higher at coarse-recall phase and the accuracy are all higher than the average accuracy of all models. The Boolq is a question answering task for yes/no questions based on given passages, and the best model selected for Boolq is bert-base-uncased-mnli which is the bert model fine-tuned on the MNLI dataset. As the input format and output label space are both different for Boolq and MNLI, the result demonstrates the proposed method could capture the latent transferability between heterogeneous tasks. On the other hand, the MultiRC dataset selects albert-base-v2 as the best model, which is a pre-trained model not fine-tuned on any downstream dataset. For computer vision task, both Flowers and MedMNIST datasets select the vit-base-patch16 models which are trained on data with different domain distribution compared with corresponding tasks, demonstrating the out-of-domain capability of proposed method in CV tasks.

VI. Related Work

Pre-training could be viewed as an application of deep transfer leanring [52], where the a model is pre-trained on a upstreaming dataset and then the pre-trained model will be used as parameter initialization and continuing trained on datasets of various downstream tasks. The upstream task could be unsupervised task, such as masked language model [8] [14] for natural language processing, or supervised task, such as image classification for computer vision [53]. Pre-trained models could help the downstream task to achieve better training effect especially for the situation where the training data of the downstream task is limited. Meanwhile, compared with training dataset, the pre-trained model are usually more safely to be published. Therefore, thousands of pre-trained models has been published on model hub website such as HuggingFace [1] , PyTorch-Hub [54], etc. On the other hand, previous work has demonstrated that for the same downstream task, the training performance may vary a lot when fine-tuned on different pre-trained models [2]. Hence, selecting a good pre-trained model from model repository is an important step to achieve high training performance for the target task.

As the number of models in model repository becomes larger, it is compute infeasible to select a good model after fine-tuning all the models on the target task, and a few methods has been proposed to speed up the model selection process. We divide the proposed model selection methods into two categories, light-weight proxy score computation and model selection during fine-tuning. Specifically, for methods of light-weight proxy score computation, Task2Vec [55] embedding the upstream and downstream tasks into the same vector space, and models trained in upstream task closed to the downstream task could be selected directly; LEEP [10] is proposed to measure the transfer-ability for a pre-trained model on the target classification task; and in [11], the KNN classifier is built on the hidden layer output of pre-trained models to approximate the training effect after fine-tuning. For methods of model selection during fine-tuning, Palette [2] adopts successive halving [54] to filter lower effect models at early training step, and Shift [3] builds cost model to predict the training cost of successive halving and fine-tuning directly. The two-phase model selection framework could be viewed as combing the advantage of the two category methods. As the two-phase model selection framework is flexible, other model selection methods could be combined in this framework even in corresponding phase, such as we can also measure the model performance on benchmark datasets as [56] and combine this strategy with LEEP [10] in coarse-recall phase, and we can also combine multi-model selection methods [2] [57] [58] [59] in the fine-selection phase to achieve high ensemble performance.

In this paper, we also propose to cluster models and mine convergence trend to speed up coarse-recall and fine-selection phase. The clustering and mining process both depend on the training performance of pre-trained models on benchmark datasets, such as GLUE [4] for natural language processing. As thousands of datasets are also published on the website, such as HuggingFace datasets [1], methods such as [55] [60] could also be applied to measure the similarity between the task datasets and help to build more effective benchmark datasets which could cover a wider range of tasks for model selection.

VII. Future Work

The future work could be summarized into three aspects. Firstly, the coarse-recall phase aims to retain the high performance pre-trained models in top models. A single proxy-score measurement may be not enough to return high performance models for different machine learning tasks. We plan to combine different light-weight tasks to return a high quality subset of models more robustly. Secondly, in this paper, we build benchmark datasets empirically. As a large of number of datasets has been published, we will study data-driven methods to build benchmark datasets which could cover more types of machine tasks, and meanwhile make benchmark datasets more compact to make performance maintained more cheaply. And thirdly, as model selection is an important step in the machine learning pipeline, the capability of automatically select high performance model enable us to build the whole machine learning pipeline for new task. We plan to build data management system which stores and maintains the pre-trained models and datasets, then support automatically selecting models efficiently to help users complete the model training for new task. Meanwhile, the newly trained models will naturally be managed by this system to be reused again, which could form an close-loop of machine learning pipeline and improve the efficiency of machine learning systematically.

VIII. Conclusion

In this paper, we propose a two-phase framework for fast model selection, where the coarse-recall phase implements light-weight proxy tasks to recall a much smaller number of candidate models and fine-selection phase only fine-tunes the models from this first phase to select the best model. To speed up these two phases, we build performance matrix by fine-tuning pre-trained models on benchmark datasets, and cluster models based on performance matrix to avoid duplicated proxy-score computation in coarse-recall phase and mine convergence trend to filter poorly-perform models more earlier in fine-selection phase. The experiments results on natural language process and computer vision tasks demonstrates the methods proposed in this paper could select a good model for the new task much faster compared with baseline methods.

References

[1] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, "Huggingface's transformers: State-of-the-art natural language processing," *arXiv preprint arXiv:1910.03771*, 2019.

[2] Y. Li, Y. Shen, and L. Chen, "Palette: towards multi-source model selection and ensemble for reuse," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 2147–2152.

[3] C. Renggli, X. Yao, L. Kolar, L. Rimanic, A. Klimovic, and C. Zhang, "Shift: an efficient, flexible search engine for transfer learning," *arXiv preprint arXiv:2204.01457*, 2022.

[4] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," *arXiv preprint arXiv:1804.07461*, 2018.

[5] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The caltech-ucsd birds-200-2011 dataset," California Institute of Technology, Tech. Rep. CNS-TR-2011-001, 2011.

[6] S. Kornblith, J. Shlens, and Q. V. Le, "Do better imagenet models transfer better?" in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 2661–2671.

[7] J. Dodge, G. Ilharco, R. Schwartz, A. Farhadi, H. Hajishirzi, and N. Smith, "Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping," *arXiv preprint arXiv:2002.06305*, 2020.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[10] C. Nguyen, T. Hassner, M. Seeger, and C. Archambeau, "Leep: A new measure to evaluate transferability of learned representations," in *International Conference on Machine Learning*. PMLR, 2020, pp. 7294–7305.

[11] C. Renggli, A. S. Pinto, L. Rimanic, J. Puigcerver, C. Riquelme, C. Zhang, and M. Lučić, "Which model to transfer? finding the needle in the growing haystack," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 9205–9214.

[12] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the royal statistical society. series c (applied statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[13] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: an overview," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, no. 1, pp. 86–97, 2012.

[14] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[15] B. Wu, C. Xu, X. Dai, A. Wan, P. Zhang, Z. Yan, M. Tomizuka, J. Gonzalez, K. Keutzer, and P. Vajda, "Visual transformers: Token-based image representation and processing for computer vision," *arXiv preprint arXiv:2006.03677*, 2020.

[16] H. Bao, L. Dong, S. Piao, and F. Wei, "Beit: Bert pre-training of image transformers," *arXiv preprint arXiv:2106.08254*, 2021.

[17] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International conference on machine learning*. PMLR, 2021, pp. 10 347–10 357.

[18] W. Yu, M. Luo, P. Zhou, C. Si, Y. Zhou, X. Wang, J. Feng, and S. Yan, "Metaformer is actually what you need for vision," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 819–10 829.

[19] A. Hassani and H. Shi, "Dilated neighborhood attention transformer," *arXiv preprint arXiv:2209.15001*, 2022.

[20] M.-H. Guo, C.-Z. Lu, Z.-N. Liu, M.-M. Cheng, and S.-M. Hu, "Visual attention network," *arXiv preprint arXiv:2202.09741*, 2022.

[21] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[22] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Superglue: A stickier benchmark for general-purpose language understanding systems," *arXiv preprint arXiv:1905.00537*, 2019.

[23] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150. [Online]. Available: http://www.aclweb.org/anthology/P11-1015

[24] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, A. Moschitti, B. Pang, and W. Daelemans, Eds. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1746–1751. [Online]. Available: https://aclanthology.org/D14-1181

[25] A. Conneau, R. Rinott, G. Lample, A. Williams, S. R. Bowman, H. Schwenk, and V. Stoyanov, "Xnli: Evaluating cross-lingual sentence representations," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2018.

[26] Y. Nie, A. Williams, E. Dinan, M. Bansal, J. Weston, and D. Kiela, "Adversarial nli: A new benchmark for natural language understanding," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020.

[27] "Software applications user reviews," 2017.

[28] X. Li and D. Roth, "Learning question classifiers," in *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002. [Online]. Available: https://www.aclweb.org/anthology/C02-1150

[29] E. Hovy, L. Gerber, U. Hermjakob, C.-Y. Lin, and D. Ravichandran, "Toward semantics-based answer pinpointing," in *Proceedings of the First International Conference on Human Language Technology Research*, 2001. [Online]. Available: https://www.aclweb.org/anthology/H01-1069

[30] M. Marelli, S. Menini, M. Baroni, L. Bentivogli, R. Bernardi, and R. Zamparelli, "A SICK cure for the evaluation of compositional distributional semantic models," in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. Reykjavik, Iceland: European Language Resources Association (ELRA), May 2014, pp. 216–223. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2014/pdf/363_Paper.pdf

[31] P. Malo, A. Sinha, P. Korhonen, J. Wallenius, and P. Takala, "Good debt or bad debt: Detecting semantic orientations in economic texts," *Journal of the Association for Information Science and Technology*, vol. 65, 2014.

[32] L. Bossard, M. Guillaumin, and L. Van Gool, "Food-101 – mining discriminative components with random forests," in *European Conference on Computer Vision*, 2014.

[33] J. Elson, J. J. Douceur, J. Howell, and J. Saul, "Asirra: A captcha that exploits interest-aligned manual image categorization," in *Proceedings of 14th ACM Conference on Computer and Communications Security (CCS)*. Association for Computing Machinery, Inc., October 2007. [Online]. Available: https://www.microsoft.com/en-us/research/publication/asirra-a-captcha-that-exploits-interest-aligned-manual-image-categorization/

[34] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[35] F. Barbieri, J. Camacho-Collados, L. Neves, and L. Espinosa-Anke, "Tweeteval: Unified benchmark and comparative evaluation for tweet classification," *arXiv preprint arXiv:2010.12421*, 2020.

[36] A. Williams, N. Nangia, and S. R. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," *arXiv preprint arXiv:1704.05426*, 2017.

[37] D. Khashabi, S. Chaturvedi, M. Roth, S. Upadhyay, and D. Roth, "Looking beyond the surface: A challenge set for reading comprehension over multiple sentences," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 252–262.

[38] C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova, "Boolq: Exploring the surprising difficulty of natural yes/no questions," in *NAACL*, 2019.

[39] D. S. Kermany, M. Goldbaum, W. Cai, C. C. Valentim, H. Liang, S. L. Baxter, A. McKeown, G. Yang, X. Wu, F. Yan *et al.*, "Identifying medical diagnoses and treatable diseases by image-based deep learning," *cell*, vol. 172, no. 5, pp. 1122–1131, 2018.

[40] J. Yang, R. Shi, D. Wei, Z. Liu, L. Zhao, B. Ke, H. Pfister, and B. Ni, "Medmnist v2-a large-scale lightweight benchmark for 2d and 3d biomedical image classification," *Scientific Data*, vol. 10, no. 1, p. 41, 2023.

[41] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *2008 Sixth Indian conference on computer vision, graphics & image processing*. IEEE, 2008, pp. 722–729.

[42] M. A. Lab. (2020, January) Bean disease dataset. [Online]. Available: https://github.com/AI-Lab-Makerere/ibean/

[43] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[44] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. [Online]. Available: https://arxiv.org/abs/1908.10084

[45] L. Sharma, L. Graesser, N. Nangia, and U. Evci, "Natural language understanding with the quora question pairs dataset," *arXiv preprint arXiv:1907.01041*, 2019.

[46] A. Warstadt, A. Singh, and S. R. Bowman, "Neural network acceptability judgments," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 625–641, 2019.

[47] R. T. McCoy, J. Min, and T. Linzen, "Berts of a feather do not generalize together: Large variability in generalization across models with similar test set performance," *arXiv preprint arXiv:1911.02969*, 2019.

[48] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 9650–9660.

[49] M. Assran, M. Caron, I. Misra, P. Bojanowski, F. Bordes, P. Vincent, A. Joulin, M. Rabbat, and N. Ballas, "Masked siamese networks for label-efficient learning," in *European Conference on Computer Vision*. Springer, 2022, pp. 456–473.

[50] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, pp. 211–252, 2015.

[51] T. Ridnik, E. Ben-Baruch, A. Noy, and L. Zelnik-Manor, "Imagenet-21k pretraining for the masses," *arXiv preprint arXiv:2104.10972*, 2021.

[52] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III 27*. Springer, 2018, pp. 270–279.

[53] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[54] K. Jamieson and A. Talwalkar, "Non-stochastic best arm identification and hyperparameter optimization," in *Artificial intelligence and statistics*. PMLR, 2016, pp. 240–248.

[55] A. Achille, M. Lam, R. Tewari, A. Ravichandran, S. Maji, C. C. Fowlkes, S. Soatto, and P. Perona, "Task2vec: Task embedding for meta-learning," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6430–6439.

[56] X. Zhai, J. Puigcerver, A. Kolesnikov, P. Ruyssen, C. Riquelme, M. Lucic, J. Djolonga, A. S. Pinto, M. Neumann, A. Dosovitskiy *et al.*, "The visual task adaptation benchmark," 2019.

[57] J.-W. Cui, W. Lu, X. Zhao, and X.-Y. Du, "Efficient model store and reuse in an olml database system," *Journal of Computer Science and Technology*, vol. 36, pp. 792–805, 2021.

[58] W. Zhang, J. Jiang, Y. Shao, and B. Cui, "Efficient diversity-driven ensemble for deep neural networks," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 2020, pp. 73–84.

[59] Y.-X. Ding and Z.-H. Zhou, "Boosting-based reliable model reuse," in *Asian Conference on Machine Learning*. PMLR, 2020, pp. 145–160.

[60] F. Feng, Y. Yang, D. Cer, N. Arivazhagan, and W. Wang, "Language-agnostic bert sentence embedding," *arXiv preprint arXiv:2007.01852*, 2020.

APPENDIX

A. *Mnli Results*

We provide different models' validation results on MNLI dataset under a different learning rate 1e-5 in Fig. 8, compared to 3e-5 in Fig. 3. It can be observed that under new hyperparameter settings, the training dynamics of the models have changed. The performances of the top two models did not continuously decline with further training, suggesting a less severe overfitting issue. This indicates that the training process of models is highly sensitive to the setting of hyperparameters. In addition, we use our two-phase model selection method for the model training process under the new hyperparameters, and the performance and efficiency are consistent. Despite the changes in the training process, the variation in model performance was not significant enough to impact the effectiveness of our method. Therefore, our approach is robust to different hyperparameter settings in model training and is applicable across various model training scenarios.

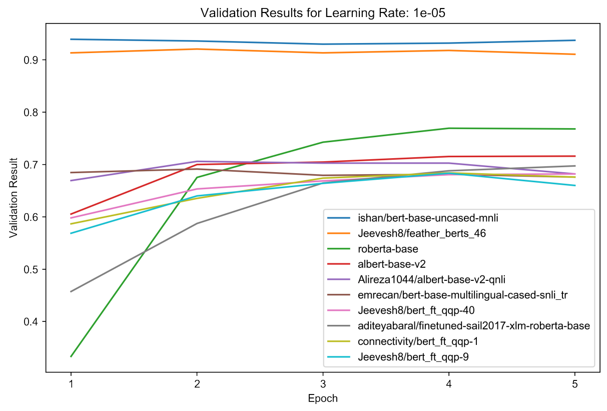


Fig. 8. Top-10 models validation and test results on MNLI dataset. Learning rate is 1e-5, which is different than 3e-5 in the Fig. 3.

B. *Model Details*

The pre-trained models we use are all from Huggingface's model hub [1]. We list the full names of all the NLP and CV models used in our work in Table VIII. Note that we sometimes use incomplete model names in the main text to save space by removing the name of the repository to which the model belongs. After removing the repository name prefix, the model names are still uniquely summarized in the list of models we use, so partial model names can also be used to pinpoint the corresponding model.

NLP models and CV models are listed in Table IX in total. All models are available using "https://huggingface.co/" as prefix.

[1]https://huggingface.co/models

TABLE VIII
NLP AND CV MODELS

NLP model name	CV model name
18811449050/bert_finetuning_test	facebook/deit-base-patch16-224
aditeyabaral/finetuned-sail2017-xlm-roberta-base	facebook/deit-base-patch16-384
albert-base-v2	facebook/deit-small-patch16-224
aliosm/sha3bor-metre-detector-arabertv2-base	facebook/dino-vitb16
Alireza1044/albert-base-v2-qnli	facebook/dino-vitb8
anirudh21/bert-base-uncased-finetuned-qnli	facebook/dino-vits16
aviator-neural/bert-base-uncased-sst2	facebook/vit-msn-base
aychang/bert-base-cased-trec-coarse	facebook/vit-msn-small
bert-base-uncased	google/vit-base-patch16-224
bondi/bert-semaphore-prediction-w4	google/vit-base-patch16-384
CAMeL-Lab/bert-base-arabic-camelbert-da-sentiment	google/vit-base-patch32-224-in21k
CAMeL-Lab–bert-base-arabic-camelbert-mix-did-nadi	lixiqi/beit-base-patch16-224-pt22k-ft22k-finetuned-FER2013-6e-05
classla/bcms-bertic-parlasent-bcs-ter	lixiqi/beit-base-patch16-224-pt22k-ft22k-finetuned-FER2013-7e-05
connectivity/bert_ft_qqp-1	lixiqi/beit-base-patch16-224-pt22k-ft22k-finetuned-FER-5e-05-3
connectivity/bert_ft_qqp-17	microsoft/beit-base-patch16-224
connectivity/bert_ft_qqp-7	microsoft/beit-base-patch16-224-pt22k
connectivity/bert_ft_qqp-96	microsoft/beit-base-patch16-224-pt22k-ft22k
dhimskyy/wiki-bert	microsoft/beit-base-patch16-384
distilbert-base-uncased	microsoft/beit-large-patch16-224-pt22k
DoyyingFace/bert-asian-hate-tweets-asian-unclean-freeze-4	mrgiraffe/vit-large-dataset-model-v3
emrecan/bert-base-multilingual-cased-snli_tr	sail/poolformer_m36
gchhablani/bert-base-cased-finetuned-rte	sail/poolformer_m48
gchhablani/bert-base-cased-finetuned-wnli	sail/poolformer_s36
ishan/bert-base-uncased-mnli	shi-labs/dinat-base-in1k-224
jb2k/bert-base-multilingual-cased-language-detection	shi-labs/dinat-large-in22k-in1k-224
Jeevesh8/512seq_len_6ep_bert_ft_cola-91	shi-labs/dinat-large-in22k-in1k-384
Jeevesh8/6ep_bert_ft_cola-47	Visual-Attention-Network/van-base
Jeevesh8/bert_ft_cola-88	Visual-Attention-Network/van-large
Jeevesh8/bert_ft_qqp-40	oschamp/vit-artworkclassifier
Jeevesh8/bert_ft_qqp-68	nateraw/vit-age-classifier
Jeevesh8/bert_ft_qqp-9	-
Jeevesh8/feather_berts_46	-
Jeevesh8/init_bert_ft_qqp-24	-
Jeevesh8/init_bert_ft_qqp-33	-
manueltonneau/bert-twitter-en-is-hired	-
roberta-base	-
socialmediaie/TRAC2020_IBEN_B_bert-base-multilingual-uncased	-
Splend1dchan/bert-base-uncased-slue-goldtrascription-e3-lr1e-4	-
XSY/albert-base-v2-imdb-calssification	-
Guscode/DKbert-hatespeech-detection	-

C. Dataset Details

NLP datasets and CV datasets are listed in Table IX. Some datasets contains multiple subsets. All datasets are available using ”https://huggingface.co/” as prefix. GLUE and SuperGLUE are the most common benchmark datasets in NLP. Cifar10 and MNIST are the most common benchmark datasets in CV. Other NLP datasets are described below:

- **LysandreJik/glue-mnli-train** This datasets contain labelled MNLI dataset. The original MNLI dataset in GLUE does not have label, and the label is necessary for our experiment. This task is to predict the relation between the premise and the hypothesis. The result could be entailment, contradiction, or neutral. The labels of this dataset are balanced.
- **SetFit/qnli** This datasets contain labelled qnli dataset. The original qnli dataset in GLUE does not have label, and the label is necessary for our experiment. This task is to predict whether or not the paragraph contains the answer to the question. The labels of this dataset are balanced.
- **xnli** This dataset contains part of MNLI dataset after translated into different languages. The labels of this dataset are balanced.
- **stsb_multi_mt** This task is to score the similarity between two sentences on the scale of 0 to 5. The labels of this dataset are not balanced.
- **anli** This task is the same as MNLI dataset. However, the dataset is collected in an adversarial procedure. The labels of this dataset are not balanced.
- **tweet_eval** This is a sentiment analysis task. The dataset is collected from Tweeter. The labels of this dataset are not balanced.
- **paws** This is a paraphrase identification task. The labels of this dataset are not balanced.
- **financial_phrasebank** This is a sentiment analysis task in the realm of finance. The dataset is collected from financial news. The labels of this dataset are not balanced.

TABLE IX
NLP AND CV DATASETS

NLP dataset name	CV dataset name
glue	food101
super_glue	nelorth/oxford-flowers
LysandreJik/glue-mnli-train	Matthijs/snacks
SetFit/qnli	beans
xnli	cats_vs_dogs
stsb_multi_mt	trpakov/chest-xray-classification
anli	cifar10
tweet_eval	MNIST
paws	alkzar90/CC6204-Hackaton-Cub-Dataset
financial_phrasebank	albertvillanova/medmnist-v2
yahoo_answers_topics	-

- **yahoo_answers_topics** This is a classification task. The dataset is collected from Yahoo. The labels of this dataset are balanced.

Other CV datasets are described below:

- **food101** This dataset contains 101 kinds of food that need to predict. The size of the image is not the same. The labels of this dataset are balanced.
- **nelorth/oxford-flowers** This dataset contains 102 kinds of flowers that need to predict. The size of the images is not the same. The labels of this dataset are not balanced.
- **Matthijs/snacks** This dataset contains 20 kinds of snacks that need to predict. The size of the images is not the same. The labels of this dataset are slightly unbalanced.
- **beans** This dataset contains 3 kinds of leaves that need to predict. The size of the images is the same. The labels of this dataset are balanced.
- **cats_vs_dogs** This dataset contains images of cats or dogs and is a subset of Asirra dataset. The size of the images is not the same. The labels of this dataset are balanced.
- **trpakov/chest-xray-classification** This dataset contains images of chest x-ray. The size of the images is the same. The labels of this dataset are not balanced.
- **alkzar90/CC6204-Hackaton-Cub-Dataset** This daatset contains images of birds. The size of the images is not the same. The labels of this dataset are not balanced.
- **albertvillanova/medmnist-v2** This dataset contains images about biomedical. The size of the image is the same. The labels of this dataset are not balanced.

D. *Experiment on the Number of Dimensions for Max Average Error*

As discussed in Eq. 1 and Section V.B., we use top-k maximum average error to measure the model similarity and the parameter k may influence the performance of the model selection algorithm. Thus, we test different values of k while fixing other items. Due to the number of datasets, we choose $k = 5, 10, 15$ for NLP clustering evaluation and $k = 3, 4, 5$ for CV clustering evaluation. The result is shown in Table X. We can find that the influence of parameter k is limited since the silhouette coefficient fluctuates within an acceptable range. Considering that the parameter k in Eq. 1 should be able to filter noise and retain valid information, we choose $k = 5$ in both tasks.

TABLE X
PARAMETER K SELECTION

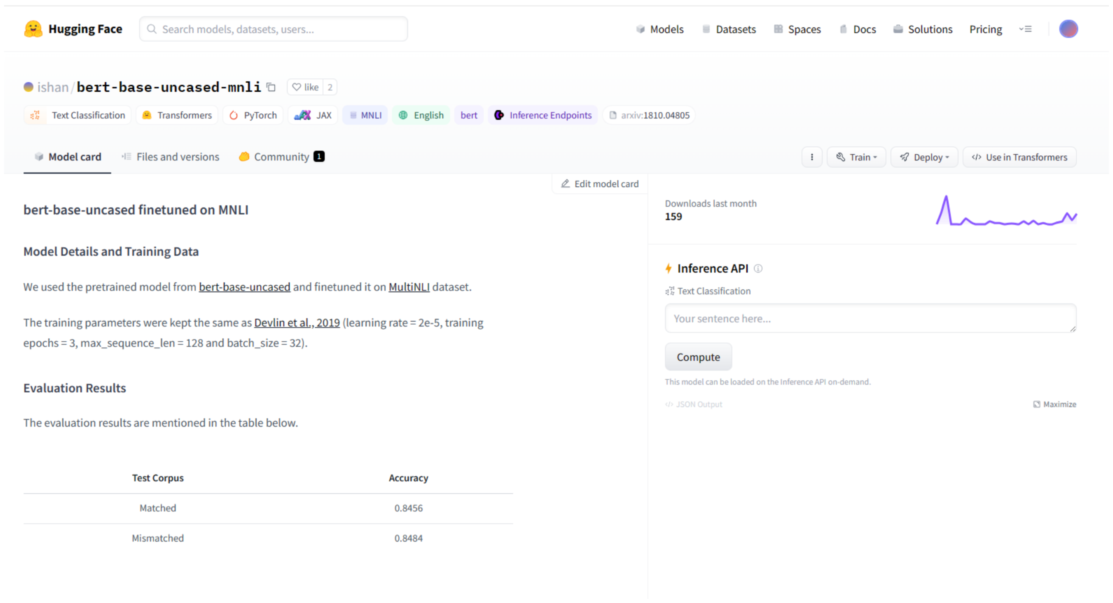
	NLP			CV		
K Value	5	10	15	3	4	5
Silhouette Coefficient	0.543	0.503	0.535	0.850	0.828	0.821

E. *Model cards*

A model card is given in Fig. 9. A model card contain the general description of the model, such as structure and training information.

F. *K-means Clustering Results*

The result of K-means clustering is shown in Table XI. This table is related to Table II in section V. B. Model Clustering. In that section, we explain the result of hierarchical clustering in detail. We conclude that the result of hierarchical clustering is effective since the in-cluster models share the same model structure or training dataset while the silhouette coefficient is high. Here we give the result of K-means clustering to better prove our conclusion. Both the NLP clustering result and CV clustering result of the K-means clustering algorithm show less connection between in-cluster models. In the NLP part, the 2 biggest clusters, C_2 and C_8, consist of a mix of models that have different structures and training datasets. In the CV part, there is a cross mixing in C_6 and C_7, and the biggest cluster, C_4, does not show consistency in either model structure or training dataset. Thus, we take the method of hierarchical clustering as the main line of this paper.



bert-base-uncased finetuned on MNLI

Model Details and Training Data

We used the pretrained model from bert-base-uncased and finetuned it on MultiNLI dataset.

The training parameters were kept the same as Devlin et al., 2019 (learning rate = 2e-5, training epochs = 3, max_sequence_len = 128 and batch_size = 32).

Evaluation Results

The evaluation results are mentioned in the table below.

Test Corpus	Accuracy
Matched	0.8456
Mismatched	0.8484

Fig. 9. Model card of bert-base-uncased-mnli. Each model on HuggingFace has a model card to describe the model.

TABLE XI
Model Clustering Results Using K-means

Model Clusters of Natural Language Processing		
Cluster	**Size**	**Pre-trained Models**
C_1	2	gchhablani–bert-base-cased-finetuned-rte, anirudh21–bert-base-uncased-finetuned-qnli
C_2	5	Jeevesh8–bert_ft_cola-88, DoyyingFace–bert-asian-hate-tweets-asian-unclean-freeze-4, bert-base-uncased , adityeabaral–finetuned-sail2017-xlm-roberta-base, Jeevesh8–512seq_len_6ep_bert_ft_cola-91
C_3	2	manueltonneau–bert-twitter-en-is-hired, aychang–bert-base-cased-trec-coarse
C_4	2	XSY–albert-base-v2-imdb-calssification, distilbert-base-uncased
C_4	4	ishan–bert-base-uncased-mnli, Alireza1044–albert-base-v2-qnli, albert-base-v2, Jeevesh8–feather_berts_46 :
C_5	2	CAMeL-Lab–bert-base-arabic-camelbert-mix-did-nadi, aliosm–sha3bor-metre-detector-arabertv2-base
C_6	3	socialmediaie–TRAC2020_IBEN_B_bert-base-multilingual-uncased, jb2k–bert-base-multilingual-cased-language-detection, emrecan–bert-base-multilingual-cased-snli_tr
C_7	2	dhimskyy–wiki-bert, bondi–bert-semaphore-prediction-w4
C_8	5	Jeevesh8–init_bert_ft_qqp-33, Jeevesh8–bert_ft_qqp-68, Jeevesh8–bert_ft_qqp-40, connectivity–bert_ft_qqp-1, Jeevesh8–bert_ft_qqp-9
C_9	4	connectivity–bert_ft_qqp-96, connectivity–bert_ft_qqp-7, connectivity–bert_ft_qqp-17, Jeevesh8–init_bert_ft_qqp-24
C_{10}	2	Splend1dchan–bert-base-uncased-slue-goldtrascription-e3-lr1e-4, Jeevesh8–6ep_bert_ft_cola-47
Model Clusters of Computer Vision		
Cluster	**Size**	**Pre-trained Models**
C_1	6	shi-labs/dinat-large-in22k-in1k-224, shi-labs/dinat-large-in22k-in1k-384, microsoft/beit-base-patch16-224-pt22k-ft22k, lixiqi/beit-base-patch16-224-pt22k-ft22k-finetuned-FER2013-7e-05, lixiqi/beit-base-patch16-224-pt22k-ft22k-finetuned-FER2013-6e-05, lixiqi/beit-base-patch16-224-pt22k-ft22k-finetuned-FER-5e-05-3
C_2	2	nateraw/vit-age-classifier, facebook/dino-vitb16
C_3	3	sail/poolformer_m48, sail/poolformer_m36, sail/poolformer_s36
C_4	7	facebook/vit-msn-small, facebook/vit-msn-base, facebook/deit-base-patch16-384, google/vit-base-patch32-224-in21k, Visual-Attention-Network/van-large, facebook/deit-base-patch16-224, facebook/dino-vits16
C_5	4	Visual-Attention-Network/van-base, microsoft/beit-large-patch16-224-pt22k, facebook/deit-small-patch16-224, shi-labs/dinat-base-in1k-224
C_6	2	microsoft/beit-base-patch16-384, google/vit-base-patch16-384
C_7	2	microsoft/beit-base-patch16-224, google/vit-base-patch16-224