Models Teaching Models: Improving Model Accuracy with Slingshot Learning

Lachlan O'Neill, Nandini Anantharama, Satya Borgohain, Simon D. Angus
SoDa Labs, Monash Business School, Monash University, Clayton Australia

Abstract

One significant obstacle to the successful application of machine learning to real-world data is that of labeling: it is often prohibitively expensive to pay an ethical amount for the human labor required to label a dataset successfully. Human-in-the-loop techniques such as active learning can reduce the cost, but the required human time is still significant and many fixed costs remain. Another option is to employ pre-trained transformer models as labelers at scale, which can yield reasonable accuracy and significant cost savings. However, such models can still be expensive to use due to their high computational requirements, and the opaque nature of these models is not always suitable in applied social science and public use contexts.

We propose a novel semi-supervised method, named Slingshot Learning, in which we iteratively and selectively augment a small human-labeled dataset with labels from a high-quality "teacher" model to slingshot the performance of a "student" model in a cost-efficient manner. This reduces the accuracy trade-off required to use these simpler algorithms without disrupting their benefits, such as lower compute requirements, better interpretability, and faster inference. We define and discuss the slingshot learning algorithm and demonstrate its effectiveness on several benchmark tasks, using ALBERT to teach a simple Naive Bayes binary classifier. We experimentally demonstrate that Slingshot learning effectively decreases the performance gap between the teacher and student models. We also analyze its performance in several scenarios and compare different variants of the algorithm.

1 Introduction

In standard computational linguistics modeling, the typical strategy for achieving high model accuracy is to obtain a large fraction of high-quality human labels on the dataset at hand. However, it is not uncommon in applied social science settings to find that the scope of human labeling of a dataset is highly constrained (Liew et al., 2014). For example, it may be prohibitively expensive to pay an ethical amount for human labeling. Alternatively, the project's institutional access to raw corpora may have ended, or perhaps the domain experts are no longer available.

One solution is to use an active learning approach where a model is trained concurrently with the labeling process to "maximize a model's performance gain while annotating the fewest samples possible"(Ren et al., 2021). While this can be more efficient than unguided labeling, it still requires significant human labeling resources as the learning process is fundamentally human-guided. Some issues are relatively fixed costs, such as bias, repeatability, and initial training requirements; this enforces a non-trivial lower bound on the efficiency of human-in-the-loop active learning methods. Active learning also does not help if the data has already been collected, and there is no way to collect further data.

Alternatively, one might directly apply a large, pre-trained (and possibly fine-tuned) transformer model to obtain state-of-the-art outcomes (Brown et al., 2020; Schick and Schütze, 2020; Wei et al., 2021). Pre-trained models are well suited to learning from relatively small amounts of data due to the large amount of prior knowledge they already contain. But such a strategy comes with complications for applied social science, including high (and rising) costs (Patterson et al., 2021; Schwartz et al., 2019; Mayfield and Black, 2020) a lack of model interpretability (Yang et al., 2021; Zafar et al., 2021), and issues with repeatability when using models hidden behind an API.

Given the power of modern machine learning models, a natural question arises: Can the power of modern machine learning architectures (such as transformer models) be harnessed to enhance the accuracy of standard models where human labeling

Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pages 3233–3247

is finished?

Here, we introduce and evaluate *Slingshot Learning*, an efficient, low-cost, widely applicable, and powerful methodology to enhance the accuracy of standard inference models. Slingshot Learning adopts a student–teacher framework, such that the standard model (the 'student') is iteratively taught by the transformer model (the 'teacher'). Akin to the gravitational slingshot maneuver, in which a spacecraft uses the gravity of a large body to increase its own speed, the student model leverages the knowledge and performance of the teacher model, in a way that allows it to express where it expresses low certainty in its predictions, essentially asking "questions" of the teacher model. This iterative behavior can be modified, or entirely replaced with purely random behavior through the selection of hyperparameters.

We demonstrate that, under slingshot learning, the accuracy of the student model can be at a relatively low cost. After the algorithm finishes iterating, the student model can be used as a compromise between the high performance of the complex teacher model, and the inherent benefits of the student model, which might include reproducibility, interpretability, computational efficiency, and compute costs that are orders of magnitude lower than the teacher model. This student model can then be used to label the rest of the corpus, and/or label future, unseen data.

2 Background

Active learning (AL) seeks to minimize the human labeling cost of training a model while collecting enough data to successfully train the model (Ren et al., 2021). It has proven popular in recent years due to the prevalence of large, unlabelled datasets, and methods are increasingly leveraging powerful, pre-trained models such as BERT (Grießhaber et al., 2020), but active learning is still a fundamentally human-led labeling approach which adopts the pitfalls of experimental research involving humans. The ability to repeat experiments in the first place (let alone with any consistency), expand their scope and analyze the underlying models are all taken for granted in machine learning but, once humans are involved in the labeling process, the difficulty of such tasks can vary from non-trivial to impossible. Iterative research design is also very difficult without significantly increasing the cost and time requirements of the overall experiment.

Large language models such as GPT-3 (Brown et al., 2020) have achieved exceptional performances in NLP tasks under zero-shot and few-shot learner constraints, making these a compelling approach for low-resource tasks (Chia et al., 2019). GPTs have been used in semi-supervised settings to generate pseudo-labels via training offline/student models (Chia et al., 2019). The study by Wang et al. (2021) leverages the use of GPT-3 to reduce costs in comparison to human annotators. In one of their labeling strategies, they employ GPT-3 (teacher) to generate a fully labeled dataset which is then used to train an offline model (student), a transformer model (Liu et al., 2019). While they show improved performance of the offline model on the fully labeled GPT-3 dataset in comparison to using human annotators, the performance on multi-class classifications was lower than that of human annotators and showed no improvement after increasing the number of GPT-3 labels. The authors also explore an active learning version, where the low-confidence labels of GPT-3 or the most challenging examples are relabelled using human annotators and the result is used to train the offline model. While this approach demonstrates better performance, it has the same pitfalls as other active learning approaches. Our approach assumes a low budget and no access to human annotators beyond a small "seed" labeled dataset.

In the realm of model interpretability, a similar approach is that of surrogate models (SM), which employ an interpretable simple model to explain the behavior of a more complex ML model (Rey and Neuhäuser, 2011). They include global models that explain the overall predictions and local models that provide local explanations on the predictors by varying their sampling strategies. The global models are trained on the same data set (or a similar data set) as the complex model (Molnar, 2022, ch 8.6). The output of the surrogate is then compared to the complex model to determine how well the surrogate replicates the complex model within an arbitrary threshold. The ones that employ sampling use simpler models to gain insight into interpretability through locally selected regions from the complex model's predictions (Ribeiro et al., 2016; Lundberg and Lee, 2017). The goal or output here is interpretability and mostly requires the continued use of the underlying complex model. However, our intention is not just interpretability but to build a stable simple model that is standalone.

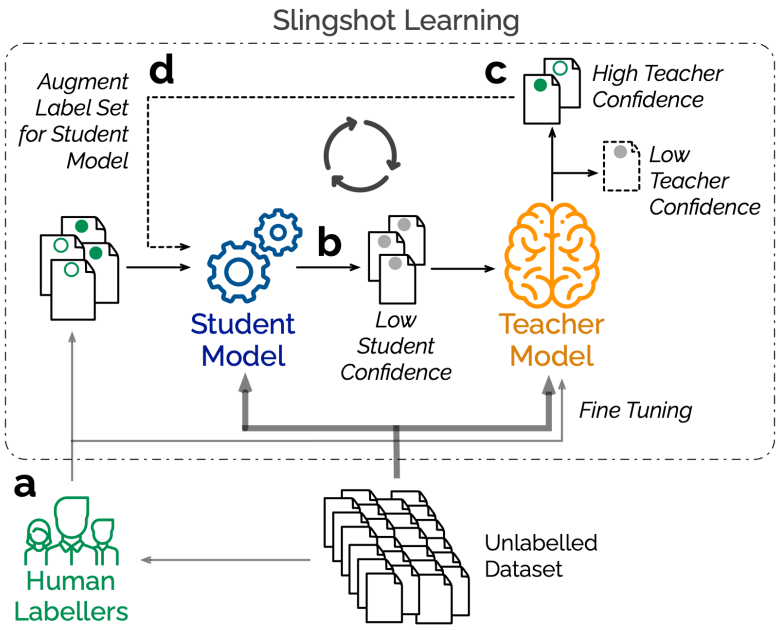

Figure 1: **Slingshot Learning.** A small fraction of the dataset is labeled by human labelers (a), which are used to train the Student Model and then identify low-confidence examples, a subset of which are passed to the fine-tuned Teacher Model (b). Where the Teacher Model has high inferential confidence on the examples (c), these are added to the human-labeled dataset for further training of the Student Model (d), and the cycle iterates.

The goal is to train the simpler model to make better predictions. Further, we are limited by our training dataset size. The idea is to also leverage a largely unlabelled dataset into training.

Knowledge distillation (KD) is another well-known approach that uses the teacher-student model (Hinton et al., 2015). The teacher - a deep teacher network - distills its knowledge onto a smaller model. The smaller models are also neural nets, and draw feature learnings from the parameters and activations of the intermediate layers of the teacher model. The intention here is to deploy a high-performing, compressed, smaller model in a low-resource environment (Gou et al., 2021) and thus they lack easy interpretability and transparency. While there have been some studies that use KD in an interpretable model (Liu et al., 2018; Che et al., 2016), the data size limitation discussed earlier applies to this approach as well.

Our goal is to build a standalone model that can achieve high accuracy despite limited training data, through the use of large unlabelled data sets.

3 Slingshot Learning

Here we explain slingshot learning from an intuitive view, and then present a more rigorous algorithmic definition.

3.1 Intuition

Consider a scenario where we have a large dataset of unlabelled data, far larger than could be labeled by hand, and we want to label the whole dataset and/or train a model to label future, unseen data. One solution is to hand-label a small portion by hand and feed the rest to a state-of-the-art model (such as a modern Transformer-based model, for NLP problems). While this is quicker and cheaper than labeling by hand, it can still be expensive (especially if using an API, or a large model requiring significant GPU compute capabilities). Additionally, these complex models take a while to run, making them potentially unsuitable for real-time applications. Of course, a simpler and more efficient model can be used, but this generally comes at the cost of decreased model performance.

In slingshot learning, we use the more expensive (HQ) model to label a portion of the unlabelled dataset, and then leverage that to *slingshot* the performance of the smaller (LQ) model, much like a gravitational assist maneuver accelerates a small spacecraft using the gravity of a large celestial object. Under this training paradigm, we train a "high-quality" (HQ) model on our ground truth dataset, use this model to iteratively teach the LQ model by labeling a portion of the unlabelled dataset, and

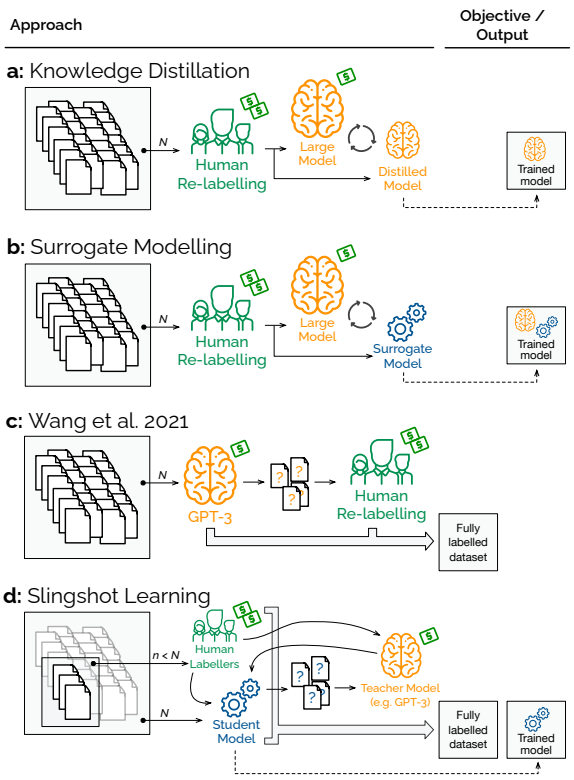

Figure 2: **Model comparison.** See text for details.

then use the low-quality model to label the rest of the data and/or make future predictions on unseen data.

We assume the number of human labels is fixed and cannot be increased - instead, we will use the HQ model (trained on the human-labeled dataset) as our more accurate (and more expensive) oracle. Our goal is thus to minimize the number of HQ predictions, while successfully labeling the dataset and/or training the LQ model. This is the crux of the slingshot learning approach; although the implementation details and hyperparameter selections might vary, the core concept of using a high-quality (but expensive) model to improve the performance of a lower-quality (but cheap) model remains constant.

The comparison to slingshot learning with other existing methods is presented in Table 1. The primary objectives of SM and KD are interpretability and high-accuracy compressed models respectively. Slingshot achieves both these objectives. Comparing slingshot to Wang et al. (2021), the latter continues to use human annotators for labeling, while SSL supports automated labeling. In contrast to Wang et al. (2021), KD's trained smaller models can perform automated labeling but require a large data set to achieve a desirable accuracy.

Model Characteristics	Wang et al.	SM*	KD*	SSL*
Automated labelling	✗	✗	✓	✓
Standalone re-usable model	✗	✗	✓	✓
Interpretability	✗	✓	✗	✓
Budget restriction	✓	✗	✗	✓
Small labelled dataset	✓	✗	✗	✓
Leverages Unlabelled data	✓	✗	✗	✓

Table 1: Comparison of existing methods with slingshot learning. SM*: Surrogate Models, KD*: Knowledge Distillation, SSL*: Slingshot Learning

3.2 The Slingshot Learning Algorithm

In Figure 3, a high-level pseudocode implementation of the core slingshot algorithm is described. Depending on the choice of hyperparameters, the algorithm's performance might be further optimizable. For example, in the purely random variant of the algorithm (discussed below), there is no need to compute the LQ predictions at each iteration.

Note that the algorithm does not assume that the confidence values of the HQ and LQ models are comparable with one another. The algorithm only relies on the relative ordering of confidence values from one of the models at a time and does not compare confidence values between the two models. The relationship between prediction confidence and accuracy is not necessarily monotonic, but we assume it will be somewhat correlated and treat it as monotonic for the purposes of the algorithm.

3.2.1 Hyperparameters

There are three hyperparameters in the slingshot learning algorithm:

n the number of data points selected for prediction by the *low-quality* model at each iteration.

m the number of data points selected for prediction by the *high-quality* model at each iteration.

k the number of data points added to the labeled set at each iteration.

We discuss the different choices for these hyperparameters throughout this paper. Note the necessary inequality $n \geq m \geq k$.

3.3 Expected Performance

Part of the intuition of slingshot learning stems from the idea that the improved performance of the HQ model over the LQ model stems from two sources: both the inherent superior ability of the HQ model to understand the *population* as a whole

procedure SLINGSHOT LEARNING ALGORITHM(L, U, V, n, m, k)
  $HQ \leftarrow$ **Train_HQ**(L) ▷ Train the HQ model on L
  $LQ \leftarrow$ **Train_LQ**(L) ▷ Train the LQ model on L
  while Not converged & more data **do**
    $D_n =$ **Sample**(L, n) ▷ Sample n elements from the unlabelled dataset.
    $P_n = LQ(D_n)$ ▷ Compute LQ predictions and confidence for each datum (note we only use the confidence).
    $D_m =$ **LowConfidence**(D_n, P_n) ▷ Choose m lowest confidence LQ predictions
    $P_m = HQ(D_m)$ ▷ Compute HQ prediction and confidence for each datum.
    $D_k =$ **HighConfidence**(D_m, P_m) ▷ Choose k highest confidence HQ predictions.
    $L \leftarrow L \cup D_k$ ▷ Add these to the labelled set.
    $U \leftarrow U \setminus D_k$ ▷ Remove these from the unlabelled set.
    $LQ =$ **Train_LQ**(L) ▷ Train a new LQ model on the full labeled dataset.
    Validate(LQ) ▷ Evaluate the performance of the LQ model on the validation set.
  end while
end procedure

Figure 3: Slingshot Learning Algorithm: Note that L is the labeled set, U is the unlabelled set, V is the validation set, LQ is the low-quality model, and HQ is the high-quality model. n, m, and k are as described in Section 3.2.1.

and also the superior ability of the HQ model to learn from the *sample*.

Let us denote the *intrinsic* capability of a model m to approximate the population, if given an arbitrarily large sample (i.e. "all the data", or an "infinite sample") from that population, as I_m. Since our sample will generally not cover the entire population, the true model accuracy will have a penalty applied dependent on the true*sample size* dependent capability of the model, which we denote as S_m. In our notation, we will assume the penalty value S_m is positive, and subtract it from the total model performance. We can then denote the LQ model performance as $LQ = I_{LQ} - S_{LQ}$, and the HQ model performance as $HQ = I_{HQ} - S_{HQ}$. We can then express the performance difference between the two models as:

$$HQ - LQ = I_{HQ} - S_{HQ} - (I_{LQ} - S_{LQ}) \quad (1)$$

Rearranging terms, we find:

$$HQ - LQ = \Delta_I + \Delta_S \quad (2)$$

Where $\Delta_I = (I_{HQ} - I_{LQ})$ and $\Delta_S = (S_{LQ} - S_{HQ})$. Intuitively, Δ_I is the performance uplift due to the *inherent* superiority of the HQ model (which cannot be conferred to the LQ model), and Δ_S is the extra penalty applied to the LQ model's performance due to the superiority of the HQ model given the *sample size* being used for training. In the limit, with infinite data, we would therefore expect Δ_I to approach some constant value greater than zero, while Δ_S goes to zero.

In situations where Δ_I is the dominant term, slingshot learning may have lower utility. But in scenarios where Δ_S is the dominant term (or at least a significant component of the overall performance delta), slingshot learning has the potential to bridge part of the knowledge gap between the HQ and LQ models. We conjecture that there are many instances where simpler, cheaper models would compare well to more complex and expensive models, and that much of the gap between traditional and newer models stems from both an increase in the I terms *and* a decrease in the S sample-size penalties for smaller samples.

3.4 Algorithm Variations

Depending on our choice of hyperparameters n, m, and k, we can modify the behavior of the slingshot learning algorithm in a variety of ways. An empirical comparison of the differences in resulting behavior from these three variants can be found in the Results section.

3.4.1 Deterministic Variant

We can choose to have the LQ model evaluate the entirety of the unlabelled dataset in each iteration, by setting $n = \|U\|$[1], where U is the set of data points that remain unlabelled at a given iteration.

This removes the stochasticity from the algorithm, assuming the HQ and LQ models remain constant under a given input (since m and k are only used to determine how many data points to

[1]We use the value $n = -1$ to denote this in our code.

take from the n randomly selected data points). While it will take longer to run - potentially a lot longer for larger datasets, since the unlabelled dataset must be passed through the LQ model at every iteration of the algorithm - his approach ensures that the LQ model can select the lowest confidence data points at every iteration.

3.4.2 Random Variant

Alternatively, we can choose to set $k = m = n$, meaning that every data point that is passed to the student model is sent to the teacher model for evaluation, and every data point sent to the teacher model for evaluation is added to the corpus. This is akin to purely random sampling, without allowing the student to provide input on the training process.

3.4.3 Stochastic Variant

We refer to any configuration of the hyperparameters n, m, k which is neither deterministic nor purely random as a "stochastic" configuration, where there is a combination of random sampling and student input during the sampling process. In stochastic slingshot learning, $k <= m <= n$ and $k < n$.

4 Experimental Design

4.1 Models

We initially ran experiments with ALBERT teaching Naive Bayes, ALBERT teaching SVM, and SVM teaching Naive Bayes. SVM teaching Naive Bayes proved difficult, as the SVM was very often not significantly more powerful than the Naive Bayes model. We performed some initial experiments with ALBERT teaching SVM, but the required computation time proved insurmountable. However, preliminary results indicate that SVM is a reasonable candidate as a student of ALBERT under the slingshot learning paradigm.

4.1.1 Model Details

ALBERT (Lan et al., 2020) from the HuggingFace Model repository (Wolf et al., 2020) (using the "albert-base-v2" model)

SVC (Cortes and Vapnik, 1995), an SVM classification model, as implemented by Sci-Kit Learn (Pedregosa et al., 2011). Not used in the final results.

Naive Bayes as implemented in Sci-Kit Learn (Pedregosa et al., 2011).

4.2 Datasets

We evaluate performance on four popular benchmark binary text classification datasets from the HuggingFace Datasets repository, as shown in Table 2.

4.3 Training

We tested a number of values for each of the three hyperparameters, n, m, and k. We evaluated $n = 128$, $n = 1024$, and $n = -1$. m and k were set to either $k = m = n$ or $m = n/4, k = m/4 = n/16$. When $n = -1$, a value of 1024 was used to calculate m and k. Each experiment was repeated up to 10 times. Experiments where the HQ (ALBERT fine-tuned) model was not at least 15% more accurate than the LQ (Naive Bayes) model were discarded, as we found they added a lot of noise to the dataset (and, in any case, slingshot learning is most useful when the models are of significantly different strength). Some number of samples [2] (one of 2500, $20,000$, or $50,000$) was taken from each dataset, and then randomly split into 80% training and 20% validation sets.

A total of $1,480$ experiments were run, representing up to five runs [3] of each of the 52 hyperparameter choices, over each of the three datasets. For each experiment, 95% of the dataset labels (chosen randomly but with a fixed seed of 42) were masked, to mimic a semi-supervised learning problem with a proportionally low (but reasonable) amount of ground-truth data.

Experiments were performed using virtual machines with four CPUs and a single NVIDIA T4 GPU, using AWS Batch. SVM and Naive-Bayes models were trained with default values from the sklearn library, as was the HuggingFace ALBERT model (which was trained for a total of three epochs over the dataset, with a batch size of 16).

5 Results

We have included all experimental results in tabular form in the supplementary materials. Source code can be found here.

When performing these experiments, we ran the full slingshot learning algorithm until there is no more data left to classify. However, in practice,

[2] For the Rotten Tomatoes dataset, we used all the data for the 20k and 50k runs since it has less than 20,000 data points.

[3] Due to our use of AWS spot instances, some experiments failed due to loss of the VM. Each experiment had at least 8 successful runs, before pruning for the disparity between the HQ and LQ models.

			Large				Small			
Dataset	**n**	**Variant**	$\delta_{10\%}$	$\delta_{30\%}$	δ_{max}	**#**	$\delta_{10\%}$	$\delta_{30\%}$	δ_{max}	**#**
Amazon Polarity	128	random	0.296	0.384	0.449	1	0.224	0.214	0.489	8
	1024	random	0.080	0.145	0.328	1	0.266	0.266	0.335	5
	1024	stochastic	0.178	0.346	0.478	3	0.208	0.344	0.507	8
IMDB	128	random	0.464	0.661	0.697	2	0.052	0.144	0.401	4
	1024	random	0.547	0.634	0.665	2	-0.249	-0.249	-0.204	3
	1024	stochastic	0.466	0.512	0.665	3	-0.070	-0.119	0.060	5
Rotten Tomatoes	128	random	0.170	0.305	0.527	11	0.008	0.099	0.226	3
	1024	random	0.235	0.334	0.507	16	0.216	0.216	0.293	4
	1024	stochastic	0.228	0.336	0.570	9	-0.020	0.047	0.236	4
Yelp	128	random	0.034	0.085	0.215	17	0.002	0.009	0.035	10
	1024	random	0.038	0.096	0.229	18	0.005	0.005	0.013	9
	1024	stochastic	0.028	0.084	0.218	14	0.001	0.002	0.025	9

Table 2: Here we show the performance of slingshot learning aggregated over the "small" samples (2,500 samples) and "large" samples (20,000 and 50,000 samples). All "small" experiments masked 90% of the dataset, leaving 10% as the seed for slingshot learning. All "large" experiments masked 99% of the dataset, leaving 1% as the seed for slingshot learning. Note that some of the experiments have very small sample sizes - these results should not be relied upon. The algorithm is random if $n = m = k$, and stochastic otherwise (we exclude $n = -1$ from this table as this configuration is not practical).

slingshot learning is most useful (and most distinct from other techniques such as surrogate models) when *early stopping* is applied. Our results demonstrate that there are cases where most of the accuracy gains occur early, and other cases where progress is more consistent. However, throughout almost all the results there is a consistent, monotonic increasing accuracy, as well as the *rate of change of accuracy* (that is, there are few sudden accuracy spikes). This indicates that simply monitoring the LQ model accuracy, and stopping the algorithm when it seems to be plateauing (or when resources have been fully expended, of course) might be a reasonable method of applying the algorithm. One could also attempt to quantitatively determine when the accuracy plateaus (if it ever does), but further exploration of this is beyond the scope of this paper.

Throughout these results, we refer to the δ ("Delta") metric. Intuitively, the Delta metric δ_{S_i} is the ratio between the amount of knowledge that the teacher model has "taught" the student model (equal to the difference between the performance of S_i and the baseline LQ model), and the knowledge gap between the teacher and the student (equal to the difference in their accuracy).

More rigorously, we define the Delta metric δ as

$$\delta_{S_i} = \frac{A(S_i) - A(S_{-1})}{A(T) - A(S_{-1})} \tag{3}$$

where S_i is the LQ model trained at the $i-th$ iteration of the algorithm, and $A(m)$ is the accuracy of model m on the dataset at hand. S_{-1} is the performance baseline of the LQ model before any iterations of the slingshot learning algorithm. T is the HQ model.

5.1 Empirical Results

Table 2 includes the average δ values at the 10% and 30% milestones (that is, after 10% or 30% of the total number of slingshot iterations), as well as the average point throughout the algorithm (as a percentage of the number of iterations taken) it reached its peak. The 10% and 30% points were chosen as reasonable benchmark points during our initial experiments. In the supplementary materials, we include extended tables including the first time the LQ model reached accuracy within 5%, 1%, and 0.1% of its peak accuracy.

In Figures 4, 5, and 6, we plot the δ values of the low-quality (LQ) model slingshot learning throughout the training process for each experiment (grey lines), along with the average δ across experiments (thick red line). We also show the 95% confidence

interval of the delta value (envelope), and, on the x-axis, the earliest iteration in which the LQ model obtains statistically equivalent performance (filled markers: $\alpha = 0.01$; open: $\alpha = 0.10$) to that of the final iteration (100%).[4] Figure 4 shows a best-case scenario for slingshot learning, where there is a significant increase in performance early in the iteration process. Figure 5 demonstrates how, in some cases, accuracy uplift can be more consistent throughout the slingshot learning cycle - in these cases, there is a stronger trade-off between the stopping point for slingshot learning and the potential accuracy gains of continuing the algorithm.

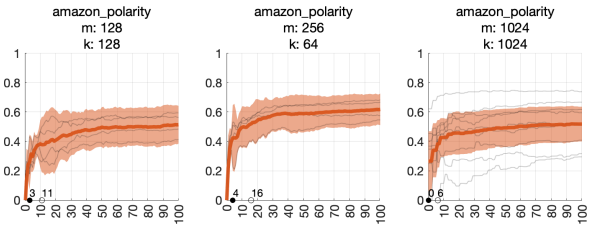

Figure 4: **Best-case scenario (Amazon Polarity dataset, 50k):** Large dataset sees the HQ model able to transfer most of the information to the LQ model without going through the whole dataset. Early stopping has less of a penalty.

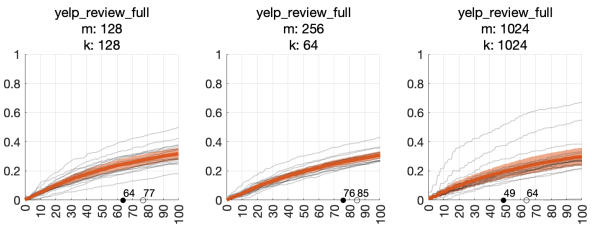

Figure 5: **More difficult scenario (Yelp dataset, 50k):** With much data, the HQ model can give information to the LQ model, but gains continue almost linearly throughout the process, penalizing early stopping.

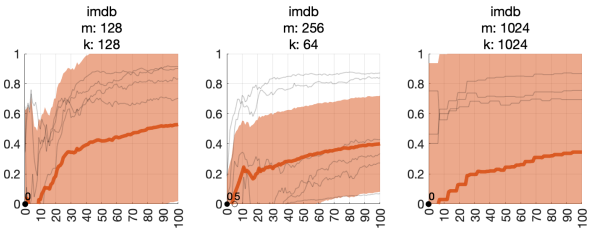

Figure 6: **High variability (IMDB dataset, 20k):** Slingshot learning performance was relatively poor on the IMDB dataset, though not always (see the supplementary materials for more information).

[4] In each figure: left panel, $n = 128$; middle, right panels, $n = 1024$.

6 Discussion

These empirical results show slingshot learning to be an effective and reliable method for improving model performance in semi-supervised classification problems. While the exact values of δ vary between datasets, each dataset shows a clear correlation between the number of iterations and the performance of the resulting model, as well as (in most cases) the effect of diminishing returns as the number of iterations is increased.

Due to the degeneracy problem discussed below, we have elected to remove any runs for which the initial HQ model did not show at least a 15% improvement in accuracy over the LQ model baseline (before any iterations of the algorithm).[5]

6.1 Hyperparameter Selection

We saw no consistent trends between the experiments where $n = m = k$ and $n > m > k$. It does seem that, in situations where an $n = m = k$ configuration would be too computationally expensive, an $n > m > k$ configuration (or perhaps an $n > m = k$ configuration) is a reasonable option. The supplementary materials include many more plots and more comprehensive tables for the 20,000 and 50,000 sample experiments.

6.2 Degenerate Cases

While slingshot learning has proven effective, and relatively reliable under the right conditions, it is not guaranteed to improve performance in all cases. Slingshot learning does not appear to be very beneficial in cases where the HQ model is not significantly more powerful than the LQ model. We found erratic behavior in such scenarios, to the extent that we chose to exclude any results from our tables and figures for which the HQ model did not perform with at least 15% higher accuracy than the LQ model.

6.3 Benefits of Slingshot Learning

Slingshot learning allows simpler models to capture some of the performance of more powerful models, and as discussed throughout this paper, this has tangible benefits in cost and compute requirements for several common machine learning applications. We have also identified several other benefits that this approach provides.

[5] The full, unpruned data table has been included in the supplementary materials.

6.3.1 Enabling Model Selection/Trade-offs

Traditional machine learning methods, such as Linear/Logistic Regressions, Naive Bayes, decision trees (Kotsiantis, 2013) and tree-based ensemble methods (Friedman, 2001; Chen and Guestrin, 2016), and SVMs, have existed in the literature for decades, and they have many advantages and trade-offs between them. However, as transformer-based models (and deep-learning models in general) become more powerful, it is becoming more and more difficult to justify the use of these traditional methods regardless of the benefits and trade-offs that may come with them, such as interpretability, efficiency, accuracy, and inherent applicability to the problem at hand.

The ability to choose between different models as the "low quality" model in the slingshot learning paradigm restores some of the power of model selection. For example, users can choose between simpler, more easily interpretable methods such as Linear/Logistic Regression or Naive Bayes, or opt for more complex models such as SVMs for the sake of accuracy. Through the use of slingshot learning, users can make these decisions and trade-offs based on their requirements, without sacrificing as much accuracy.

6.3.2 Enabling Experimental Repeatability

One negative effect of the proliferation of commercial, API-based deep learning models is the inability to reliably repeat experiments. APIs, and the models beneath them, are subject to change without notice, and will of course be improved over time. An unmodified experiment might perform significantly differently on two different days, in a manner entirely beyond the control of the user.

One interpretation of slingshot learning is as a method of capturing a "slice" of a large model, using a smaller model to store the captured information. These slices can then be stored and used at a later date, both reliably and without further interaction with the original model. Aside from the computational and financial benefits of this, the ability to take such a slice and store it locally enables repeatable, shareable experimental models. This may have legal and ethical implications, especially if it negatively affects the financial viability of commercial machine learning offerings, but a detailed discussion of this is beyond the scope of this paper.

6.3.3 Facilitating Interpretable Models

Deep learning models are not inherently interpretable by default, and although techniques such as LIME (Di Cicco et al., 2019) and SHAP (Lundberg and Lee, 2017) somewhat facilitate the interpretation of such models, there have been calls in recent years for inherently interpretable deep learning methods (Yang et al., 2021). By leveraging these complex, non-interpretable models to train simpler, interpretable models, slingshot learning can facilitate the creation of models which are both powerful *and* interpretable. Of course, the idea of training smaller models with larger ones is not new, and techniques such as knowledge distillation (Gou et al., 2021) and surrogate models (Rey and Neuhäuser, 2011; Molnar, 2022) are an area of active research.

6.3.4 Early Stopping

The iterative slingshot learning algorithm allows users to decide when to stop iterating when they determine that the model has likely converged. As discussed previously, this tends to happen relatively early in the iteration process, and further iterations tend to yield diminishing returns. Though there are no guarantees as to whether the model has converged (or is close to its final, converged performance) at a given iteration, the experimental data included in this paper can provide a guide to the expected future behavior of the model.

6.4 Further Work

Slingshot learning has proven to be useful in these applied NLP classification problems, but there is no reason it cannot be applied to other ML domains such as regression, multi-class classification, or text generation. Further experiments in these domains, and others, will be an important phase in determining the scope of the applicability of slingshot learning. We expect the algorithm would potentially be useful in a wide variety of scenarios.

Further analysis and testing of the different potential configurations of the slingshot learning algorithm's hyperparameters will also be critical to understanding how the values of n, m, and k can be selected to optimize the algorithm's performance in different use cases. We also plan to explore a potential "meta" slingshot learning algorithm, where progressively simpler models are taught from the model above them in the hierarchy, to propagate knowledge and avoid "overwhelming" the LQ model with the power of the HQ model.

7 Limitations

Our experiments were performed on English NLP datasets, collated from the internet. The results in this paper are intended and should be taken, purely in the context of evaluating the performance of the slingshot learning algorithm. We have no reason to believe slingshot learning would not be equally applicable to other languages, domains, problems, etc., and have simply chosen these datasets due to their prevalence and familiarity in the literature.

We conjecture that different (perhaps more complex) language structures might make slingshot learning more useful. We do not have expertise in this area - as we discussed in the paper, we consider it necessary to run further experiments evaluating the application of slingshot learning in other domains to be necessary. We are hopeful, however, that the method can be applied to many problems besides the ones discussed in this paper.

8 Ethics Statement

This paper discusses methods that are intended to enhance the ability of researchers within domains such as social sciences to more easily apply state-of-the-art NLP algorithms to their work. This inherently comes with the risk of bias, and it is entirely possible that seed datasets based on incomplete or biased datasets will cause slingshot learning to serve as an "amplifier". We encourage caution when applying machine learning to social or other "real-world impactful" domains, and doubly so when extrapolating from smaller amounts of data using machine learning. This paper serves as an initial evaluation and high-level exploration of slingshot learning, and we would not recommend using slingshot learning in situations that could potentially have negative consequences. Further exploration is certainly required, both specific to slingshot learning and regarding the more general use of machine learning in these scenarios.

On the other hand, methods like slingshot learning can empower researchers and allow them to leverage tools that would otherwise be infeasible. It also has other benefits, which are discussed in the paper - for example, it facilitates taking a "snapshot" of a black box model (such as an API-based NLP model). It is our intention, and our hope, that slingshot learning is a useful tool that helps democratize machine learning within the social sciences and other applied domains.

Acknowledgements

We thank all those who have reviewed this paper for their advice and contributions, which have included improvements to explanations and wording, discussion of relevant literature (particularly regarding active learning), and numerous other additions.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Zhengping Che, Sanjay Purushotham, Robinder Khemani, and Yan Liu. 2016. Interpretable deep models for icu outcome prediction. In *AMIA annual symposium proceedings*, volume 2016, page 371. American Medical Informatics Association.

Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA. Association for Computing Machinery. Event-place: San Francisco, California, USA.

Yew Ken Chia, Sam Witteveen, and Martin Andrews. 2019. Transformer to cnn: Label-scarce distillation for efficient text classification.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Vincenzo Di Cicco, Donatella Firmani, Nick Koudas, Paolo Merialdo, and Divesh Srivastava. 2019. Interpreting deep learning models for entity resolution: An experience report using lime. In *Proceedings of the Second International Workshop on Exploiting Artificial Intelligence Techniques for Data Management*, aiDM '19, New York, NY, USA. Association for Computing Machinery.

Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232. Publisher: JSTOR.

Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. 2021. Knowledge Distillation: A Survey. *International Journal of Computer Vision*, 129(6):1789–1819.

Daniel Grießhaber, Johannes Maucher, and Ngoc Thang Vu. 2020. Fine-tuning bert for low-resource natural language understanding via active learning. *arXiv preprint arXiv:2012.02462*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network.

Sotiris B Kotsiantis. 2013. Decision trees: a recent overview. *Artificial Intelligence Review*, 39(4):261–283. Publisher: Springer.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *ICLR*. OpenReview.net.

Jasy Suet Yan Liew, Nancy McCracken, Shichun Zhou, and Kevin Crowston. 2014. Optimizing features in active machine learning for complex qualitative content analysis. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 44–48.

Xuan Liu, Xiaoguang Wang, and Stan Matwin. 2018. Improving the interpretability of deep neural networks with knowledge distillation.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Elijah Mayfield and Alan W Black. 2020. Should you fine-tune BERT for automated essay scoring? In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 151–162, Seattle, WA, USA → Online. Association for Computational Linguistics.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, page 165–172, New York, NY, USA. Association for Computing Machinery.

Christoph Molnar. 2022. *Interpretable Machine Learning*, 2 edition.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*.

David A. Patterson, Joseph Gonzalez, Quoc V. Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David R. So, Maud Texier, and Jeff Dean. 2021. Carbon emissions and large neural network training. *CoRR*, abs/2104.10350.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B Gupta, Xiaojiang Chen, and Xin Wang. 2021. A survey of deep active learning. *ACM computing surveys (CSUR)*, 54(9):1–40.

Denise Rey and Markus Neuhäuser. 2011. *Wilcoxon-Signed-Rank Test*, pages 1658–1659. Springer Berlin Heidelberg, Berlin, Heidelberg.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144.

Timo Schick and Hinrich Schütze. 2020. Exploiting cloze questions for few-shot text classification and natural language inference. *CoRR*, abs/2001.07676.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2019. Green AI. *CoRR*, abs/1907.10597.

Shuohang Wang, Yang Liu, Yichong Xu, Chenguang Zhu, and Michael Zeng. 2021. Want to reduce labeling cost? GPT-3 can help. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4195–4205, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2021. Finetuned language models are zero-shot learners. *CoRR*, abs/2109.01652.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing.

In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Jian Yang, Gang Xiao, Yulong Shen, Wei Jiang, Xinyu Hu, Ying Zhang, and Jinghui Peng. 2021. A survey of knowledge enhanced pre-trained models. *CoRR*, abs/2110.00269.

Muhammad Bilal Zafar, Philipp Schmidt, Michele Donini, Cédric Archambeau, Felix Biessmann, Sanjiv Ranjan Das, and Krishnaram Kenthapadi. 2021. More than words: Towards better quality interpretations of text classifiers. *CoRR*, abs/2112.12444.

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

A Datasets

Table 3 includes some summary information about each of the datasets used in our experiments.

B Extended Results

The raw experimental data can be found in the supplementary materials, in CSV format. Selected plots and tables are included here.

Note that the Rotten Tomatoes dataset has less than $20,000$ elements, and thus all elements were used in every experiment of sizes $20,000$ and $50,000$ (i.e. with no random sampling). Rows with less than three experiments (i.e. $N < 3$) were dropped due to lack of statistical certainty.

The tables show the maximum δ value achieved throughout the training process ("Max"), along with when we first came within 5%, 1%, and 0.1% of the maximum. The place where the maximum value occurred is omitted as (due to noise) it is not a reliable or consistent statistic. The tables also include the δ values found after completing 10%, 30%, and 50% of the slingshot iterations.

Dataset	Training Set Size	Test Set Size
IMDB (Maas et al., 2011)	25,000	25,000
Amazon Polarity (McAuley and Leskovec, 2013)	3,600,000	400,000
Rotten Tomatoes (Pang and Lee, 2005)	8,530	1,066
Yelp Reviews (Zhang et al., 2015)	650,000	

Table 3: Description of datasets used in our experiments.

Dataset	Mask	n=m=k	Max	$\sim$5%	$\sim$1%	$\sim$.1%	Max	δ 10%	δ 30%	δ 50%	N
Amazon	0.9	FALSE	0.48	0.20	0.69	0.99	0.81	0.18	0.35	0.41	3
Amazon	0.99	TRUE	0.55	0.14	0.37	0.78	0.74	0.37	0.43	0.48	17
Amazon	0.99	FALSE	0.46	0.15	0.44	0.94	0.75	0.16	0.31	0.35	11
Amazon	0.999	TRUE	0.23	0.08	0.21	0.03	0.68	0.17	0.20	0.22	3
Amazon	0.999	FALSE	0.42	0.08	0.45	0.58	0.80	0.24	0.35	0.37	4
IMDB	0.9	FALSE	0.67	0.17	0.65	0.93	0.95	0.47	0.51	0.58	3
IMDB	0.99	TRUE	0.49	0.43	0.75	0.98	0.99	0.19	0.31	0.39	11
IMDB	0.99	FALSE	0.33	0.28	0.49	0.88	0.74	-0.03	0.07	0.18	8
R. Tom.	0.9	TRUE	0.51	0.43	0.83	1.00	0.93	0.24	0.33	0.43	16
R. Tom.	0.9	FALSE	0.57	0.34	0.72	0.94	0.92	0.23	0.34	0.45	9
R. Tom.	0.99	TRUE	0.67	0.40	0.86	0.91	0.95	0.36	0.52	0.57	5
Yelp	0.9	TRUE	0.23	0.41	0.86	1.00	1.00	0.04	0.10	0.15	18
Yelp	0.9	FALSE	0.22	0.34	0.84	0.98	0.99	0.03	0.08	0.13	14
Yelp	0.99	TRUE	0.21	0.26	0.71	1.00	0.99	0.04	0.09	0.14	19
Yelp	0.99	FALSE	0.24	0.31	0.77	0.98	0.99	0.03	0.09	0.14	17

Table 4: Experimental results, $\mathbf{n = 1024}$, combined data from sizes $20,000$ and $50,000$.

Dataset	Mask	n=m=k	Max	$\sim$5%	$\sim$1%	$\sim$.1%	Max	δ 10%	δ 30%	δ 50%	N
Amazon	0.99	TRUE	0.475	0.096	0.295	0.682	0.683	0.331	0.390	0.421	13
Amazon	0.999	TRUE	0.521	0.198	0.753	0.939	0.946	0.333	0.431	0.465	3
IMDB	0.99	TRUE	0.435	0.325	0.568	0.938	0.760	-0.008	0.210	0.294	13
R. Tom.	0.9	TRUE	0.527	0.349	0.794	0.963	0.917	0.170	0.305	0.378	11
Yelp	0.9	TRUE	0.215	0.352	0.832	0.988	0.994	0.034	0.085	0.132	17
Yelp	0.99	TRUE	0.250	0.289	0.766	0.992	0.938	0.039	0.107	0.158	18

Table 5: Experimental results, $\mathbf{n = 128}$, combined data from sizes $20,000$ and $50,000$.

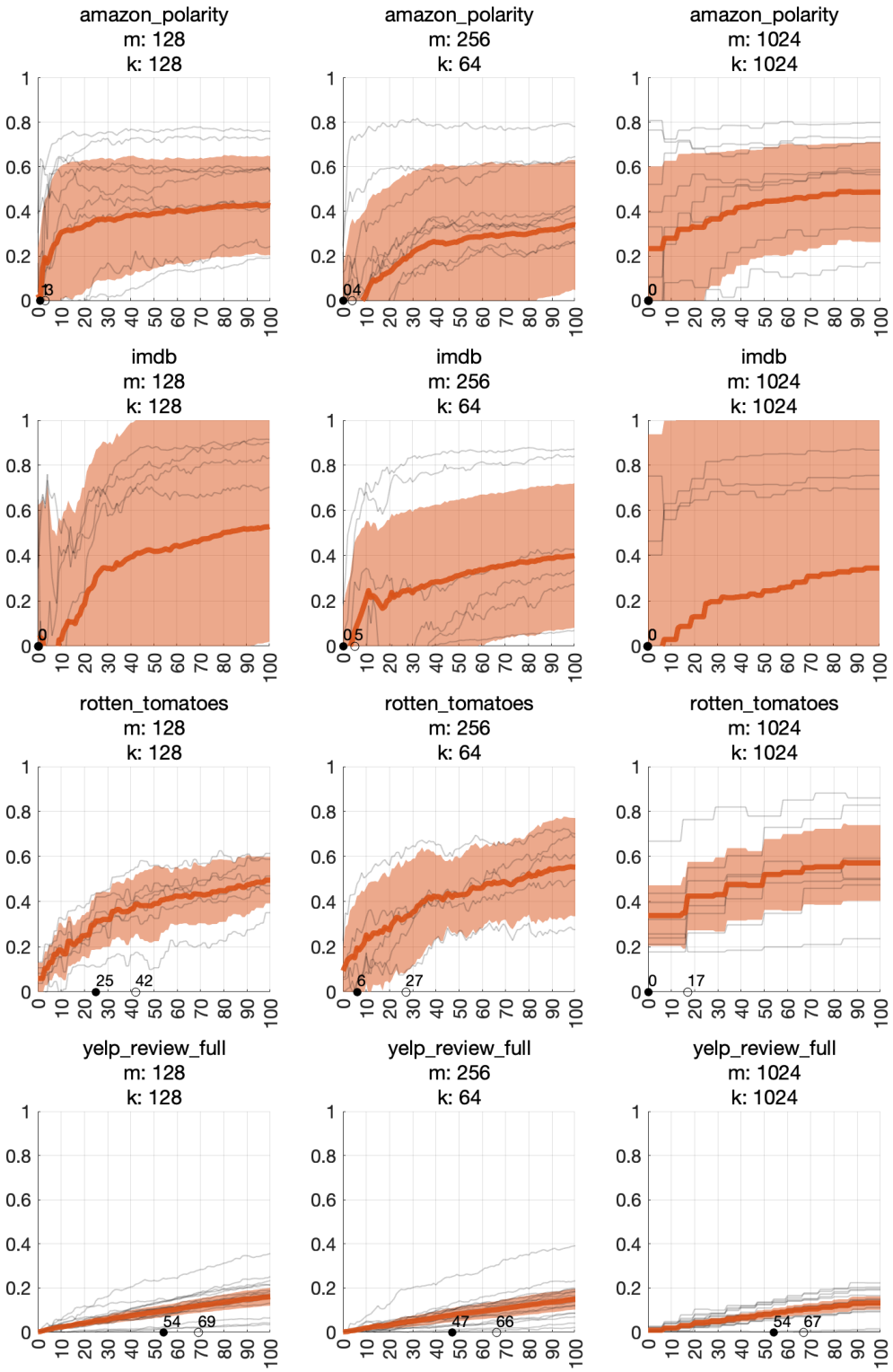

Figure 7: **Results for datasets of size** $20,000$ **(or less for Rotten Tomatoes).**

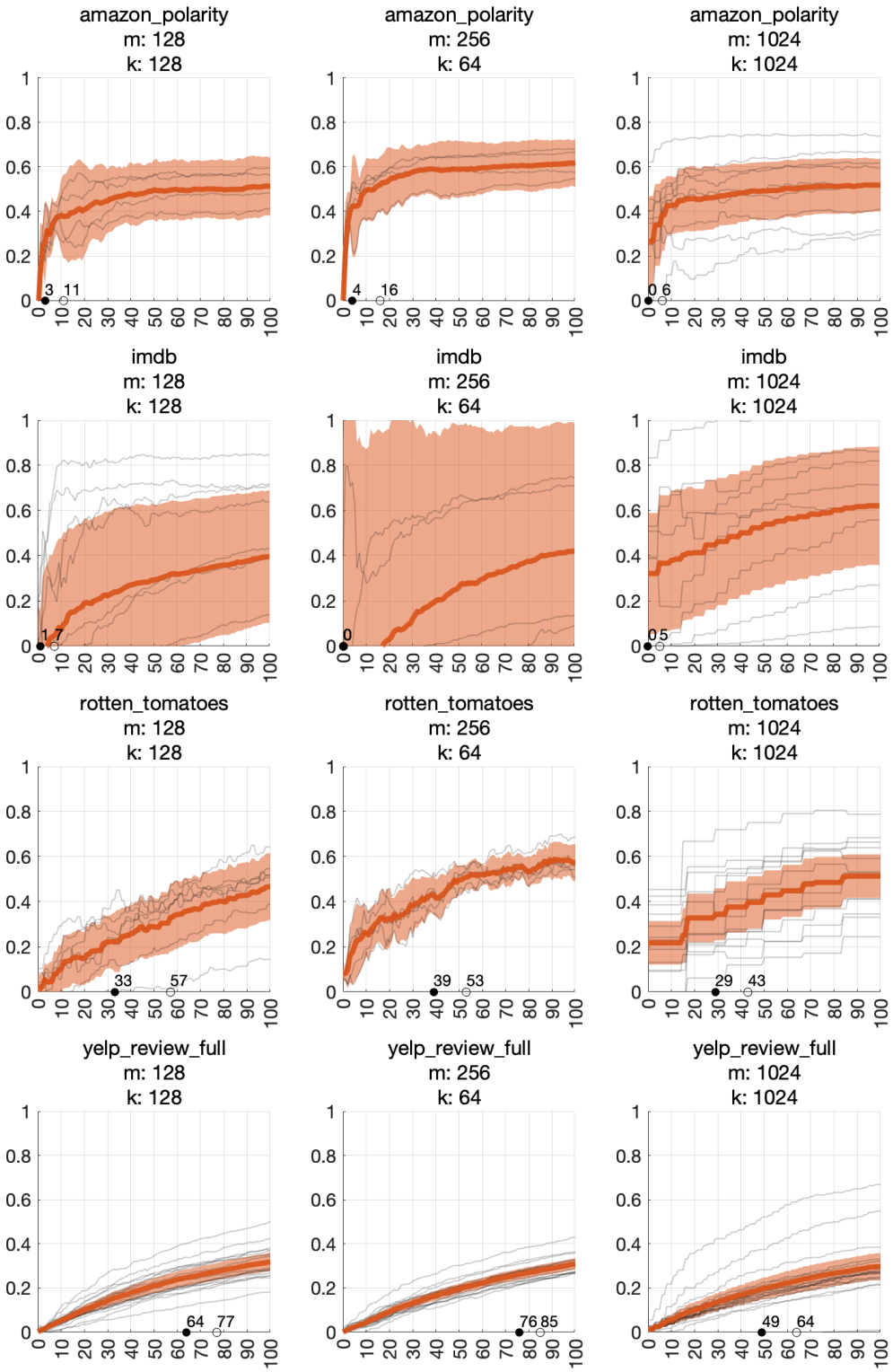

Figure 8: **Results for datasets of size** $50,000$ **(or less for Rotten Tomatoes).**