On Speeding Up the Training of Deep Neural Networks Using the Streaming Approach: The Base-Values Mechanism

Mateusz Wojtulewicz* WOJTULEWICZ@AGH.EDU.PL
Center of Excellence in Artificial Intelligence, AGH University, Krakow, Poland

Piotr Duda
Czestochowa University of Technology, Czestochowa, Poland
Faculty of Computer Science, AGH University, Krakow, Poland

Robert Nowicki
Czestochowa University of Technology, Czestochowa, Poland

Leszek Rutkowski
Center of Excellence in Artificial Intelligence, AGH University, Krakow, Poland
Systems Research Institute of the Polish Academy of Sciences, Warsaw, Poland

Editors: Manuel Grana, Pawel Ksieniewicz, Leandro Minku, Pawel Zyblewski

Abstract

Efficient and stable neural network training is crucial for advancing machine learning applications. This study explores the promising streaming approach as an alternative to traditional epoch-based methods. This paradigm shift involves transforming training data into a continuous stream, prioritizing challenging examples to enhance the learning process. Building upon this approach, we introduce an innovative Base-Values mechanism aimed at further improving the speed and stability of the streaming training process. We apply this framework to the original streaming training algorithms, resulting in algorithms such as Persistent Loss-Based (PLB) and Persistent Entropy-Based algorithm (PEB). We conduct a comprehensive experimental comparison on the EMNIST dataset, analyzing traditional epoch-based methods and the streaming approach, including both original and new methods employing the Base-Values mechanism. The exploration of various hyperparameters, including pre-training length, mini-batch size, and learning rate, provides valuable insights into their impact on the performance and stability of those neural network training methods. The results demonstrate the superior performance of a novel streaming algorithm that incorporates the Base-Values mechanism compared to both a traditional epoch-based method and other methods.

Keywords: Data stream, High dimensional data

1. Introduction

A shortening and stabilization of the learning process for artificial neural networks are tasks of major importance. Given the continued expansion of training sets and the development of increasingly complex models, no further justification is required. Researchers strive to achieve these objectives through various approaches.

Currently, the most popular approach focuses on convergence optimization of the Stochastic Gradient Descent (SGD) algorithm. The most widely adopted method for this purpose is Adaptive Moment Estimation (ADAM) (Kingma and Ba, 2015; Nanni et al., 2021).

* Corresponding author.

Although it has gained substantial popularity, ongoing efforts persist to enhance its performance and explore alternative methodologies (Gidel et al., 2019; Karabayir et al., 2021; Yun, 2023). In the literature, various techniques exist that aim to enhance the training process, such as active learning (Liu et al., 2022; Ren et al., 2021; Khosla et al., 2023) and continual learning (Parisi et al., 2019; Wang et al., 2024; Wu et al., 2023).

The aforementioned approaches have been investigated for several years and exhibit certain limitations. However, in recent years, the streaming approach has been proposed and has emerged as a promising methodology capable of concurrently achieving both the shortening and stabilization of the learning process, see (Duda et al., 2020) and (Duda et al., 2024). This approach is considered one of the most promising approaches that have to be addressed in future research (Borisov et al., 2022). It diverges from the conventional epoch-based data processing by creating a continuous stream of training data fed to the neural network. The method of drawing data elements into the stream aims to sample more challenging examples at higher frequencies, thereby intensifying the impact of each processed batch of data on the learning process.

In this paper, we introduce a mechanism for transforming non-stream data into an artificially designed data stream based on the assessment of how much they are challenging during the learning process. Furthermore, we analyze various data-drawing methods, with particular emphasis placed on the Persistent Entropy-Based (PEB) algorithm, which utilizes the entropy function as a measure of uncertainty. Our experimental study systematically compares traditional epoch-based methods with the most effective streaming approach algorithms. The results provide evidence of the performance enhancements brought about by the proposed methods.

The main novelties and characteristics of our approach can be summarized as follows:

1. We propose epoch-free learning: instead of processing the entire training set multiple times, the proposed method generates data streams from the original dataset. This departure signifies a fundamental shift in optimizing deep neural network learning, eliminating the need for repetitive passes over entire training sets. In our approach the original data are transformed into a streaming sequence, which is then fed into the neural network.

2. We propose the so-called Base-Values Mechanism - an innovative mechanism introduced to improve the speed and stability of the streaming training process. The concept behind PLB and PEB is to prioritize challenging data elements throughout the learning process, e.g., those associated with higher loss function values. In other words, the suggested methods increase the likelihood of selecting such elements for inclusion in subsequent batches.

3. The proposed algorithms outperform traditional epoch-based methods, even when using significantly fewer batches of data. More specifically, the PLB and PEB demonstrate improved speed, stability, and generalization capabilities compared to traditional epoch-based methods.

4. The introduced approach is not confined to specific deep structures but is adaptable to a broad class of deep neural network architectures, as well as other machine learning structures.

5. Our approach has been tested through extensive simulations on a well-known benchmark dataset, i.e. the EMNIST. The results confirm its high effectiveness. Importantly, our approach demonstrates superior accuracy over previous methods even when processing a substantially smaller number of batches. Specifically, the PEB method achieves, on average, approximately 2.5% higher final accuracy on the training dataset, and about 2% higher on the validation dataset, as compared to the MB (Mini-Batch) baseline.

The rest of the paper is organized as follows. In Section 2, the current advancements in various training techniques are presented. Following that, Section 3 introduces the innovative Base-Values Mechanism along with the new PLB and PEB algorithms. Subsequently, Section 4 outlines the research methodology, details the experimental procedure and provides a comprehensive analysis of the results. Finally, in Section 5, we draw conclusions, discuss key findings, and explore potential future developments in this domain.

2. Related works

The training process of a neural network involves two key steps: forward propagation and backward propagation. In forward propagation, the network processes input data through sequential layers, producing the output from its final layer. This output is compared to the target value using the loss function. Then, during the backward propagation, the algorithm calculates gradients for each weight in the network using the chain rule of calculus. Subsequently, these gradients are used to update the weights based on an optimization algorithm, such as SGD or one of its numerous variants, such as AdaGrad (Duchi et al., 2011), AdaDelta (Zeiler, 2012), or Adam (Kingma and Ba, 2015). A comprehensive comparison of different variants of the ADAM optimizer is presented in the paper (Nanni et al., 2021), where the authors evaluate optimizer effectiveness using medical images.

In all of the aforementioned procedures, the step size is mainly determined by the gradient value. Specifically, if the error is close to zero, the learning step becomes negligible. Therefore, the selection of data for gradient computation during network training is highly significant.

This problem was independently noticed by a group of researchers, leading to various approaches for selecting the most important elements. An approach assuming the calculation of the gradient based only on selected elements is proposed in (Jiang et al., 2019). This method requires performing a forward propagation phase on each element and then deciding whether to include this element in the gradient estimation. The decision to include or reject an element is not deterministic but random.

In (Thao Nguyen et al., 2024), the authors proposed a method for hiding the least-important samples during training. This paper suggests the use of specific scheduling to gradually reduce the maximum sample hiding fraction as the number of epochs increases. A specific importance sampling strategy is considered in (Zeng et al., 2021). The method, called Mercury, is designed for embedding systems. It consists of three stages: a group-wise importance computation and sampling scheme, an importance-aware data resharding scheme, and a bandwidth-adaptive computation-communication schedule.

In order to move away from sequential data processing, (Duda et al., 2020) proposed identifying the training set with the data distribution from which elements will be drawn for

network training. Based on this distribution, a data stream is created to feed the learning process with data, considering their suitability for training. The authors propose three variants for determining the usefulness of an element. Additionally, they note that the usefulness of elements may change over time, and therefore the distribution of data from which the elements are selected also changes. To detect such changes, they proposed using a drift detector that tracks the quality of the network's operation on elements not seen during training. This work was further developed in (Duda et al., 2024).

3. Novel Streaming Approach with Base-Values Mechanism

In this section a method of training neural networks using the Base-Value Mechanism is presented. The method, similar to the method presented in (Duda et al., 2020) and (Duda et al., 2024), creates a stream of data. However, it completely changes the way elements are selected. The Base-Values introduce a more reliable measure of the importance of individual training elements. As a result, the obtained data distribution better reflects the current usefulness of elements and may allow abandoning the need to track changes in network operation, which was the crucial step the method proposed in (Duda et al., 2020).

For every element in a training dataset $D = \{(x_i, d_i) : i = 1, \ldots, N\}$, a positive real value $b_i \in \mathbb{R}_+$, referred to as a *Base-Value* is attached. This constructs a set of triplets $\mathcal{T}_D$:

$$\mathcal{T}_D = \{(x_i, d_i, b_i) : i = 1, 2, \ldots, N\}. \tag{1}$$

The Base-Value serves as a measure of the network's uncertainty or difficulty for a particular training element, indicating its importance in the learning process. More specifically, the higher the Base-Value, the more challenging the element is, and thus the more significant it is for the learning process. Importantly, the Base-Values establish a partial order for the training dataset D, providing a reliable mechanism for determining the relative importance of its element at any time.

However, the resulting set $\mathcal{T}_D$ does not share the properties of the statistical population, i.e., the Base-Values do not form a probability mass function for the elements in the training dataset, as it is not constrained that they sum up to 1. Therefore, sampling from $\mathcal{T}_D$ is redefined as sampling from a probability mass function that is directly proportional to the Base-Values. More precisely, the corresponding probability mass function values $\{p_i : i = 1, 2, \ldots, N\}$ are calculated as normalized Base-Values:

$$p_i = \frac{b_i}{\sum_{j=1}^{N} b_j}. \tag{2}$$

Now, based on the population, a stream S_D of length t is defined as a sequence:

$$\begin{aligned} S_D = (s_1, s_2, \ldots, s_t : s_j = (x_{i_j}, d_{i_j});\ P(i_j = k) = p_k; \\ k, i_j \in \{1, 2, \ldots, N\};\ j = 1, 2, \ldots, t). \end{aligned} \tag{3}$$

To clarify, each s_j is a training example from D, sampled independently from the population created based on $\mathcal{T}_D$ and formula (2).

Unlike traditional methods, the streaming learning process is not divided into epochs, but is composed of a series of successive steps, also called iterations. Each iteration involves constructing a new mini-batch B_τ comprising n elements recently taken from the stream:

$$B_\tau = (s_{t-n+1}, s_{t-n+2}, \ldots, s_t), \tag{4}$$

where $\tau = 1, 2, \ldots, T$ denotes the current step, and $t = \tau \cdot n$ is the total number of drawn elements. Following this, the training proceeds in a conventional way. The mini-batch is processed by the network in a forward pass, which is followed by the computation of loss values. Subsequently, the backpropagation phase is initiated to compute the gradient of the objective function with respect to the weights. Depending on the chosen optimization algorithm, the update of the weights is then executed. At the end of the iteration, the Base-Values for the training elements of the considered mini-batch are updated, which influences the elements sampled into the following mini-batch.

To clarify, a step *Sample a new mini-batch B_τ from $\mathcal{T}_D$* used later in the pseudo-code signifies, firstly, calculating the probability mass function values p_i by normalizing Base-Values b_i using the formula (2), and subsequently independently sampling n elements according to this distribution to construct a new mini-batch B_τ.

The Base-Values can be seamlessly integrated with many proposed in literature measures of the usefulness of data elements. Here, we propose two methods. The newly introduced ones are labeled as Persistent Loss-Based (PLB), and Persistent Entropy-Based (PEB). The pseudo-code for these methods is provided in Algorithm 1. The difference between these algorithms lies in the process of updating Base-Values in lines 3 and 9 (using formulas (5) for PLB and (6) for PEB).

Algorithm 1: The PLB/PEB streaming training algorithm

Input : D: Training dataset of size N,
n: Size of the mini-batch,
T: Total number of iterations,
L: The loss function

Construct a set of triplets $\mathcal{T}_D$ from D `// Initializing Base-Values in` $\mathcal{T}_D$
foreach $i = 1, 2, \ldots, N$ **do**
  Initialize b_i according to formula (5) or (6)
end
for $\tau = 1, 2, \ldots, T$ **do**
  Sample a new mini-batch B_τ from $\mathcal{T}_D$ according to the probability mass function derived from formula (2) Train the network on the mini-batch B_τ `// Updating Base-Values`
  foreach *instance* s_j *in* B_τ **do**
    Update b_{i_j} according to formula (5) or (6)
  end
end

3.1. Persistent Loss-Based Approach (PLB)

The Persistent Loss-Based approach (PLB) can be applied to both classification and regression tasks. It uses the loss function L employed in the training process to measure the importance of an example. Consequently, Base-Values are calculated as:

$$b_i = L(x_i, d_i), \tag{5}$$

In the PLB approach, the Base-Values are initialized as the loss value calculated for each training example. During training, the Base-Values for each element in the mini-batch are updated at each iteration, reflecting the newly calculated loss values after the training step.

It is worth considering how accurately the Base-Values reflect the actual distribution of loss values. The quality of this estimation can be influenced by several factors, including the size of the mini-batch, the rate of convergence, and the nature of the task. Possibly, due to most of the loss values being outdated, the estimation may be skewed. Specifically, the Base-Values might underestimate the real loss values, as they were calculated many iterations ago. Consequently, if the difference is significant, the likelihood of sampling such an element increases, leading to its update. This process results in sparse updates of Base-Values, which enhances the accuracy of the real estimation.

3.2. Persistent Entropy-Based Approach (PEB)

The Persistent Entropy-Based (PEB) approach is suitable only for classification problems. This is because it uses the entropy of the softmax values from the network's final classification layer as a measure of uncertainty.

In the context of the PEB approach, the uncertainty of a training example is measured by the entropy calculated from the softmax applied to the activations of the network's final classification layer for that example. In simpler terms, if y_i represents the K-dimensional output vector from a neural network classifier $f(\,\cdot\,;\theta) : \mathbb{R}^m \to \mathbb{R}^K$ for the m-dimensional input vector x_i, the Base-Value for this element is computed as:

$$b_i = H(\sigma(y_i)), \tag{6}$$

where $\sigma(y_i)_j = \frac{\exp y_{ij}}{\sum_{k=1}^{K} \exp y_{ik}}$ for $j = 1, 2, \ldots, K$, denotes the softmax function, and $H(\cdot)$ is the discrete version of the well-known Shannon's entropy (Shannon, 1948).

To clarify, the examples that the network is uncertain about (i.e., those for which the softmax probabilities are spread out over many classes) will have higher entropy and, thus, higher Base-Values. On the other hand, examples that the network is certain about (i.e., those for which the softmax probabilities are concentrated into one class) will have smaller entropy and thus lower Base-Values.

The choice of entropy over the cross-entropy loss as a measure of uncertainty is based on the fact that it results in more diversity in the Base-Values. This is because the cross-entropy loss value is computed based only on one activation (for a true class), while the entropy is calculated using all the activations, thus providing more meaning. We can expect

the network to lower cross-entropy values, as this is the usual loss function employed in classification tasks, therefore it only tries to increase the true class activation. As a result, the other activations may be spread arbitrarily, which provides diversity in the entropy values.

4. Experimental Results

The primary objective of the main experimental study was to compare the performance of the traditional mini-batch approach with the streaming approach in a well-known image classification benchmark problem under a range of conditions. These conditions were defined by different sets of hyperparameters, allowing us to explore the impact each individual hyperparameter has on the learning process and to evaluate the robustness of the methods to those hyperparameters.

For a defined set of hyperparameter values, the experiment was designed in two stages: a pre-training phase and a fine-tuning phase. The pre-training phase followed the traditional mini-batch approach, serving as a foundation and starting point for the subsequent training. The idea behind the pre-training phase was to prevent potentially chaotic behavior during the initial training steps, allowing for accurate initialization of both Base-Values.

The fine-tuning phase evaluates the performance and effectiveness of three different data processing methods. These methods included the traditional mini-batch approach (MB), serving as a baseline for the comparison, the Loss-Based (LB) approach, recognized as the optimal method for data sampling in (Duda et al., 2020), and the Persistent Entropy-Based (PEB) approach, identified as the most effective novel technique leveraging the mechanism of Base-Values.

The metrics used for the evaluation were the value of the loss function and accuracy throughout the entire learning process, calculated on both the training dataset and the test dataset. Each instance of the second phase was executed 12 times. This repetition was designed to collect sufficient data for a robust analysis, providing a more reliable comparison between the methods. Furthermore, reproducibility was strictly maintained throughout the experiment to ensure consistency in the results.

4.1. Experimental Procedure

For each of the varying hyperparameters, a complete experiment was conducted for each of its potential values, while other hyperparameters were set to their default values, allowing us to comprehensively analyze their individual effects. Those hyperparameters were the number of pre-training epochs, the mini-batch size, and the learning rate. The number of pre-training epochs determines the length of the pre-training phase, which influences the stability of the calculated probability values but potentially postpones the improvements introduced by the streaming approach. The possible values for this hyperparameter were 0, 1, 2 and 5, with a default value of 2. The mini-batch size defines the number of examples sampled in each iteration, which impacts the sparsity of the probability updates. Its possible values were 4, 32 and 1024, with 32 as the default. The learning rate, which controls the update step at each iteration and affects the learning variance, had possible values of 0.1, 0.05 and 0.01, with 0.05 as the default.

The dataset used for the experiments is the EMNIST dataset, a commonly used benchmark in image classification (Cohen et al., 2017). The dataset serves as a more challenging alternative to the original MNIST dataset. The EMNIST dataset consists of 124800 training examples and 20800 test examples. Each example is a 28×28 grayscale image representing a handwritten letter from one of 26 classes in the EMNIST dataset. The dataset is balanced, which means that the examples are evenly distributed among existing classes. The EMNIST dataset was selected for this study due to being a well-established benchmark in the field, offering a higher level of complexity compared to the standard MNIST dataset.

The Convolutional Neural Network (CNN) used in the experiments is a straightforward five-layer model, mirroring the one used in the previous streaming approach studies in (Duda et al., 2020). It begins with two convolutional layers, with 32 and 64 filters of size 3×3, respectively. This is followed by a max pooling layer with a 2×2 filter that reduces the spatial dimensions of the output from the convolutional layers. To prevent overfitting, a dropout mechanism with a probability of 0.25 is then applied. The output is subsequently flattened to a single dimension and fed into a fully connected layer containing 128 neurons. Another dropout mechanism, this time with a rate of 0.5, is applied before the final fully connected layer, which consists of 26 neurons, each representing a class label. Except for the final classification layer, all layers employ a ReLU activation function. The network is composed of approximately 1.2 million trainable parameters.

The experiments used the Adadelta optimization algorithm, with the learning rate being one of the varying hyperparameters, and the γ set to 0.95.

The experiments were implemented in Python using the PyTorch library (Paszke et al., 2019). The computations were performed on a machine equipped with an AMD Ryzen 5 4600H @ 3.0 GHz CPU, an NVIDIA GeForce GTX 1650 GPU and 32 GB of RAM. The metrics were tracked using the Weights & Biases library (Biewald, 2020). The code used in the experiments is available upon request.

4.2. Results and Analysis

In the subsequent sections, the impact of each of the varying hyperparameters will be discussed. The trajectories of metrics throughout the learning process for different experiments are plotted, and used in the analysis. Figures containing the plotted metrics share a predefined template.

Plots in each figure are organized into a grid structure, with columns corresponding to specific experiments, and rows to different metrics. These metrics include accuracy on the training dataset (acc/train), accuracy on the validation dataset (acc/val), cross-entropy loss on the training dataset (loss/train), and cross-entropy loss on the validation dataset (loss/val).

Each line in the plot shows the mean value of the corresponding metric, averaged across all repetitions of the experiment. The colored region surrounding each line illustrates the range within two standard deviations of the mean. The x-axis of the plots represents the number of iterations, expressed in terms of epochs, as calculated in the traditional mini-batch method. This allows for a clear comparison between methods.

4.2.1. Comparison of the Persistent Methods

In order to identify the most effective Persistent method using the Base-Values mechanism for further analysis, a comprehensive experiment was conducted with default hyperparameters. The pretraining phase spanned 2 epochs, both phases employed a batch size of 32 and a learning rate was set to 0.05. Each fine-tuning phase run was repeated 12 times. The metric trajectories are illustrated in Fig. 1.

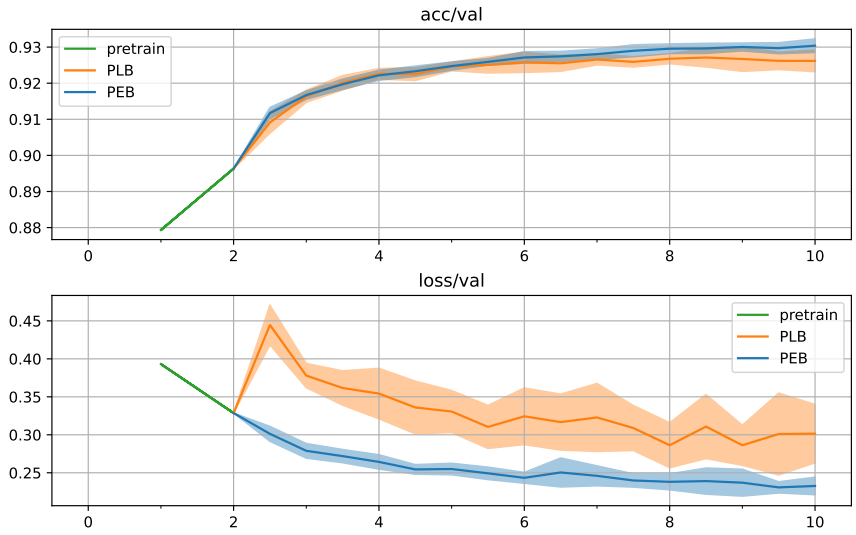

Figure 1: The comparison of metric trajectories for the methods utilizing the Base-Values mechanism.

Notably, the PEB method consistently outperforms the PLB method in both accuracy and loss metrics on the validation dataset. The PEB method maintains greater stability in the loss metric trajectory and achieves significantly lower loss values immediately after the pre-training phase and at the end of the training process. Although both methods show similar accuracy at the beginning of the fine-tuning phase, the PEB method ultimately reaches a higher final accuracy. This comparison establishes the PEB method as the most effective among both Persistent methods, justifying its selection for subsequent experiments.

4.2.2. Impact of the Pre-training Length

The number of pre-training epochs, which dictates the length of the pre-training phase, can have several effects. The initial training steps of the network tend to be chaotic, as the gradient of the objective function is significant for most of the randomly initialized weights. Therefore, the traditional, more systematic approach that ensures each training example is used repeatedly seems to be a rational choice.

Moreover, as the steepness of the objective function decreases and the weight updates become less impactful, the calculated and then updated PoD values are expected to stabilize. However, extending the pre-training phase could delay the potential enhancements brought by the streaming approach.

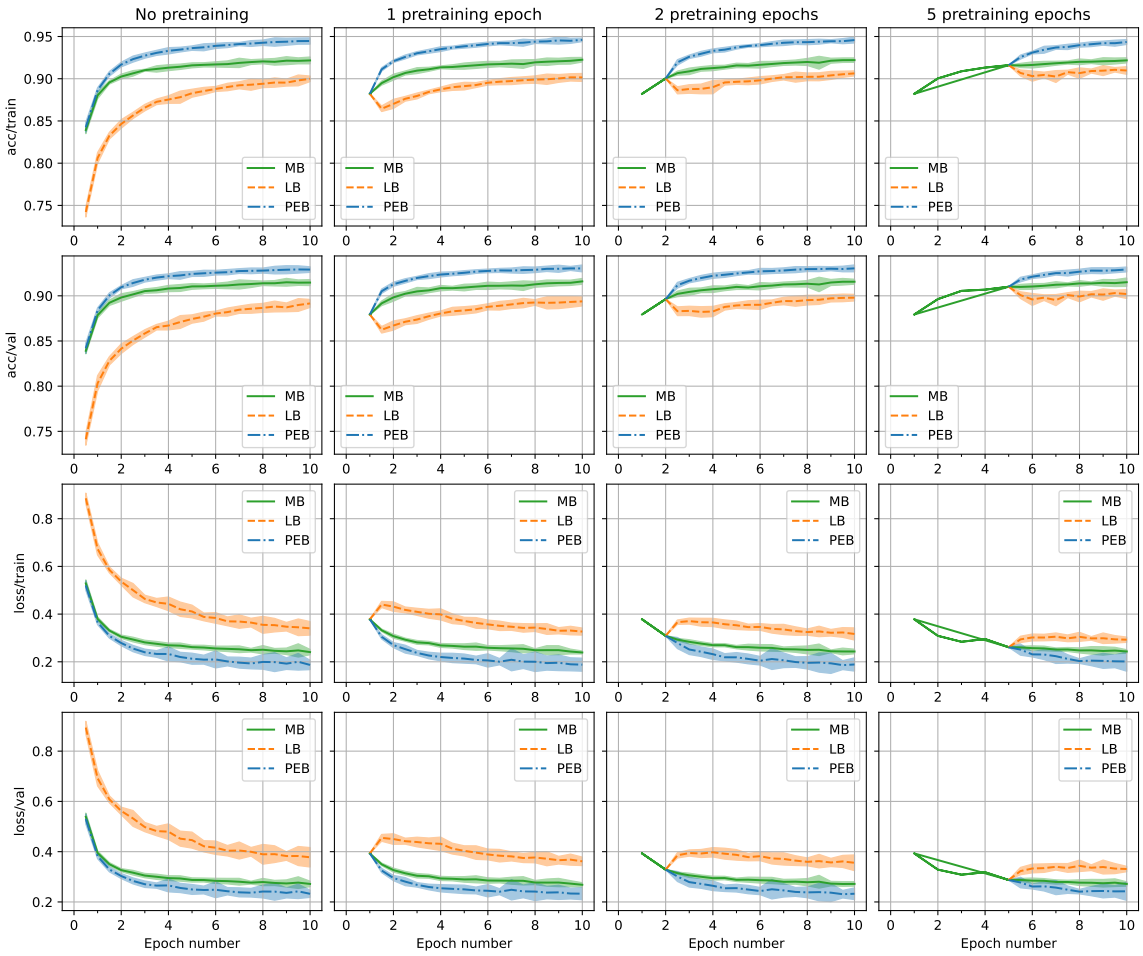

Figure 2: The side-by-side comparison of metric trajectories for experiments with 0, 1, 2 or 5 pre-training epochs on the EMNIST dataset.

Given these considerations, it can be hypothesized that there exists an optimal length for the pre-training phase. After this point, the streaming approach is likely to provide the most stable and consistent improvement. By conducting experiments with 1, 2 or 5 pre-training epochs, as well as without any pre-training phase, we can test and address this hypothesis. The metric trajectories for those experiments on the EMNIST dataset are presented in Fig. 2. Based on the observations from its plots, several key insights can be drawn regarding the performance of data sampling methods across varying lengths of the pre-training phase.

Firstly, it is very noticeable that the PEB approach outperforms all other methods in terms of final accuracy and loss values. Specifically, it achieves, on average, approximately 2.5% higher final accuracy on the training dataset, and about 2% higher on the validation dataset, as compared to the MB baseline. In terms of the loss function, the PEB approach shows a decrease of 0.5 on the training dataset and 0.2 on the validation dataset compared

to the baseline. Furthermore, the areas of the two standard deviations away from the mean for the PEB approach and the baseline do not overlap on any plot, showing a consistent improvement. This superior performance of the PEB approach suggests that the utilization of the Base-Values is highly effective for this problem.

Secondly, the LB method displays performance close to the MB baseline in terms of accuracy on both datasets. However, its loss functions are significantly worse, especially on the validation dataset. This divergence between accuracy and loss performance is somewhat unexpected, yet it indicates the effectiveness of the streaming approach. The higher loss function values, yet at the same time improved accuracy, suggest that the LB method has successfully traded off a higher overall loss function to increase the network's ability to correctly classify a greater number of examples.

Interestingly, for all the methods the final results are very similar across all experiments using the same dataset. However, the earlier the pre-training phase is finished, the steeper the trajectories are, particularly for the PEB approach. This leads to quicker enhancement of the network's performance, suggesting that the most effective strategy might be to omit the pre-training phase entirely, at least for the studied problem. This is especially true for the PEB approach, which shows immediate improvements at the beginning of the fine-tuning phase in all metrics compared to the MB baseline approach. That is not the case for the LB approach, which does not observe such significant improvements.

4.2.3. Effect of the Mini-Batch Size

The mini-batch size is a key hyperparameter that determines the number of elements sampled in each iteration. This sampling can be either deterministic, as in the traditional methods, or random, as in the streaming approach. The size of the mini-batch affects the accuracy and variance of the gradient estimation and, for the streaming approach, the sparsity of the Base-Values updates. Broader updates, which occur with larger mini-batches, result in stored values that are closer to the true distribution. However, larger mini-batches can slow convergence down and increase the likelihood of the learning process becoming trapped in local minima. As such, the careful selection of the mini-batch size is crucial to the learning process, especially for streaming approaches.

The analysis of Fig. 3, which presents the results of experiments with varying mini-batch sizes, provides a deeper understanding of the impact of this hyperparameter, not only on the learning process itself but also on the distinct sampling methods.

The first observation refers to the speed of optimization convergence. As anticipated, the lower the mini-batch size, the faster the changes in the learning process. This is caused by the higher frequency of the weights' updates, which accelerates the convergence in the initial iterations.

However, a closer look at the results of the experiment with a mini-batch of size 4 reveals a slight overfitting effect for the baseline MB method. This is indicated by a small increase in the loss values and a decrease in the accuracy values for the validation dataset. Interestingly, this is not observed in the results of the PEB streaming approach. In this case, the validation metrics, after achieving the best values, plateau and maintain it steadily without a decrease in performance. The LB approach also does not show any typical signs of overfitting. However, loss values for this approach are considerably larger, from the very beginning of the fine-tuning phase, as compared to the other methods.

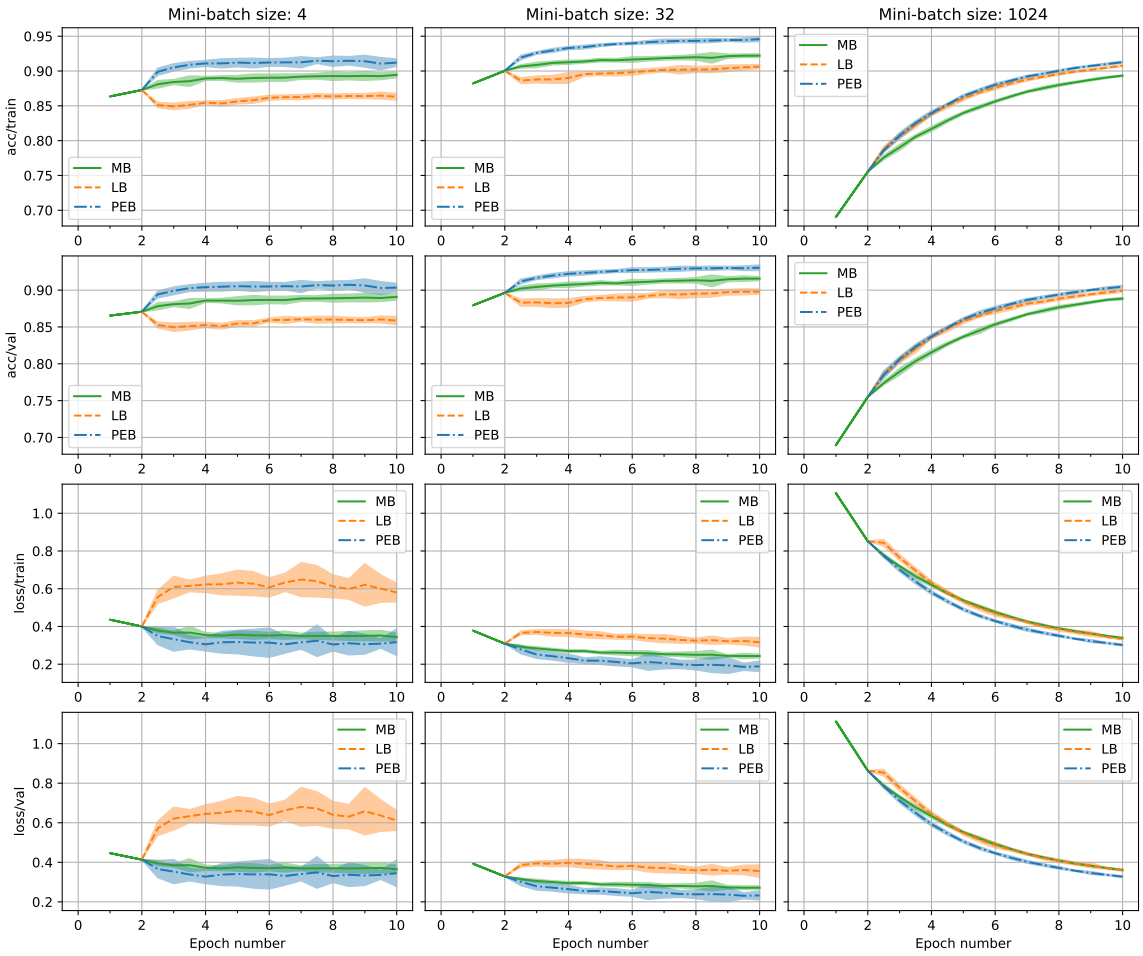

Figure 3: The side-by-side comparison of metric trajectories for experiments with mini-batch size of 4, 32 or 1024 on the EMNIST dataset.

The PEB method outperforms other methods for both mini-batch sizes of 4 and 32. However, when the mini-batch size increased to 1024, all methods showed similar performance, as indicated by all metrics. In this setting, the LB method achieves the highest accuracy on both datasets, while maintaining similar loss values. Both streaming approaches improve slightly over the baseline. At the same time, the PEB method employing the mechanism of Base-Values presents a stable improvement over all of the mini-batch sizes. This suggests that the use of Base-Values contributes to a more accurate stored distribution of the training elements' challenge levels.

The best performance after 10 epochs is obtained by the PEB method with a mini-batch size of 32. In this variant, the accuracy on the validation dataset improves over the baseline by more than 2%. Additionally, this setting demonstrates the biggest stability of the learning process, as indicated by the variance over the experiment repetitions. However, the learning process for experiments with mini-batch of sizes 32 and 1024 is far from finished;

therefore to assess which method achieves the best final performance, longer tests have to be executed.

4.2.4. Influence of the Learning Rate

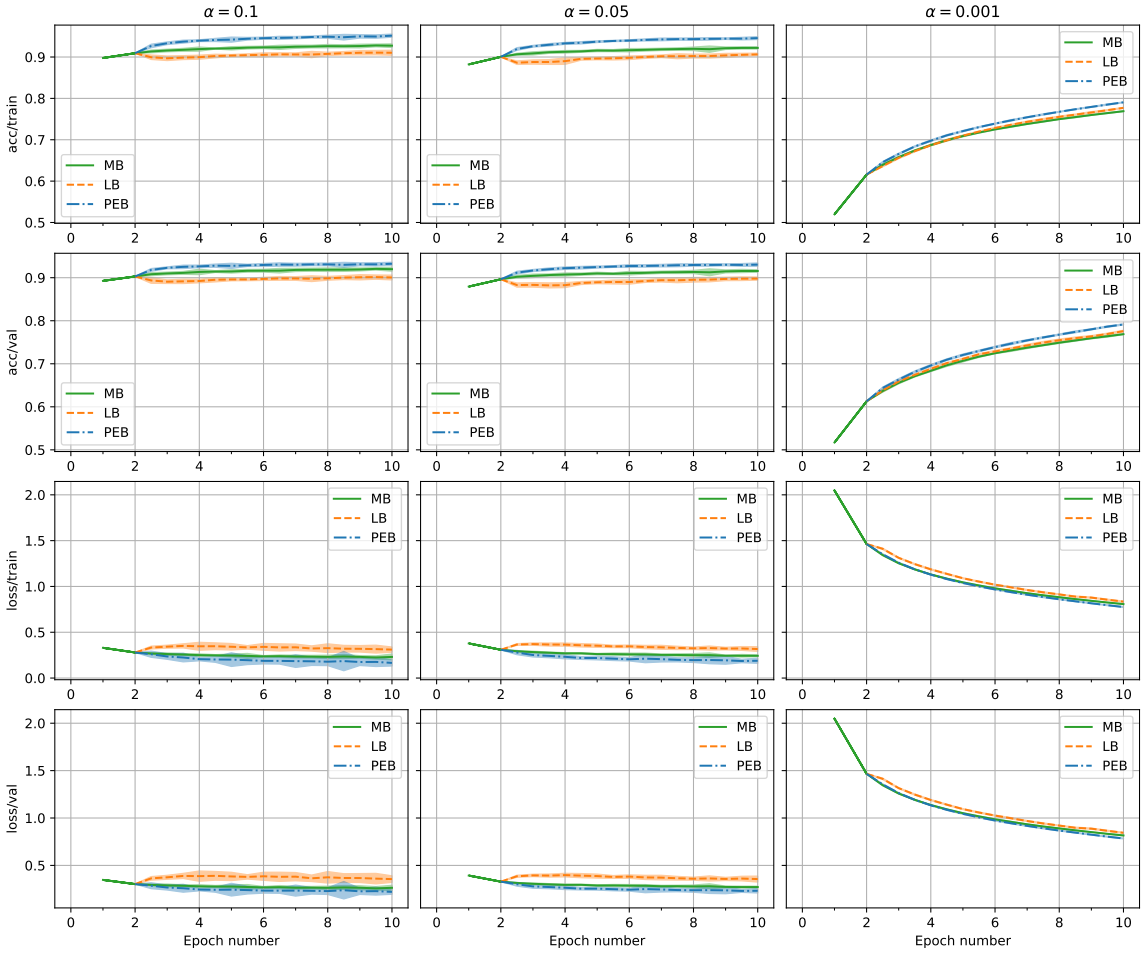

Figure 4: The side-by-side comparison of metric trajectories for experiments with different learning rate, denoted as α, on the EMNIST dataset.

The learning rate is a crucial hyperparameter that dictates the magnitude of weight updates at each iteration of the learning process. It significantly influences the speed and stability of the optimization convergence. In the streaming approach, the learning rate also affects the variance of the Base-Values. Larger updates, which occur with a higher learning rate, can result in less accurate stored distributions, as elements updated a while ago are likely to have significantly different values at present. On the contrary, too small learning rate can slow down the optimization process, potentially leading to a worse final model.

Figure 4 presents the results of experiments with different learning rate values. The analysis of these trajectories brings valuable insights into the influence of the learning rate value on each individual sampling method.

Firstly, as expected, the lower the learning rate, the slower the convergence. However, a smaller learning rate also results in lower variance, which can lead to a more stable model.

For the highest learning rate (0.1), the PEB method outperforms the baseline, but not as visibly as in the experiment with the middle valued learning rate (0.05). The visible gap between metric values is covered by the wider variance areas, which is an effect of less stable trajectories for both methods. In the same two experiments, the LB method shows comparable results with the baseline in terms of metrics calculated on the training dataset, but performs worse on the validation dataset, which is especially apparent analyzing the loss values.

When the learning rate is smallest, i.e. set to 0.001, the variance for all the metrics across all the sampling methods is the smallest compared to other experiments. In this setting, both streaming methods show higher accuracy on both datasets, while maintaining very comparable loss values, with the PEB method slightly outperforming the rest. However, the learning process is again far from finished in this setting, suggesting running longer experiments for the model to fully converge. The far better performance and less diverging metric trajectories observed for the LB approach in the lowest learning rate setting could be a result of smaller and less drastic updates of the distribution, which in turn leads to a more stable learning process.

The best performance is achieved by the PEB approach in the experiments with the highest learning rate. However, the values in this setting are much less stable compared to other settings, indicating that while the higher learning rate can lead to the fastest, very good performance, it can also lead to more instability in the learning process.

5. Conclusions

The results and analysis presented in this study provide a comprehensive understanding of the impact of various hyperparameters on the performance of both traditional and streaming training methods. The exploration of different pre-training lengths, mini-batch sizes, and learning rates has revealed key insights into the behavior and effectiveness of these methods under varying conditions.

The Persistent Entropy-Based (PEB) streaming approach consistently outperformed the traditional mini-batch (MB) approach and the Loss-Based (LB) streaming approach across most experiments. This superior performance suggests that the utilization of Base-Values in the PEB method is highly effective for the chosen problem. The PEB method demonstrated higher accuracy and lower loss values throughout the entire learning process, introducing immediate improvements over the traditional MB baseline, no matter the length of the pre-training.

In conclusion, this study demonstrates the potential of streaming training methods, particularly the one employing the Base-Values mechanism, in improving the performance and stability of the learning process. It also underscores the importance of carefully selecting and tuning hyperparameters to achieve optimal results.

It is worth noting that stored Base-Values are not accurate at all times, as the difficulty of the corresponding elements may have changed since the last update. Future research should delve into this discrepancy, examining its impact on the sampling process, and work towards developing methods to possibly mitigate this effect. Furthermore, exploring alternative metrics for assessing the difficulty level of training elements and examining potential alternative sampling methods could offer valuable insights for further advancements in this domain. Additionally, it is important to assess how these methods perform across a wider range of machine learning models and tasks, particularly on larger or more complex datasets, such as those with noisy labels. We are currently working on extending the framework to better handle problems involving noisy labels to evaluate its efficacy in such challenging scenarios.

Acknowledgments

We gratefully acknowledge the funding support by program "Excellence initiative–research university" for the AGH University in Krakow as well as the ARTIQ project: UMO-2021/01/2/ST6/00004 and ARTIQ/0004/2021.

References

Lukas Biewald. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

Vadim Borisov, Tobias Leemann, Kathrin Sessler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, page 1–21, 2022. ISSN 2162-2388. doi: 10.1109/tnnls.2022.3229161. URL http://dx.doi.org/10.1109/TNNLS.2022.3229161.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: an extension of mnist to handwritten. *Proceedings of the IEEE*, 4322, 2017.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61): 2121–2159, 2011. URL http://jmlr.org/papers/v12/duchi11a.html.

Piotr Duda, Maciej Jaworski, Andrzej Cader, and Lipo Wang. On training deep neural networks using a streaming approach. *Journal of Artificial Intelligence and Soft Computing Research*, 10(1):15–26, 2020. doi: doi:10.2478/jaiscr-2020-0002. URL https://doi.org/10.2478/jaiscr-2020-0002.

Piotr Duda, Mateusz Wojtulewicz, and Leszek Rutkowski. Accelerating deep neural network learning using data stream methodology. *Information Sciences*, 669:120575, 2024.

Gauthier Gidel, Francis Bach, and Simon Lacoste-Julien. Implicit regularization of discrete gradient dynamics in linear neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Angela H Jiang, Daniel L-K Wong, Giulio Zhou, David G Andersen, Jeffrey Dean, Gregory R Ganger, Gauri Joshi, Michael Kaminksy, Michael Kozuch, Zachary C Lipton, et al. Accelerating deep learning by focusing on the biggest losers. *arXiv preprint arXiv:1910.00762*, 2019.

Ibrahim Karabayir, Oguz Akbilgic, and Nihat Tas. A novel learning algorithm to optimize deep neural networks: Evolved gradient direction optimizer (evgo). *IEEE Transactions on Neural Networks and Learning Systems*, 32(2):685–694, 2021. doi: 10.1109/TNNLS.2020.2979121.

Savya Khosla, Chew Kin Whye, Jordan Ash, Cyril Zhang, Kenji Kawaguchi, and Alex Lamb. Understanding and improving neural active learning on heteroskedastic distributions. In *ECAI 2023*, page 1248–1255. IOS Press, 2023. doi: 10.3233/FAIA230402.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diega, CA, USA, 2015.

Peng Liu, Lizhe Wang, Rajiv Ranjan, Guojin He, and Lei Zhao. A survey on active deep learning: From model driven to data driven. *ACM Comput. Surv.*, 54(10s), 9 2022. ISSN 0360-0300. doi: 10.1145/3510414. URL https://doi.org/10.1145/3510414.

Loris Nanni, Gianluca Maguolo, and Alessandra Lumini. Exploiting adam-like optimization algorithms to improve the performance of convolutional neural networks. *arXiv preprint arXiv:2103.14689*, 2021.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

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

C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x.

Truong Thao Nguyen, Balazs Gerofi, Edgar Josafat Martinez-Noriega, François Trahay, and Mohamed Wahib. KAKURENBO: Adaptively hiding samples in deep neural network training. *Advances in Neural Information Processing Systems*, 36, 2024.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Lilei Wu, Zhen Wang, and Jie Liu. Adaptive self-supervised continual learning. In *ECAI 2023*, page 2680–2687. IOS Press, 2023. doi: 10.3233/FAIA230576.

Juyoung Yun. An efficient approach to mitigate numerical instability in backpropagation for 16-bit neural network training. *arXiv preprint arXiv:2307.16189*, 2023.

Matthew D. Zeiler. Adadelta: An adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.

Xiao Zeng, Ming Yan, and Mi Zhang. Mercury: Efficient on-device distributed dnn training via stochastic importance sampling. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, pages 29–41, 2021.