Mitigating Catastrophic Forgetting in Deep Learning in a Streaming Setting Using Historical Summary

1st Sajal Dash
National Center for Computational Sciences
Oak Ridge National Laboratory
Oak Ridge, TN, USA
dashs@ornl.gov

2nd Junqi Yin
National Center for Computational Sciences
Oak Ridge National Laboratory
Oak Ridge, TN, USA
yinj@ornl.gov

3rd Mallikarjun Shankar
National Center for Computational Sciences
Oak Ridge National Laboratory
Oak Ridge, TN, USA
shankarm@ornl.gov

4th Feiyi Wang
National Center for Computational Sciences
Oak Ridge National Laboratory
Oak Ridge, TN, USA
fwang2@ornl.gov

5th Wu-chun Feng
Department of Computer Science
Virginia Tech
Blacksburg, VA, USA
feng@cs.vt.edu

Abstract*—Recent advancements in scientific equipment and the adaptation of electronics and the Internet of Things (IoT) in our everyday lives resulted in large and complex data production at a high rate. Making meaningful and timely knowledge discovery at a modest cost from this big data is difficult for computing power and storage limitations. Training deep learning models incrementally in a streaming setting can help us with overcoming these limitations. However, in a well-known phenomenon named *catastrophic forgetting*, incrementally trained models increasingly perform poorly on the past data. To mitigate catastrophic forgetting in training in a streaming setting, we propose constructing a historical summary over time and use the summary with newly arrived data during incremental training. We propose various data summarization techniques such as random sampling, micro clustering, coreset computation, and Auto Encoders to counteract catastrophic forgetting. We built a pipeline for incremental training with a historical summary for training deep learning models for streaming data. We demonstrate the effectiveness of historical summary in mitigating catastrophic forgetting using three case studies involving three different deep learning applications: an Artificial Neural Network (ANN) for classification task on MNIST dataset, a language model (RNN-LM) on the WikiText2 dataset, and a Convolutional Neural Network (CNN), ResNet50 to classify the ImageNet dataset. Through the training of the models, we observe that catastrophic forgetting is evident in ANN and CNN but not in an RNN. For the first task, our method recovers up to 47.9% lost accuracy due to catastrophic forgetting. For the third task, the historical summary recovers classification accuracy by up to 25%. For the second task, though there is not proof of catastrophic forgetting, the training performance (PPL) improves by up to 26% with historical summary.

I. Introduction

In many real-life and scientific applications, observational data arrives in a stream with a need to gradually discover knowledge. For example, we need to make instantaneous routing decisions from real-time traffic surveillance images. Weather scientists need to make a live predictive map of the storm's path from streaming meteorological data in a developing storm. In in-situ scientific simulations, where outcomes of previous steps determine the next steps and their parameters, a decision must be made online from the simulation's streaming data.

Training Deep Learning (DL) models in streaming settings can be helpful in these scenarios. Besides naturally occurring streaming data sources, big data analytics can also benefit from learning in a streaming setting when the data is too large so that learning from it overwhelms system memory and compute capability. In that case, we can break down the data into manageable chunks and emulate a streaming setting. Incremental learning from a streaming data source can be beneficial when we have access to the streaming data for a limited amount of time. We cannot store the entire dataset due to memory and storage constraints or analyze the whole dataset due to memory and compute constraints.

Deep neural networks, when trained incrementally, suffer from the changing nature of the data in a stream. A few challenges arise in training machine learning models in streaming settings [1].

1) **Online model parameter adaptation:** How to update the model parameters with the new data?
2) **Concept drift:** How to deal with data when the data distribution changes over time?
3) **The stability-plasticity dilemma:** If the model is too stable, it can't learn from the new data. When the model is too plastic, it forgets knowledge learned from the old data. How to balance these two?
4) **Adaptive model complexity and meta-parameters:** The presence of concept drift or introducing a new class of data might necessitate updating and increasing model complexity against limited system resources.

The primary focus of our work is to identify the challenges associated with catastrophic forgetting. Catastrophic forgetting is a special case of the stability-plasticity problem, where the model becomes too plastic and forgets about the tasks learned

earlier [2]–[4].

Our specific contributions are following:

1) We introduce a streaming pipeline for training DL models with historical summary (Section III).
2) We demonstrate catastrophic forgetting and its mitigation using historical summary through 3 case studies: (a) an artificial neural network (ANN) on the MNIST datase (Section IV), (b) a language model based on a Recurrent Neural Network (RNN) (Section V), and (c) a complex DL model, ResNet50 using the ImageNet dataset (Section VI).

Next, we present our survey on various state-of-the-art approaches to tackle these challenges in training DL models in a streaming setting in Section II.

II. Addressing Challenges with Learning in a Streaming Setting

Different bodies of research have addressed the learning from streaming data challenge with data distributional and neural architectural approaches. For example, *ADAIN* framework [5] describes an adaptive learning method in a streaming setting that learns a distribution of the current data based on its classification performance on the past model and then adjusts the past model using the current distribution. The distribution assigns higher weights to the data points harder to learn and lower weights to the data points which are easier to learn. At any given time t, this framework assumes there is a known distribution P_{t-1} of the data from the earlier arrival at time $t-1$ and a hypothesis (model) h_{t-1}. It learns current distribution by first mapping the past distribution P_{t-1} to current data and then by assessing the error when this mapped distribution is tested against the past distribution, h_{t-1}. ADAIN is tested on simple datasets comprising up to 6 classes and several models, including Multi-layer Perceptron (MLP). While ADAIN shows comparable performance, its usability for more complex data such as ImageNet (1000 classes) and DL models is unknown.

SCIL algorithm [6] deals with streaming data by maintaining a set of neurons per class and by occasionally merging the neurons to keep the network size under control. The set of neurons form a set of hyper-ellipsoids to capture clusters of data in the same class. Versatile Elliptic Basis Function (VEBF) is used to combine the neurons. VEBF is used to define the boundary of the discarded data from earlier arrivals, and with the arrival of new data and new classes, these boundaries are updated. SCIL was tested on the dataset with the number of classes up to 26, data points up to $581K$, and outperformed other reported streaming algorithms in the classification task. One obvious limitation of this work is that it was not tested on a dataset with many classes. SCIL addresses concept drift and adaptive model complexity.

Sahoo et al. [7] proposed an online training model to address adaptive model complexity. It uses Hedge Back-propagation, which dynamically decides on the depth of the training model. Since the too shallow model converges too fast and performs poorly on new data, and too deep model converges very slowly to achieve a reasonable amount of learning, this training model maintains the range of layers. With time, the number of layers required converges.

Besedin et al. [8] designed an evolutive deep model to allow training in an online streaming setting. This model addresses concept drift and catastrophic forgetting. Their model learns from the available data. To forward the current data, they construct a Generative Adversarial Network (GAN) on the current data and forward this GAN to the next step. Every time new data arrives, this approach trains the model on that data and the data generated by the GAN from the previous step.

Kemker et al. [9] categorized approaches to mitigate catastrophic forgetting in the following classes.

1) Regularization methods. Elastic Weight Consolidation (EWC) [10] adds a regularization term to the loss function to preserve weights contributing to past learning.
2) Ensemble methods. Learn++ [11] learns a new model per episode with the help of the distribution of the data and ensembles these models.
3) Rehearsal methods. A few rehearsal methods reuse past data; one such method, GeppNet [12], preserves past data and incorporates these into the current episode's training through a Self Organizing Map (SOM) [13].
4) Dual-memory methods. GeppNet+STM [12] maintains a short-term memory for new uncertain predictions and consolidates these with more long-term memory.

We present here a new approach which has similarities to a distribution-driven approach we call "Historical Summary" which we discuss next, and we present evaluation case studies in Section III.

III. Mitigating Catastrophic Forgetting Using Historical Summary

We propose to train the model on historical data as well as current data without stressing the storage requirement. In every episode (Definition 1), we train our model on the currently arrived data and the summary of historical data.

Definition 1. Episode (i)*: A discrete time step i when a new set of data arrives in a streaming setting.*

Figure 1 illustrates the proposed pipeline. First, the arriving data unit will go through a data-reduction tool (clustering, dimension reduction using PCA/auto-encoder, core-set approximation, sampling, etc.). We will deposit the summary to a *Historical Summary* module. At any *Episode*, the *Historical Summary* up to the previous *Episode* will be used along with the current *Episode*'s data to incrementally train the model.

a) Effectiveness of historical summary in mitigating catastrophic forgetting: Catastrophic forgetting prevents effective learning while training deep learning models in a streaming setting. There are various efforts to mitigate this phenomenon with varying degrees of success. We propose historical summary construction as a means to counteract this. Several summarization techniques such as k-means clustering,

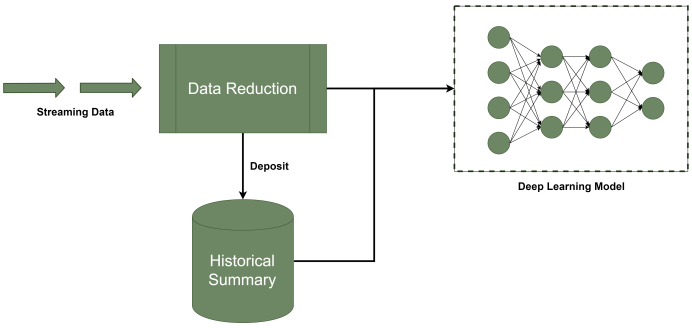

Fig. 1: A streaming model of training Deep Learning (DL) models. As the data arrives in a stream, we perform a fast summarizing process on this. We combine this summary with the data in "Historical Summary" buffer. This way, we train the model incrementally on newly arrived data along with a representative set from the past.

random sampling, autoencoder, and PCA can be useful for constructing historical summary.

Definition 2. *Model(i): Incrementally trained model at the end of* Episode i.

A. Learning Tasks in Streaming Episodes

To demonstrate efficacy of our approach, we will emulate four *Episode*s. There will be a task associated with each *Episode*. The model gets trained incrementally after every data arrival (at every *Episode*). We name the classification tasks on the data from these four *Episode*s as Task A, Task B, Task C, and Task D respectively.

B. Historical Summary Construction

To construct the summary for one Episode's data, we randomly sample 10% of the training examples. We construct *Historical Summary*, we accumulate the *Episode* summaries. Ideally, we expect accumulating summaries from all past *Episode*s to deliver the best performance in mitigating catastrophic forgetting. In some of the case studies, we will only use the immediate past summary to construct the *Historical summary*. We will provide more details in the later sections describing the case studies.

IV. Case Study I: Streaming Learning on MNIST Dataset

We investigated the conditions under which catastrophic forgetting occurs. When data arrives uniformly in all *Episode*s, catastrophic forgetting is less likely. When a subset of classes arrives in each *Episode*, we observe catastrophic forgetting to take place.

MNIST is a database of handwritten digits widely adopted to instruct and demonstrate various concepts related to deep learning [14]. It has 60000 training examples and 10000 test examples. The most popular machine learning task associated with this database is classifying the images into one of the ten digit classes.

We simulate two types of streaming scenarios using the MNIST dataset to demonstrate catastrophic forgetting and our mitigation effort. In the first scenario, in each of the streaming *Episode*s, the entirety of the MNIST dataset arrives, but with an *Episode*-unique permutation of the pixels (Figure 2). This way, we are introducing new shapes which are not consistent with shapes from an earlier *Episode*. This is a common practice for emulating streaming scenarios adopted by [2], [10].

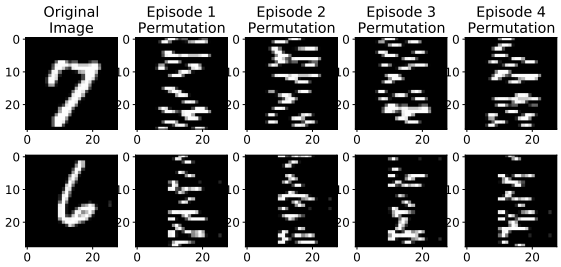

Fig. 2: Emulation of data arrival in four *Episode*s for the first scenario. In a given *Episode*, the digits **7** and **6** are permuted in the same way. For simplicity we only permuted the rows of pixel matrix in this example.

In the second scenario, we use unaltered MNIST images; but we emulate only a subset of the ten classes being present in each *Episode*. This way, the model is continually forced to learn more about images from a new subset of classes. The subsets are not mutually exclusive, so the model does not need to move away from past data completely. We simulated the arrival of images from six randomly chosen classes from a total of ten classes at any *Episode* (Figure 3).

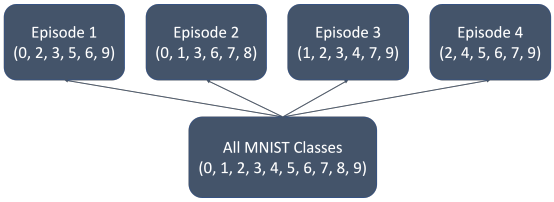

Fig. 3: Emulation of data arrival in four *Episode*s for the second scenario. A subset of six classes out of 10 classes is randomly selected in each *Episode*; examples from these classes are drawn to emulate the data arrival for this *Episode*.

In both scenarios, we present the model with some new data and test it for its knowledge retention about past data. To investigate the potential of historical summary in mitigating catastrophic forgetting, we compare a simple summary method (i.e., random sampling) against the EWC method on two different types of tasks.

A. Historical Summary Construction

For *Episode* summary, we take 10% of the training examples from that *Episode*. To construct *Historical Summary* for *Episode* i, we accumulate summary from all Episode j, where $j < i$.

B. Evidence of Catastrophic Forgetting

During every *Episode*, we train the model incrementally with the newly arrived data and set it to perform classification tasks on the data from earlier *Episode*s. *Model(1)* will have no past tasks, *Model(2)* will have tasks *Task A*, and *Task B*. *Model(2)* will have tasks Task A, Task B, and Task C. The final Model, *Model(4)*, will have all four tasks to perform: Task A, *Task B*, *Task C*, and *Task D*. *Model(i)* is expected to perform best on the data from *Episode* i. We investigated whether a *Model(i)* performs poorly on data from *Episode*s earlier than i.

In the top left panels of Figures 4 and 5, we show how *Model (2)* performs on *Task A* (on data from *Episode* 1) and *Task B* (the data from the current *Episode*). As we see, with the progressing training iteration, Model(2) performs increasingly better on *Task B* but loses more than 50% classification accuracy on *Task A* (Figure 5a). Similarly, we see that *Model(3)* gets increasingly better at *Task C* while performing poorly on *Task A* and *Task B* (Figure 4d,5d). The final Model, *Model(4)*, performs best on the task involving the most recent data (*Task D*) while performs worse on the previous tasks (*Task A*, *Task B*, *Task C*) (Figures 4g, 5g). The general trend here is that the incrementally trained model performs best on the tasks associated with the most recent data. But the earlier into the past we look, the worse its performance becomes. Hence, the incrementally trained model is *forgetting* the knowledge learned from the past data. This is evidence of catastrophic forgetting.

C. Mitigation Using Historical Summary

On the tasks from the first scenario, the historical summary is as good as the EWC when recovering from catastrophic forgetting for up to three *Episode*s. During the fourth *Episode*, the EWC becomes unstable, but the *Historical Summary* performs consistently through all four *Episode*s (Figure 4). On the second type of task, EWC suffers in recovering from catastrophic forgetting even though it performs better than vanilla stochastic gradient descent (SGD). The historical summary outperforms both of these methods (Figure 5) by a margin.

V. Case Study II: Streaming Learning on LSTM using WikiText2 Dataset

We use an RNN-based language model (RNNLM) [16], [17] to see if catastrophic forgetting happens when we train a language model on a text dataset. Our RNNLM consists of an encoder, an LSTM, and a decoder. We train this model on the WikiText2 dataset and measure the performance of the model using perplixity score (PPL). We split the WikiText2 Dataset into four *Episode*s. The result of the training is shown in Figure 6. The language model's task is to predict next sequence once it has seen the current sequence. A good model will assign high probabilities to realistic and syntactically correct sequences and a low probabilities to fake, incorrect, or artificial sequences. Hence, it should assign high probability scores to the sentences from the test set. Perplexity is the normalised inverse probability of the test set ($\sqrt[N]{1/P(w_1, w_2, \ldots, w_N)}$). A low perplexity on the test set indicates that the model is well trained to predict realistic and correct sentences.

A. Historical Summary Construction

To emulate data arrival for an *Episode*, we randomly select 25% of the sentences. The length of the sentences may vary, so overall character count across *Episode*s can vary. To construct *Episode* summary, we randomly select 10% of the sentences from that *Episode*, and we accumulate all past summaries to construct *Historical Summary*.

B. Catastrophic Forgetting in RNNLM

WikiText2 dataset is a collection of Wikipedia articles. We split the dataset sequentially into four *Episode*s in a way that each *Episode*'s data has roughly the same number of words and the split happens between sentences to avoid fractional sentences in any of the dataset.

However, unlike the two other case studies, we see that the incrementally trained model actually shows better performance on all *Episode*'s data (Figure 6, blue bars), including the past *Episode*s. For example, while the PPL score for *Episode* 0's data stays roughly the same throughout the four *Episode*s, the PPL score for *Episode* 1 and 2's data actually improves (decreases, lower PPL indicates better model performance). So, there is no evidence of catastrophic forgetting in a RNN based model from our experiment. This is probably due to nature of RNN model. A recurrent neural network is trained using new data over time and an LSTM has built in mechanism for preserving *long short-term memory*. So, this is probably free from catastrophic forgetting because of its training paradigm.

C. Effect of Historical Summary on Incremental Training of RNNLM

We have not seen any evidence of catastrophic forgetting in an RNN-based language model. Keeping in line with the other two case studies, we wanted to see the impact of historical summary on the training process. So, we used 10% randomly selected sentences from the earlier *Episode*s in the incremental training of the model at any given *Episode* and measured the same perplexity scores on the test dataset from the four *Episode*s. The PPL scores with summary improve further compared to the PPL scores without any summary (Figure 6). Hence, while the catastrophic forgetting is not evident from the streaming learning of RNNLM, historical summary helps with the training process nevertheless.

VI. Case Study III: Streaming Learning on ImageNet

ImageNet is an image database widely adopted by deep learning researchers with 1000 categories. The *ImageNet Large Scale Visual Recognition Challenge (ILSVRC)* has driven a stream of research in developing large-scale deep learning models and efficient scaling techniques. The dataset has 1.21 million images.

For emulating the data arrivals in four different *Episode*s, we take a similar strategy as in Figure 3. In each *Episode*, we

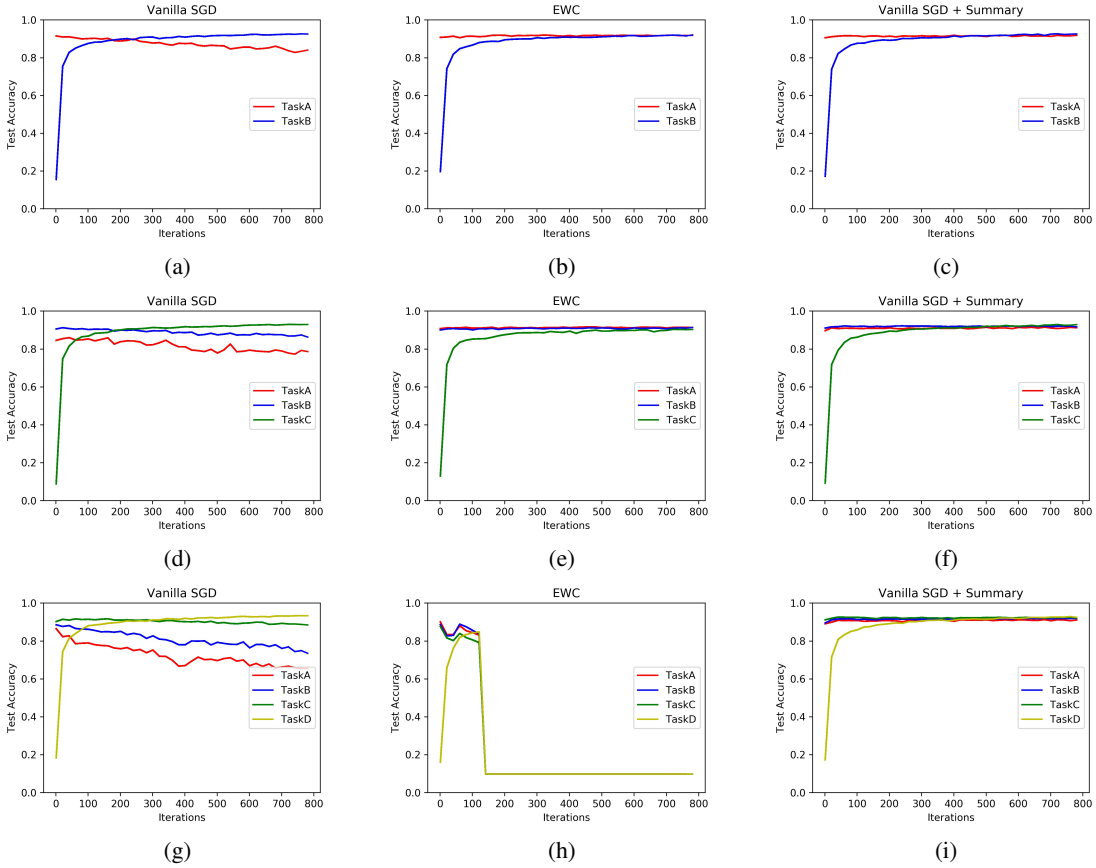

Fig. 4: Test accuracy for the first scenario of the case study I while training the models using three methods: incrementally trained only with SGD (left), trained with EWC (middle), and trained with *Historical Summary* constructed through random sampling with SGD. Each row corresponds to an *Episode* (2, 3, 4). *Historical Summary* recovered up to 92.3% - 65.2% = 27.1% lost accuracy (Task A). We used EWC implementation by [15]. For pixel permutation, we only permuted the columns. So, this change is not entirely destructive; it still retains some spatial locality information. This is likely the reason for the smaller value of loss in accuracy on past tasks. Subfigure h shows a drastic drop in the accuracy resulting from EWC's loss computation. However, with an early stop at around the 100th iteration, we can avoid that.

randomly select a subset of 600 classes out of the 1000 classes and take the examples from these classes to emulate the data arrival for that *Episode*. Then we train the model incrementally with each *Episode*'s arrived data. In each *Episode*, we train the model for only 30 epochs.

For large dataset like ImageNet arriving in each *Episode*, we need to deploy data-parallel or model-parallel training. In this casestudy, we deploy data parallel training where available data is distributed across available machines using PyTorch dataset reader and Horovod [18] for managing efficient communication across nodes of Summit supercomputer. The workload across nodes is balanced by the nature of how a large minibatch is distributed across the models. Figure 7 illustrates the result of this experiment.

A. Historical Summary Construction

Since ImageNet is a large dataset, we only took 5% of the data to build an *Episode*'s summary. For the same reason, we also take only the immediate past summary to construct *Historical Summary*.

B. Evidence of Catastrophic Forgetting in Training of ResNet50 with ImageNet Dataset

When we trained a regular ResNet50 model incrementally without any summary, the classification accuracy for the first Task A dropped from 60% to 50% in three *Episode*s. For Task B, the accuracy dropped from 78% to 55% in two *Episode*s. For Task C, the accuracy dropped from 78% to 55% in just one *Episode*. So, there is clear evidence of catastrophic forgetting while we train a ResNet50 model using the ImageNet dataset without any summary. The incrementally trained model $Model(i)$ performs best on the data from *Episode* i. The catastrophic forgetting can be seen in the Figure 7.

C. Mitigation Using Historical Summary

We use a summary of the past data to mitigate catastrophic forgetting during training in a streaming setting. Here, to train a model at *Episode* i, $i > 1$, we use 10% examples from the

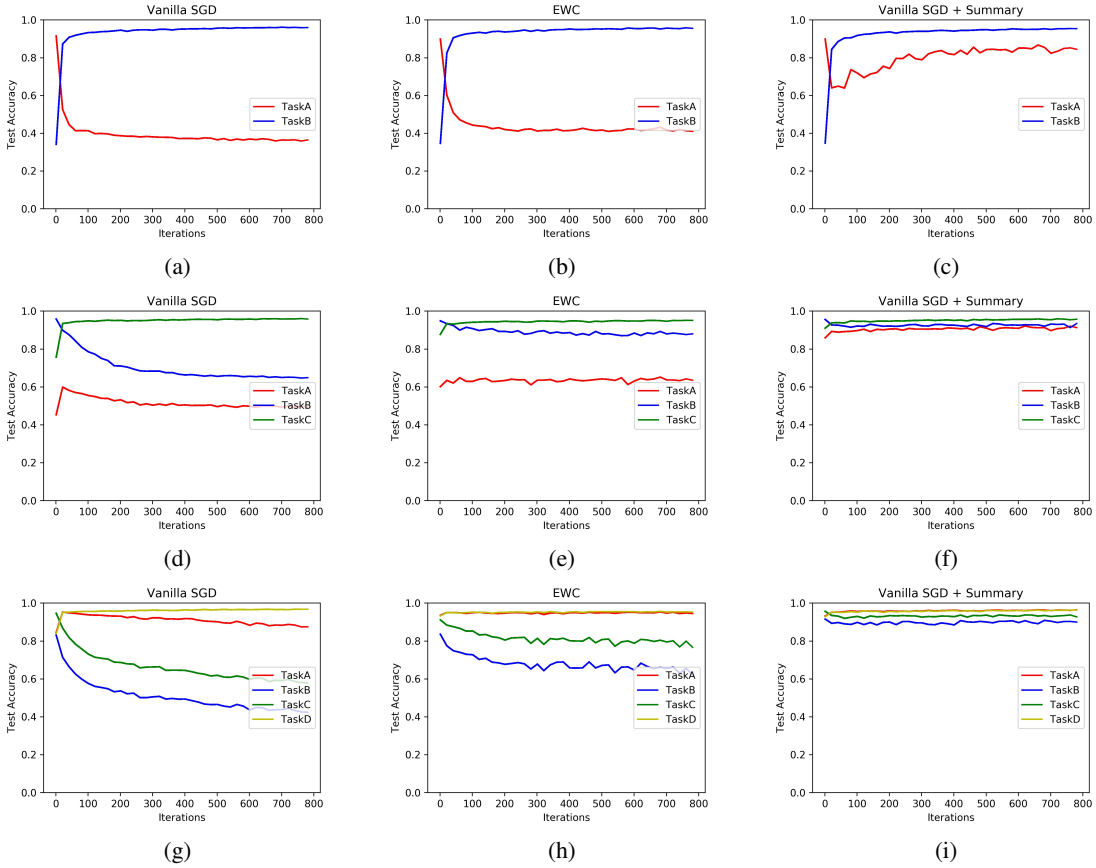

Fig. 5: Test accuracy for the first scenario of the case study II while incrementally training the models using three methods: trained only with SGD (left), trained with EWC (middle), and trained with *Historical Summary* constructed through random sampling with SGD. Each row corresponds to an *Episode* (2, 3, 4). *Historical Summary* recovered up to 83.9% - 36% = 47.9% lost accuracy in Episode 2.

Episode $i-1$. We use a summary of past data just from the earlier *Episode*, then mix it with the newly arrived data at the current *Episode* to train our model incrementally.

With the summary from just the earlier *Episode*, we can see that the incrementally trained models perform much better on the past data (Figure 7, orange plot). For Task A, the classification accuracy is retained at 60% as it was in *Episode* 1. For Task B, the accuracy is improved to 60% from its 50% accuracy without summary. For Task C, the accuracy is at 65%, which is an improvement from its accuracy without summary (55%).

While summary helps to mitigate catastrophic forgetting, it does not entirely eliminate it. Summary from more than one *Episode* into the past might be more beneficial.

VII. Conclusion and Discussion

Catastrophic forgetting of the incrementally trained models in a streaming setting presents a problem with retention of past knowledge. In this paper, we have proposed a training paradigm which uses *Historical Summary* to mitigate catastrophic forgetting. There are many choices for summary construction, either through dimension reduction, or through choosing a smaller number of samples. To illustrate a simple summarization technique such as random sampling can be effective in mitigating catastrophic forgetting, we designed experiments with various types of tasks and compared our method against a state-of-the-art method, EWC and the vanilla training. Our experiments on both types of tasks show that *Historical Summary* performs better than EWC and vanilla SGD.

We found that catastrophic forgetting is more evident in ANN and CNN, while it might not be as evident in an RNN. However, the *Historical Summary* helps with the training performance whether there is catastrophic forgetting or not. While The findings of this work show promise in solving a long-standing problem in streaming learning.

Acknowledgements

This research was sponsored by and used resources of the Oak Ridge Leadership Computing Facility (OLCF), which is a DOE Office of Science User Facility at the Oak Ridge National Laboratory supported by the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

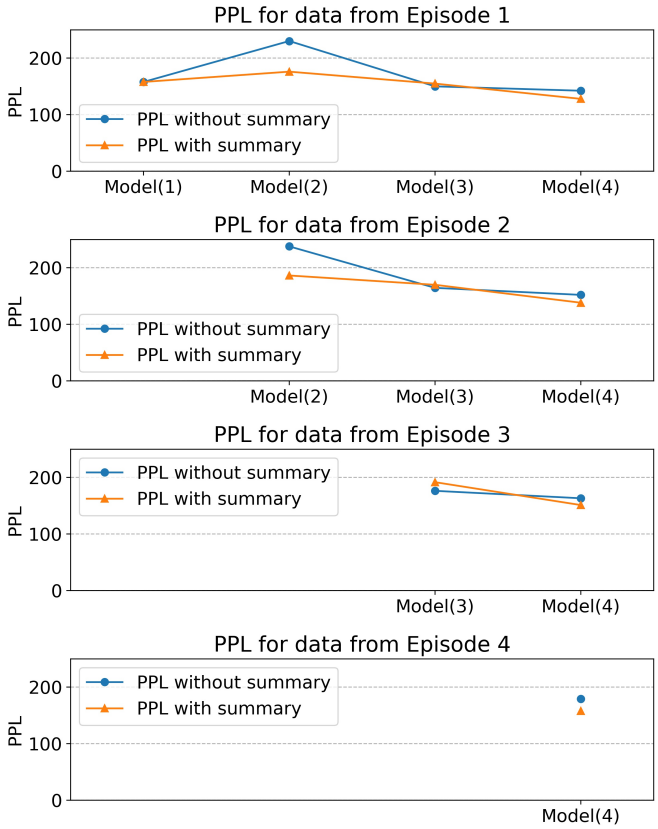

Fig. 6: Comparison between two training strategies in a streaming setting. Training with summary shows improved PPL, though there is no evidence of catastrophic forgetting. The lower the PPL, the better the model is performing. Historical Summary can improve performance up to $(230 - 170)/170 = 26\%$ (*Episode* 1's data after second *Episode*).

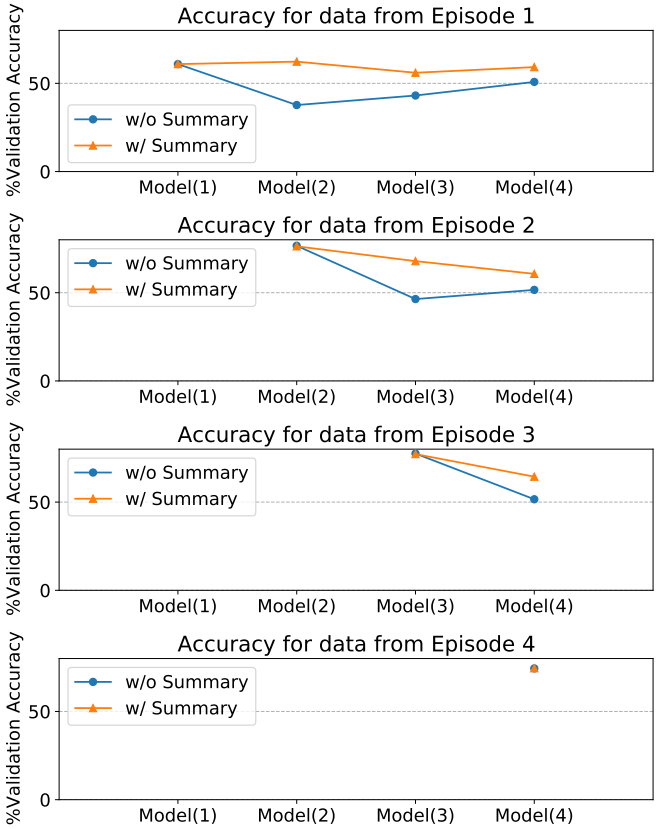

Fig. 7: Comparison between two streaming learning strategies. Classification accuracy on an *Episode*'s data gradually falls with incrementally trained models which provides evidence of catastrophic forgetting in a complex model like ResNet50 (Blue plots). *Historical Summary* mitigates the catastrophic forgetting to some extent (Orange plots).

REFERENCES

[1] A. Gepperth and B. Hammer, "Incremental learning algorithms and applications," in *ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2016.

[2] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks," *arXiv preprint arXiv:1312.6211*, 2013.

[3] B. Pfülb and A. Gepperth, "A comprehensive, application-oriented study of catastrophic forgetting in dnns," *arXiv preprint arXiv:1905.08101*, 2019.

[4] M. Mermillod, A. Bugaiska, and P. Bonin, "The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects," *Frontiers in psychology*, vol. 4, p. 504, 2013.

[5] H. He, S. Chen, K. Li, and X. Xu, "Incremental learning from stream data," *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 1901–1914, 2011.

[6] P. Junsawang, S. Phimoltares, and C. Lursinsap, "Streaming chunk incremental learning for class-wise data stream classification with fast learning speed and low structural complexity," *PloS one*, vol. 14, no. 9, p. e0220624, 2019.

[7] D. Sahoo, Q. Pham, J. Lu, and S. C. Hoi, "Online deep learning: Learning deep neural networks on the fly," *arXiv preprint arXiv:1711.03705*, 2017.

[8] A. Besedin, P. Blanchart, M. Crucianu, and M. Ferecatu, "Evolutive deep models for online learning on datastreams with no storage." in *IOTSTREAMING@ PKDD/ECML*, 2017.

[9] R. Kemker, M. McClure, A. Abitino, T. L. Hayes, and C. Kanan, "Measuring catastrophic forgetting in neural networks," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[10] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[11] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE transactions on systems, man, and cybernetics, part C (applications and reviews)*, vol. 31, no. 4, pp. 497–508, 2001.

[12] A. Gepperth and C. Karaoguz, "A bio-inspired incremental learning architecture for applied perceptual problems," *Cognitive Computation*, vol. 8, no. 5, pp. 924–934, 2016.

[13] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[14] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[15] Y. Kant, "Elastic weight consolidation," https://github.com/yashkant/Elastic-Weight-Consolidation, 2017.

[16] A. Koloskova, S. U. Stich, and M. Jaggi, "Decentralized stochastic optimization and gossip algorithms with compressed communication," in *ICML 2019 - Proceedings of the 36th International Conference on Machine Learning*, vol. 97. PMLR, 2019, pp. 3479–3487. [Online]. Available: http://proceedings.mlr.press/v97/koloskova19a.html

[17] A. Koloskova*, T. Lin*, S. U. Stich, and M. Jaggi, "Decentralized deep learning with arbitrary communication compression," in *ICLR 2020 - International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=SkgGCkrKvH

[18] A. Sergeev and M. D. Balso, "Horovod: fast and easy distributed deep learning in TensorFlow," *arXiv preprint arXiv:1802.05799*, 2018.