MIXED SETTING TRAINING METHODS FOR INCREMENTAL SLOT FILLING TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Model training remains a dominant financial cost and time investment in machine learning applications. Developing and debugging models often involve iterative training, further exacerbating this issue. With growing interest in increasingly complex models, there is a need for techniques that help to reduce overall training effort. While incremental training can save substantial time and cost by training an existing model on a small subset of data, little work has explored policies for determining when incremental training provides adequate model performance versus full retraining. We provide a method-agnostic algorithm for deciding when to incrementally train versus fully train. We call this setting of non-deterministic full- or incremental training "Mixed Setting Training". Upon evaluation in slot-filling tasks, we find that this algorithm provides a bounded error, avoids catastrophic forgetting, and results in a significant speedup over a policy of always fully training.

1 INTRODUCTION

The recent explosion in machine learning interest has led to substantial societal impact. However, with ever-growing model complexity comes corresponding growth in training time and cost. Recent models such as BERT (Devlin et al., 2019) can cost thousands of dollars and days to complete training, and similarly complex models consume staggering amounts of energy during training (Strubell et al., 2019). Approaches for reducing the training burden are necessary for enabling continued growth in the machine learning community.

Techniques such as incremental training can reduce the total time and financial investments associated with training models, while also enabling model refinement when original training data is unavailable. By using newly-added data as a basis for training, incremental training can lead to dramatic train-time speedups with marginal losses in accuracy (Ade & Deshmukh, 2013). However, little work has explored developing policies that inform users when incremental training can be used versus a full retraining phase of a model.

Previous research/findings in incremental training have been limited to a single setting in which the model either incrementally trains, or fully trains. In this paper, we describe an algorithm that chooses whether to train a model fully after a new observation of data, or simply leave the model incrementally trained. We refer to this kind of training as "Mixed Setting Training", as the model will have a decision rule of whether to train incrementally or not. Our goals for this method are as follows:

- **Training speedup**. Training represents a substantial burden financially and with respect to time. We seek a technique that saves training time overall.
- **Bounded errors**. We seek to apply incremental training such that a model's predictive performance is retained. We must provide a guarantee about how much error we introduce, and fully train instead if we cannot meet that guarantee.
- **Prevents catastrophic forgetting**. Approaches to saving on training time should avoid catastrophic forgetting.

We propose a new method, called 'Parameterized Incremental Training', to meet these goals. Our design provides a substantial speedup over always fully training, and guarantees by design that there

will be an upper bound on error ($\epsilon\%$). A hyperparameter, N, describes the size of the reservoir and thus provides predictable space requirements. Constant checks over an unbiased sample of the dataset prevent catastrophic forgetting.

2 BACKGROUND

In this section, we introduce incremental training and slot filling. We also discuss a motivating scenario that places our work in context.

Incremental training is is a setting where a machine learning model must be able to learn from newly added data without access to the entire dataset or model. This is particularly useful in situations where models must learn in an iterative fashion as new data is received, such as in online learning. Incremental training can produce dramatic train-time speedups with only marginal losses in accuracy (Ade & Deshmukh, 2013). Approaches typically reuse the previously trained weights and model on new tasks, modifying either the incoming dataset or the model itself. In an approach called One Shot Stochastic Gradient Descent (SGD), a single stochastic descent epoch is applied over the preexisting model and with a new observation. This approach proves easy to implement but has previously yielded disappointing results on text classification Golmant et al. (2017). Another approach is called reservoir sampling, which is similar to one-shot SGD but re-samples a small subset from the original dataset and trains the new example alongside the re-sampled examples. This has previously shown more promise and remains relatively computationally efficient and prevents a significant drop-off in accuracy as found with one shot gradient descent.

Slot-filling is a problem in natural language processing (NLP) where a model must be able to fill predefined 'slots' of information from a sentence of which 'values' are to be extracted. Common approaches to this are usually language models such as BERT (Devlin et al., 2019), ELMo (Peters et al., 2018), and XLNet (Yang et al., 2019), coupled with sequence prediction models (Mesnil et al., 2015). Slot-filling has applications across numerous domains, and is especially used in natural language understanding (NLU) settings in task-driven dialog systems where the goal is to extract relevant information (i.e. slots) from a human user who is trying to accomplish some task.

Research topics in incremental training have primarily focused on image processing, as computer vision processing often occurs on low power computers such as cell phones (Alyamkin et al., 2019). Information on the efficacy of incremental training techniques when applied to NLP is limited, and in the case of slot-filling, practically non-existent.

3 RELATED WORK

In this section, we discuss three lines of related work: (1) extant incremental training methods, (2) the concept of catastrophic forgetting, and (3) slot-filling as an example motivating application of our approach.

3.1 INCREMENTAL TRAINING METHODS

The ability to train Support Vector Machine (SVM) models in an incremental fashion has been explored by many researchers to handle the increasingly large training datasets, including Syed et al. (1999), Cauwenberghs & Poggio (2000), and Shilton et al. (2005). Fei-Fei et al. (2004) introduced a generative probabilistic model that can be trained incrementally in a Bayesian manner, which speeds up the training process significantly and shows superior accuracy compared to maximum-likelihood methods. Zhou et al. (2017) proposed AIFT, a convolutional neural network (CNN) architecture that can be fine-tuned continuously incrementally, which leverages transfer learning techniques to reduce the cost of data annotation for computer vision tasks by more than a half. Rebuffi et al. (2017) explored a novel training technique, iCaRL, which allows only a small number of classes to present at the beginning of the training process and new classes and samples can be added incrementally for image classification tasks.

3.2 CATASTROPHIC FORGETTING

Goodfellow et al. (2013) investigated the impact of catastrophic forgetting on a variety of machine learning algorithms including gradient-based neural networks, where they found the dropout algorithm can significantly improve the ability to adapt to new tasks for neural networks. Lee et al. (2017) presented a technique called incremental moment matching (IMM) that matches the posterior distribution of the neural networks trained on different tasks in an incremental fashion, therefore overcoming catastrophic forgetting. Kemker et al. (2018) formalized the metrics and benchmarks for evaluating the ability to mitigate catastrophic forgetting for neural networks, and also presented an empirical study of such techniques including regularization, ensembling, rehearsal, dual-memory, and sparse-coding. To mitigate the negative impact of catastrophic forgetting, Ritter et al. (2018) introduced the Kronecker factored online Laplace approximation that recursively approximates the posterior after each task using a Gaussian distribution, penalizing changes to the weights quadratically.

3.3 SLOT-FILLING

Surdeanu et al. (2010) presented an early attempt to use a distant supervision approach to tackle the slot-filling task, and an improved version that allows multi-label predictions is introduced by Surdeanu et al. (2011) one year later. Mesnil et al. (2015) proposed a recurrent neural network (RNN) architecture to capture the past and future temporal dependencies in natural language for semantic slot-filling tasks, which outperformed shallower conventional conditional random field (CRF) models. A more generalized multi-domain model architecture that can perform multiple tasks including slot-filling was described by Hakkani-Tür et al. (2016), which is composed of a bi-directional RNN with long-short term memory (LSTM) cells. To better extract relational knowledge among entities in slot-filling tasks, Zhang et al. (2017) combine an LSTM sequence model with a position-aware attention model achieving a 26.7% F_1 score improvement. This last approach, using an attention model, is time consuming to train. Moreover, none of the slot-filling approaches described above have been investigated under an incremental training setting.

4 APPROACH

In this section, we discuss our approach to informing incremental training policy. At a high level, we consider a scenario in which a user has built and deployed a model, and seeks to debug or refine the model by training it on samples seen subsequent to deployment. In such a setting, we seek to reduce the amount of time and resources the user must expend to train their model. We elect to apply incremental training, then rapidly evaluate the model to determine whether the accuracy of the model falls below a given threshold defined by the user (we denote this threshold with ϵ). We hypothesize that this policy will enable rapid retraining of a model in light of new data while retaining acceptable predictive performance. We describe our approach more rigorously in the remaining subsections.

4.1 PARAMETERIZED INCREMENTAL TRAINING

Algorithm 1 Parameterized Incremental Training

```
procedure PARAMETERIZED_TRAINING(examples, n, ϵ, W)
    for example in examples do
        reservoir ← RESERVOIR_SAMPLE(examples, n)
        W ← TRAIN_INCREMENTALLY(example, W)
        F1 ← EVAL(reservoir)
        if 100 − F1 ≤ ϵ then
            W ← TRAIN_FULLY(examples)
        end if
    end for
    return W
end procedure
```

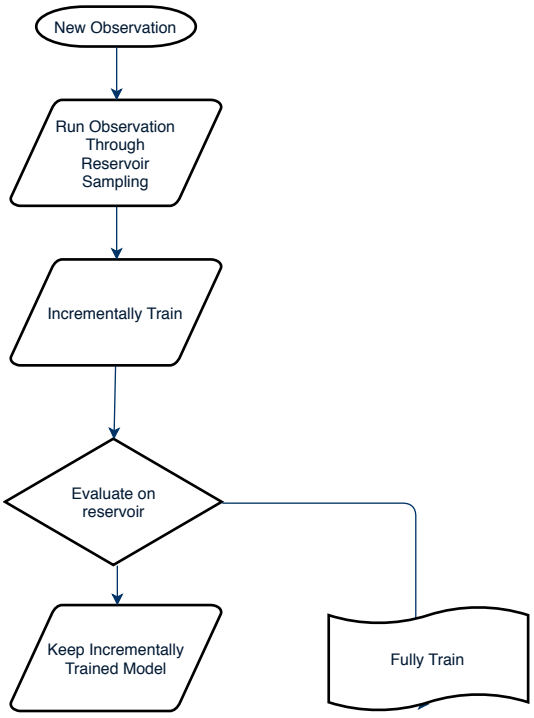

Figure 1: Visualization of Parameterized Incremental Training

We present a policy for determining when incremental training is an acceptable substitute for fully training a model. We assume we are given an initial dataset $\mathcal{D}$ on which we wish to tune a model characterized by a weight matrix W. We seek to ensure that the model does not catastrophically forget previously-learned patterns, so the algorithm retains a *reservoir* of size n that is randomly sampled from a stream of samples for which we have ground truth labels. This reservoir of samples is inferred before a training epoch. If the accuracy in this inference session is below an observed accuracy in the training session, then we fully train. Otherwise, we keep the candidate incrementally trained weight parameters, W. The algorithm is expressed in Algorithm 1.

In this algorithm, RESEVOIR_SAMPLE is a reservoir sampling algorithm that yields a resovoir set that enables incremental training while avoiding catastrophic forgetting; TRAIN_INCREMENTALLY is function that updates the model weights given a training sample (e.g., One Shot SGD or Reservoir Sampling Based SGD); TRAIN_FULLY is a function that trains the model from scratch (e.g., SGD); and EVAL is a function that computes the model performance measured using f_1 score on all of the samples in the reservoir. Since the reservoir is an unbiased sample of the training dataset as a whole and we ensure that the model's performance does not decrease below ϵ on the sample, the model's performance will not degrade by more than ϵ on the full training set.

4.2 Reservoir Sampling

Our approach leverages *reservoir sampling* to simultaneously retain predictive performance of incrementally-trained models while reducing the likelihood that catastrophic forgetting occurs during model training. Reservoir sampling is class of algorithms for sampling k items from a list S containing n items where n is unknown or very large or of limited access (Vitter, 1985). Let there be a sequence of items, denoted by S. We want to sample N number of items from that sequence. At the i^{th} item of the sequence, $S[i]$, where $i < N$, those items are stored into a sample array. When we see $S[i]$ where $i \geq N$ we decide with probability i^{-1} to add the new item into the set. We then decide to replace one of the old items, with each having i^{-1} probability of replacement.

The reservoir sampling algorithm that our incremental training algorithm uses is defined in Algorithm 2. Here, UNIF generates an integer from a discrete uniform distribution.

Algorithm 2 Reservoir Sampling

```
procedure RESERVOIR_SAMPLE(S, k)
    R ← []
    for i = 1 to k do
        R[i] ← S[i]
    end for
    for i = k + 1 to n do
        j ← UNIF(1, i)
        if j ≠ k then
            R[i] ← S[i]
        end if
    end for
    return R
end procedure
```

Table 1: Dataset features.

Dataset	Domain	Number of Samples	Number of Slots
ATIS (Hemphill et al., 1990)	—-	5,871	63
Jaech et al. (2016)	OpenTable	3,151	6

5 EVALUATION

We begin by evaluating the effectiveness of two candidate incremental training strategies for slot-filling tasks. These two strategies are One-Shot SGD and One-Shot SGD on a reservoir sample of size 100. We evaluate the effectiveness of our strategies on two commonly used slot-filling datasets: 1) ATIS (Hemphill et al., 1990), a dataset constructed for evaluating a flight-booking system; 2) the datasets introduced in Jaech et al. (2016), which are crowdsourced datasets covering 4 domains related to booking reservations for travel and restaurants. These datasets are summarized in Table 1, and the number of samples and slots in the datasets cover a wide range. Training was performed on an Intel Core i7 CPU with 16GB of RAM to simulate a resource constrained environment. Time measurements were taken just before beginning training, and just after training.

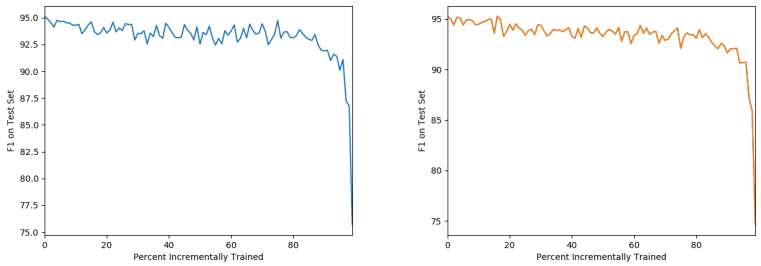

Figure 2: (Left) F_1 score on test set as model is incrementally trained in a one-shot SGD setting as a function of ratio of initial to incremental data (Right) F_1 score on test set as model is incrementally trained in a reservoir sampling batch setting as a function of ratio of initial to incremental data. n=100

We start by comparing the effects of gradually increasing the incremental set, gradually increasing the initial training set, and varying both simultaneously. For the reservoir, our N was a size of 100. This was chosen because it represents a use case for incremental training: a practitioner looks at the model and then decides to gather 100 more examples. For each data point we train a model with an initial partition of the data, incrementally train once on another set of the data, and then evaluate the F_1 score. In figure 3 we start out in all cases by initially training on 50% of the dataset. We chose 50% as an arbitrary starting point because we assume that the average practitioner will collect

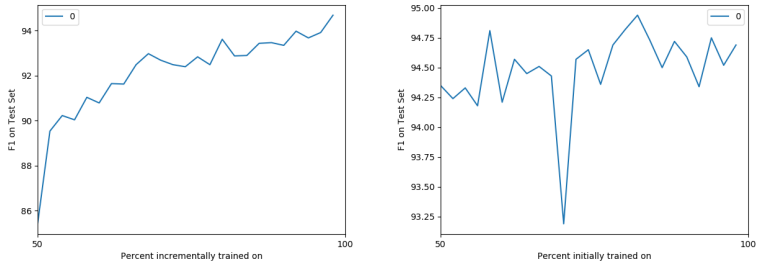

Figure 3: (Left) F_1 score on test set as model is incrementally trained in a one-shot SGD setting as the number of incremental examples increases. (Right) F_1 score on test set as model is incrementally trained in a one-shot SGD setting as the number of initial examples increases

Table 2: Empirical measurements of model training time and performance using our approach to inform training policy.

Dataset	**F_1 Score**	**Seconds to Train**	**Ratio Fully Trained**
OpenTable	87.54	15147.28	90.00%
ATIS	91.8	106718.53	78.57%

up to twice the size of the original dataset before retraining. We use a RNN slot-filling approach using ELMo embeddings. We find that One Shot SGD performs admirably in all settings similar to reservoir sampling the training set. Because of the performance and ease of implementation, we chose one shot SGD as our incremental training method for Parameterized Incremental Training. Additionally, we also report good performance as long as the model initially trains on 20% of the data.

We then move into a streamed data setting. We compare the F_1 score and ratio of training sessions that were fully trained to those that were incrementally trained. The acceptable accuracy loss in Table 2 was set to 5%. Our number of examples per observation was set to 100 and this again was done with the ATIS dataset and OpenTable. We observe that the policy that never incrementally trains is dramatically slower than the other methods, but provides higher accuracy. Parameterized Incremental Training provides a reasonable tradeoff of speed and accuracy, offering a 10% reduction in time on the OpenTable dataset and a 22.43% reduction in time on the ATIS dataset with no greater than 5% accuracy loss.

6 Conclusion and Discussion

Incremental training is an important technique when training models quickly with limited access to data. Research has been done previously on techniques, and our data shows a policy to decide when to incrementally train. We find that on slot-filling tasks on the ATIS and Opentable datasets that One Shot SGD and reservoir sampling methods have roughly equivalent accuracy on the training set. This is in contrast to the findings in Golmant et al. (2017), where they found a significant accuracy increase for reservoir sampling. We suspect that slot-filling tasks differ architecturally from classification tasks such that one shot SGD provides better accuracy. We hope that this paper will open a field of inquiry into decision algorithms for training in a mixed setting.

References

R. R. Ade and R. Deshmukh. Methods for incremental learning: A survey. *International Journal of Data Mining Knowledge Management Process*, 3:119–125, 2013.

Sergey Alyamkin, Matthew Ardi, Alexander Berg, Achille Brighton, Bo Chen, Yiran Chen, Hsin-Pai Cheng, Zichen Fan, Chen Feng, Bo Fu, Kent Gauen, Abhinav Goel, Alexander Goncharenko, Xuyang Guo, Soonhoi Ha, Andrew Howard, Xiao Hu, Yuanjun Huang, Donghyun Kang, and

Shaojie Zhuo. Low-power computer vision: Status, challenges, and opportunities. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(2):411–421, 2019.

Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine learning. In *Proc. of NIPS*, 2000.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL*, 2019.

Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *Proc. of CVPR*, 2004.

Noah Golmant, Evan R. Sparks, and Joseph Gonzalez. Batch methods for incremental learning. pp. 4, 2017.

Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint*, 2013.

Dilek Hakkani-Tür, Gokhan Tür, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Proc. of Interspeech*, 2016.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. The ATIS spoken language systems pilot corpus. In *Proc. of the Workshop on Speech and Natural Language*, 1990.

Aaron Jaech, Larry Heck, and Mari Ostendorf. Domain adaptation of recurrent neural networks for natural language understanding. In *Proc. of Interspeech*, 2016.

Ronald Kemker, Marc McClure, Angelina Abitino, Tyler L Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *Proc. of AAAI*, 2018.

Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *Proc. of NIPS*, 2017.

G. Mesnil, Y. Dauphin, K. Yao, Y. Bengio, L. Deng, D. Hakkani-Tür, X. He, L. Heck, G. Tur, D. Yu, and G. Zweig. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539, 2015.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental classifier and representation learning. In *Proc. of CVPR*, July 2017.

Hippolyt Ritter, Aleksandar Botev, and David Barber. Online structured laplace approximations for overcoming catastrophic forgetting. In *Proc. of NIPS*. 2018.

A. Shilton, M. Palaniswami, D. Ralph, and Ah Chung Tsoi. Incremental training of support vector machines. *IEEE Transactions on Neural Networks*, 16(1):114–131, 2005.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. In *Proc. of ACL*, 2019.

Mihai Surdeanu, David McClosky, Julie Tibshirani, John Bauer, Angel X. Chang, Valentin I. Spitkovsky, and Christopher D. Manning. A simple distant supervision approach for the tac-kbp slot filling task. In *Proc. of Third Text Analysis Conference*, 2010.

Mihai Surdeanu, Sonal Gupta, John Bauer, David McClosky, Angel X. Chang, Valentin I. Spitkovsky, and Christopher D. Manning. Stanford's distantly-supervised slot-filling system. In *Proc. of Fourth Text Analysis Conference*, 2011.

Nadeem Ahmed Syed, Huan Liu, and Kah Kay Sung. Handling concept drifts in incremental learning with support vector machines. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1999.

Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, March 1985. ISSN 0098-3500.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. XLNet: Generalized autoregressive pretraining for language understanding. *arXiv:1906.08237*, 2019.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. Position-aware attention and supervised data improve slot filling. In *Proc. of EMNLP*, 2017.

Zongwei Zhou, Jae Shin, Lei Zhang, Suryakanth Gurudu, Michael Gotway, and Jianming Liang. Fine-tuning convolutional neural networks for biomedical image analysis: Actively and incrementally. In *Proc. of CVPR*, 2017.