IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 9, 2022, accepted March 29, 2022, date of publication April 11, 2022, date of current version April 15, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3166158

Continual Learning With Speculative Backpropagation and Activation History

SANGWOO PARK AND TAEWEON SUH, (Member, IEEE)
Department of Computer Science and Engineering, Korea University, Seoul 02841, South Korea

Corresponding author: Taeweon Suh (suhtw@korea.ac.kr)

This work was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant by the Korean Government through Ministry of Science and ICT (MSIT) (Research on CPU vulnerability detection and validation) under Grant 2019-0-00533; in part by the National Research Foundation of Korea (NRF) Grant by the Korean Government through MSIT under Grant NRF-2022R1A2C1011469; and in part by Samsung Electronics Company Ltd., under Grant IO210204-08384-01.

ABSTRACT Continual learning is gaining traction these days with the explosive emergence of deep learning applications. Continual learning suffers from a severe problem called catastrophic forgetting. It means that the trained model loses the previously learned information when training with new data. This paper proposes two novel ideas for mitigating catastrophic forgetting: Speculative Backpropagation (SB) and Activation History (AH). The SB enables performing backpropagation based on past knowledge. The AH enables isolating important weights for the previous task. We evaluated the performance of our scheme in terms of accuracy and training time. The experiment results show a 4.4% improvement in knowledge preservation and a 31% reduction in training time, compared to the state-of-the-arts (EWC and SI).

INDEX TERMS Continual learning, lifelong learning, catastrophic forgetting, parallel training, speculative backpropagation, activation history, training accelerator, FPGA.

I. INTRODUCTION

In tandem with the explosive emergence of deep learning applications such as streaming services, E-commerce, and self-driving cars [1]–[4], continual learning is gaining more traction these days. Continual learning, also known as lifelong learning or online learning, means that the neural network is continuously trained for newly generated tasks and/or tasks from other domains without forgetting the knowledge obtained from the preceding tasks. It has already been applied in commercial applications. For example, Netflix and Amazon gather new data as people interact with their services and adjust the neural networks accordingly [1]. The self-driving system in Tesla continuously updates its Deep Neural Network (DNN) based on the data aggregated from its fleet of approximately 500,000 vehicles [2], [4].

Continual learning suffers from a severe problem called catastrophic forgetting [5]–[17], which means that the trained model loses its knowledge after being retrained with a new task. It is because a neural network is usually trained based on Stochastic Gradient Descent (SGD), which optimizes weights in the model only for the current task without considering the previous one. If all tasks are available beforehand, the weights in DNN can be jointly optimized for all tasks with training [16]. However, it is not feasible in real-world applications because data is continuously and dynamically generated as mentioned before. Revamping and completely retraining the model every time new data arrives, to cope with catastrophic forgetting, requires a lot of computational and storage costs. Accordingly, there are research efforts for addressing catastrophic forgetting in efficient and feasible ways [8]–[17]. Some studies [8], [9] regularly use the samples from previous tasks when training a new one. Some works [16], [17] take the selective approaches, which avoid drastic changes in the model parameters influencing a lot for the past tasks and allow the other parameters to change for accommodating the new tasks.

The DNN training requires a huge amount of processing time due to its complexity. For example, it takes 8.5 days for the Inception-BN model to train ImageNet by using four GeForce GTX 980 cards [48]. There are several products and studies [24]–[30] for speeding up the training with special hardware architectures. Google's Tensor Processing Unit (TPU) 3 contains two cores containing matrix multiply and vector processing unit, and 32 GiB memory with high bandwidth for the training acceleration [24]. Habana

The associate editor coordinating the review of this manuscript and approving it for publication was Marco Cococcioni.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

Labs, a startup called from Israeli, released a processor for training called Gaudi with eight AI-customized cores and a centralized matrix multiplication engine [25], [26]. Baidu has announced a low-power AI accelerator chip called Kunlun, operating in the data center for training [27], [28]. There are also hardware accelerators for continuous learning. Nanyang Technological University designed an FPGA accelerator for the streaming image classifier [34], [35]. The accelerator is capable of performing incremental class learning for object classification.

This paper proposes two novel ideas of enhancing the performance in terms of accuracy and training time in continual learning: speculative backpropagation (SB) and activation history (AH). The SB proposed in our prior work [36] enables the simultaneous operation of forward and backward propagations in training. The speculation is applied to the backpropagation based on past knowledge. We found that the SB helps preserve the previously learned information to alleviate catastrophic forgetting. The AH enables finding important weights based on the activation history from the previous tasks and discourages changes in those weights. The experiment shows that our approach provides a 4.4% better accuracy for the past tasks, compared to the state-of-the arts. The training accelerator implemented in hardware shortens the training time by up to 31%.

The rest of the paper is organized as follows: Section II introduces the related works for continual learning and for AI accelerators. Section III briefs the neural network training process. Section IV presents the proposed methods mitigating catastrophic forgetting: Speculative Backpropagation (SB) and Activation History (AH). Section V evaluates and reports the performance of the proposed methods in terms of accuracy and execution time. Section VI summarizes and concludes our work with the future direction.

II. RELATED WORK

There are many studies for overcoming catastrophic forgetting [5]–[17] in continual learning. Those can largely be classified into three categories. The first category of the works keeps and uses the previous data to mitigate catastrophic forgetting [8]–[10]. Robins [8] proposed a pseudo-rehearsal method, which uses random samples of the previous tasks when training for a new task. Rebuffi *et al.* [9] used a representative image, which is the mean feature vector of the past image data, when training for a new class. Lopez-Pa *et al.* [10] suggested the Gradient Episodic Memory (GEM), which modifies the gradient of the current task based on episodic memory storing the gradient of the previous tasks.

The second category of the works optimizes the model without storing the previous data [11]–[15]. Wu *et al.* [11] use Generative Adversarial Networks (GANs) to create the images containing the previous classes, and the created images are used when training for new tasks. Li *et al.* [13] proposed a method called the Learning Without Forgetting (LWF). The LWF trains a convolution layer commonly used for all the tasks and adds a fully connected layer separately for each new task. The LWF defines a new loss function limiting changes in the output of the fully connected layer for each previous task. Farajtabar *et al.* [15] presented an Orthogonal Gradient Descent (OGD) that projects the gradient of a new task in the orthogonal direction of the previous task's gradient. Its projected gradient is used for the model to get a low error for both new and previous tasks.

The last category of the works adds a regularization term to the loss function in order to discourage changes to weights that are important for the previous tasks [16], [17]. Elastic Weight Consolidation (EWC) [16] adjusts learning rates for certain weights to minimize changes in parameters important to the previous tasks. The EWC uses the Fisher information matrix to approximate the parameter importance and uses it as a guideline for the weight update. Zenke *et al.* [17] introduced Synaptic Intelligence (SI). The SI is similar to the EWC in that it finds the important parameters for the previous task and minimizes their change. But, it is different from EWC in that it considers the entire gradients of the previous tasks to find parameters rather than using the Fisher information matrix. The EWC and SI show outstanding performance in mitigating catastrophic forgetting. These two maintain a 90% or more in accuracy for the previous domains after training ten domains [45] and are considered as the state-of-the-arts.

There are studies for the AI accelerator to speed up the training. The Pezy SC2 [29] is a 2,048-core chip where each core is able to run eight threads. It performs linear algebra in parallel for the AI training. The Tokyo University designed the PFN-MN-3 [30] chip with a 32 GiB memory. The chip is composed of four dies, each of which has 2048 processing elements and 512 matrix arithmetic units for training. There are prior works on the FPGA-based AI accelerators. Cornami, an AI startup, has been developing an FPGA-based AI chip [32] for automobile and electronics markets. Their AI hardware can process 105,000 images per second with FP16 precision. The Flex Logix InferX X1 FPGA accelerator [33] targets both signal processing and machine learning. Its 64 processors are closely coupled with reconfigurable SRAM, which uses a proprietary interconnection for data movement of the model weights. There are also FPGA-based accelerator studies for continual learning. Piyasena et al [34], [35] proposed an accelerator for the streaming linear discriminant analysis (SLDA). They designed the SLDA accelerator with CNN implemented on Xilinx DPU, which is a programmable engine dedicated to convolutional neural networks. The accelerator is capable of object classification for life-long learning.

Table 1 summarizes the comparison between our approach and prior works. Joint training requires data of all previous tasks to optimize the model weights for all tasks. On the other hand, our approach does not require old tasks. Our proposed method is orthogonal to state-of-the-arts (EWC and SI) in methods finding important weights for the previous task. The EWC computes the Fisher information matrix at the end of

TABLE 1. Comparision of our proposed method with prior works in continual learning.

	Gradient Descent	*Joint training*	*EWC [16]*	*SI [17]*	*Proposed method (SB+AH)*	*Proposed method with EWC or SI*
Accuracy for new tasks	Good	Good	Good	Good	Good	Good
Accuracy for previous tasks	Bad	Best	Good	Good	Good	**Better**
Training speed	Normal	Slowest	Slow	Slower	Fast	**Fast**
Require data of past tasks?	No	Yes	No	No	No	No

each task, and the SI computes the parameter regularization term at the training step of each task. Thus, the EWC and SI take more time than normal training because of additional calculations. In contrast, our approach just uses the activation information to find the important weight. When combined with the state-of-the-arts, it shows superior performance in terms of accuracy and training speed.

III. THE NEURAL NETWORK TRAINING

The neural network training requires three sequential operations: forward propagation, backpropagation, and weight update. In the forward propagation, data is propagated from the input layer to the output layer; Each neuron computes a weighted sum of the inputs from the connected neurons in its prior layer, and then adds it with a bias, as shown in Eq. (1). Its output goes through an activation function that determines data to pass to the next layer. The widely used activation functions are *Rectified Linear Unit* (*ReLU*), *Tanh*, and *Sigmoid*. The ReLU in Eq. (2) is especially used in many DNNs. It propagates zero to the next layer when the input is negative, and otherwise bypasses the input value to the next layer. In the output layer, the *Softmax* function in Eq. (3) is widely used in deep learning. It computes the probability distribution of outcomes (y_z^o).

$$u_j^l = \sum_{i=1}^{N} w_{ij}^l x_i^l + Bias^l \tag{1}$$

$$y_j^l = max(0, u_j^l) \tag{2}$$

$$y_z^o = \frac{e^{u_z^o}}{\sum_{i=1}^{n} e^{u_i^o}} \tag{3}$$

where $1 \leq i \leq N$, $1 \leq l \leq O$(i, j, z, k = neuron index, l = layer index)

The backpropagation is used to adjust the weights (w_{ij}^l) by calculating derivatives. This phase begins from the output layer, which is based on Softmax in our work. The derivative in the output layer is expressed in Eq. (4). It calculates the difference between the forward propagation outcome (y_z^o) and target output (t_z). The derivative of the error with respect to the weight is calculated with Eq. (5). The derivative of the ReLU activation function is calculated with Eq. (6), which is a 0 when the outcome of Eq. (2) is zero, and a 1 otherwise.

$$\frac{dE}{dy_z^o} = y_z^o - t_z \tag{4}$$

$$\frac{dE}{dy_k^l} = \sum_z w_{kz}^{l+1} \frac{dE}{dy_z^{l+1}} \tag{5}$$

$$\delta_k^l = \frac{dE}{du_k^l} = \frac{dE}{dy_k^l}\frac{dy_k^l}{du_k^l} \tag{6}$$

In the DNN training, the weights are adjusted based on the errors computed in the backpropagation. First, Δw_{ij}^l is calculated by multiplying the result of the backpropagation δ_i^l with the result of the forward propagation y_j^{l-1}, as shown in Eq. (7). Weights (w_{ij}^l) are then updated according to Eq. (8) where the learning rate η determines the degree of learning. This process is repeated for all the weights.

$$\Delta w_{ij}^l = \delta_i^l y_j^{l-1} \tag{7}$$

$$w_{ij}^l = w_{ij}^l - \Delta w_{ij}^l \eta \tag{8}$$

IV. MITIGATING CATASTROPHIC FORGETTING

This section details our proposed methods for mitigating catastrophic forgetting: Speculative Backpropagation and Activation History. Each method is implemented in a multi-layer perceptron with two hidden layers (400 neurons in each layer) [14]–[17], [45], and its performance is demonstrated with Permuted and Split MNIST benchmarks detailed in Section V.

A. SOFTMAX HISTORY AND BIASED LERU WITH SPECULATIVE BACKPROPAGATION (SB)

In the ANN training, the backpropagation is performed based on the forward propagation outcomes. It means that the backpropagation can be carried out only after the forward propagation is finished. Speculative Backpropagation (SB), our prior work [36], enables the simultaneous operation of the forward and the backward propagations. The SB is based on the observation that the Softmax and ReLU outcomes for the same labels are similar in the temporally near-forward propagations. Figure 1 shows an example of how the SB is carried out. At the current time t(i), the backpropagation is performed based on the Softmax history and previous ReLU outcomes depicted as red and blue neurons, respectively. It breaks the inherently sequential nature of SGD in the backward computation, enabling the simultaneous operation

TABLE 2. Task 1's accuracies with SB for different pairs of α and β After training tasks 1 ~ 4 sequentially. (Task 1's accuracies with normal training are 82% and 41.1% for handwritten and fashion, respectively).

Hyperparameters value	$\alpha = 0.1$ $\beta = 0.9$	$\alpha = 0.2$ $\beta = 0.8$	$\alpha = 0.3$ $\beta = 0.7$	$\alpha = 0.4$ $\beta = 0.6$	$\alpha = 0.5$ $\beta = 0.5$	$\alpha = 0.6$ $\beta = 0.4$	$\alpha = 0.7$ $\beta = 0.3$	$\alpha = 0.8$ $\beta = 0.2$	$\alpha = 0.9$ $\beta = 0.1$
MNIST handwritten SB accuracies	90.78%	91.15%	91.14%	88.86%	88.19%	86.92%	83.57%	82.3%	81.99%
MNIST Fashion SB accuracies	52.02%	52.22%	52.2%	51.71%	50.1%	48.21%	43.77%	41.52%	40.08%

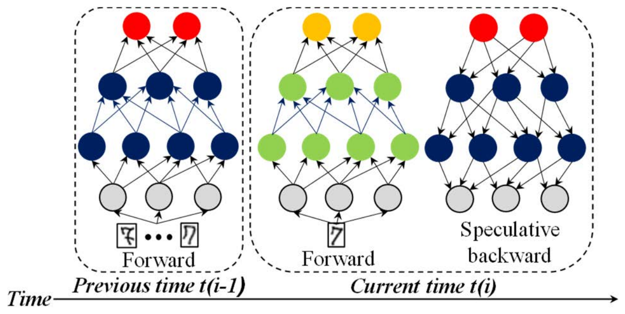

FIGURE 1. Simultaneous execution of forward and backward computations with speculative backpropagation, which uses the accumulated previous forward outcomes. The red neurons have Softmax history accumulated until time t(i-1), and the *blue* neurons have previous ReLU outcomes. These are used for speculative backpropagation at time *t*(*i*), instead of using current forward outcomes (in *yellow* and *green* neurons). Note that the weights in time *t*(*i*) are used to perform the speculative backpropagation.

of forward and backward propagations. The SB checks the speculation correctness based on the threshold. If the difference between the current Softmax output and the speculated one is larger than the threshold, the speculated execution is nullified, and the backpropagation is performed again with the current Softmax output. According to our experiments, the reciprocal of the number of classes in the dataset can be a good candidate for the threshold. In this paper, the threshold is set to 0.1 for experiments with Permuted and Split MNISTs.

In addition to the simultaneous operation, we also found that SB also helps preserve the previously learned information for continual learning. In our prior work [36], the Softmax history is updated with Eq. (9), where α and β are weights for the current and the accumulated Softmax outcomes, respectively. Setting α and β to 0.5 empirically proved to be working well in our prior work for shortening the training time, while providing comparable or even better accuracy. For continual learning, we have experimented with different pairs of α and β. Table 2 shows the experiment result reporting task 1's accuracies after sequentially training Permuted MNISTs (tasks 1 $\sim$ 4). Giving more weight to β tends to better mitigate catastrophic forgetting. It is because the past knowledge is likely to be preserved with more weight on history. Task 1's accuracy is the highest when $\alpha = 0.2$ and $\beta = 0.8$ for MNIST Handwritten and Fashion. However, the

Algorithm 1 Biased ReLU According to Activation History

```
1: $f(u_j^l)$: current ReLU output, $f(u_j^l)'$: biased ReLU
2: if $f(u_j^l) == 0$ do (when ReLU output is zero)
3:     $f(u_j^l)' \leftarrow f(u_j^l)'/2$; (ReLU is biased towards 0)
4: Else
5:     $f(u_j^l)' \leftarrow (f(u_j^l)' + 1)/2$; (ReLU is biased towards 1)
6: Where biased ReLU $f(u_j^l)'$ : $(0 <= f(u_j^l)' <= 1)$
```

performance is degraded with the extremely biased case ($\alpha = 0.1$ and $\beta = 0.9$) where the number of miss speculations is increased. The number of miss speculations is closely related to the continual learning performance. It is because, when the speculation is wrong, the backpropagation is performed with the current Softmax outcome, not the one with the accumulated history. When $\alpha = 0.1$ and $\beta = 0.9$ for MNIST Handwritten and Fashion, the miss speculation rates are 11% and 17%, which are 5% and 3% higher than the ones with $\alpha = 0.2$ and $\beta = 0.8$, respectively.

$$
\begin{aligned}
y_{t(i)}^{o'} &= (y_{t(i)}^{o}\alpha) + (y_{t(i-1)}^{o'}\beta)\ where\ (\alpha + \beta = 1)\\
y_{t(i)}^{o} &= \text{current Softmax outcome,}\\
y_{t(i-1)}^{o'} &= \text{accumulated Softmax outcome until time t(i-1)}
\end{aligned}
\tag{9}
$$

The SB performs backpropagation with the most recent ReLU outcomes. For continual learning, considering both the most recent and the history of the ReLU outcomes helps preserve the knowledge. Algorithm 1 shows a method of reflecting the history of activation outputs, where it defines $f(u_j^l)'$, referred to as the *biased ReLU*. The biased ReLU is adjusted by the current activation outcome whenever the forward propagation is finished. When the neurons are deactivated, the biased ReLU becomes closer to 0. When the neurons are activated, it gets closer to 1. Algorithm 2 shows a method of speculating the neuron's activation based on the biased ReLU. When the biased ReLU is smaller than 0.5, it predicts the neuron will be deactivated. Otherwise, it speculates the neuron will be activated. When SB is performed based on Algorithm 1 and 2, accuracies of the previous task are higher by 2.3% and 1.9% on average for Permuted Handwritten and Fashion, respectively.

Algorithm 2 ReLU Outcome Speculation

```
1: if $f(u_j^l)' < 0.5$ do (when biased ReLU is smaller than 0.5)
2:     $\delta_k^l \leftarrow 0$; (speculate as the deactivation of the neuron)
3: else
4:     $\delta_k^l \leftarrow 1$; (speculate as the activation of the neuron)
```

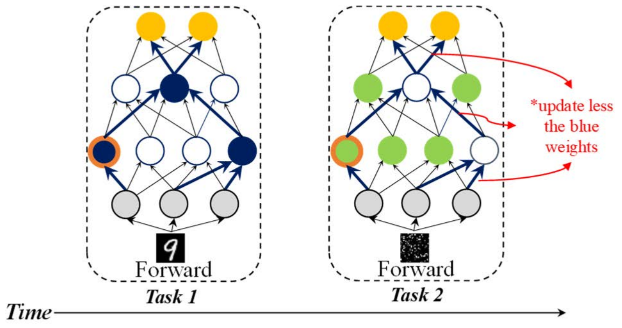

FIGURE 2. The blue and green ones are activated neurons when performing forward propagation with task 1 and task 2, respectively. The blue arrows are the weights of the blue neurons, and they play an important role in task 1. When training task 2, it is possible to preserve the knowledge of task 1 with a biased weight update of the blue neurons. The one with the orange circle is an activated neuron in common for both task 1 and task 2.

B. BIASED WEIGHT UPDATE WITH ACTIVATION HISTORY (AH)

For continual learning, it would help preserve the retained knowledge if important weights for the previous task can be isolated and affected little when training for other tasks. We have found two patterns of activated neurons when training a model for different tasks: (a) While training one specific task, the neurons activated in the past are more likely to be activated in the future, and the deactivated neurons are more likely to be deactivated. (b) The activated neurons differ from task to task. Only activated neurons affect the inference result and play an important role in that task due to the ReLU. The ReLU passes the input value to the next layer upon activated, or propagates zero to the next layer otherwise. When experimented with a multi-layer perceptron for the Permuted MNIST Handwritten and Fashion, activation probabilities of the same neurons were 87.8% and 84.2% on average, respectively. Figure 2 shows an example of the activated neurons for task 1 and task 2. The blue and green ones are activated neurons for task 1 and task 2, respectively. The neuron with the orange circle is activated both for tasks 1 and 2. The activation probabilities of blue neurons for task 1 are 81% and 77.2% on average for Handwritten and Fashion, respectively. The activation probabilities of green neurons for task 2 are 88.6% and 88.1% on average for Handwritten and Fashion, respectively. Thus, it is reasonable to say that the weights connected to the blue and green neurons are important for tasks 1 and 2, respectively.

Algorithm 3 shows a method updating the weights according to the activation history, and Table 3 describes the notation of Algorithm 3. A variable called activation history (ah_j^l) stores the tendency of neuron's activation while training for the previous tasks. After training for each task, ah_j^l is adjusted with the biased ReLU computed in Algorithm 1. r specifies the degree of weight update and directly affects the knowledge preservation performance. In proportion to r, the weights connected to activated neurons in the past are updated less (line# 3-4), when training for a new task. Table 4 shows the experiment result reporting task 1's accuracies according to r, after sequentially training Permuted MNISTs (tasks 1 $\sim$ 4). For each distinct selection of r, the accuracy is higher compared to the normal training (baseline), meaning that the retained knowledge is better preserved. In general, it tends to be better as r gets smaller. The accuracies reach the highest with $r = 0.3$ for MNIST Handwritten and with $r = 0.4$ for Fashion. However, we have found that the performance for the new task gets degraded when r is too low. For example, when r is set to 0.1, the task 4's accuracy is 98.3% on average, which is 0.7% lower than the baseline (99%). This is because some neurons (such as the one in the orange circle in Figure 2) are activated for both previous and current tasks. In such a case, a tiny r indicates that the weights connected to the activated neurons in the past are minimally trained for a new task. According to our experiment, 47.2% of neurons were commonly activated both for task1 and task2, and 49.8% of neurons were activated in common both for task2 and task3. The weights connected to these neurons affect the inference for both tasks. Therefore, in order to

Algorithm 3 Biased Weight Update According to Activation History

```
1: while weight not converged do
2:     $g \leftarrow \nabla_w f(w)$ (get gradients with objective function)
3:     if $ah_j^l > 0.5$ do (if the neuron is activated with high probability for the previous task)
4:         $g \leftarrow g \cdot r$ (reduce the gradients of the weights)
5:     $w \leftarrow w - \eta \cdot g$ (update the weight)
6: end while
7: $ah_j^l \leftarrow (ah_j^l + f(u_j^l)')/2$; (accumulate biased ReLU to activation history after training each task)
8: return $w$
```

TABLE 3. Notations used in algorithm 3.

Notation	Description
w	weight of neural network
$f(w)$	gradient descent objective function with weight
g	gradient
ah_j^l	ReLU activation history of the previous tasks
r	rate of biased weight update ($0 < r < 1$)
η	learning rate
$f(u_j^l)'$	biased ReLU

TABLE 4. Task 1's accuracies for AH with different ratios of weight update After training tasks 1 ∼ 4 sequentially. (Task 1's accuracies with normal training are 82% and 41.1% for handwritten and fashion, respectively).

weight update degree (r)	$r = 0.1$	$r = 0.2$	$r = 0.3$	$r = 0.4$	$r = 0.5$	$r = 0.6$	$r = 0.7$	$r = 0.8$	$r = 0.9$
MNIST Handwritten SB accuracies	91.10%	91.11%	92%	91.8%	91.83%	89.97%	85.12%	83.1%	82.69%
MNIST Fashion SB accuracies	60.70%	60.72%	61.35%	61.61%	61.42%	57.01%	52.83%	48.46%	45.19%

balance knowledge retention and new learning, it is crucial to find the appropriate r. We empirically found that $r = 0.3$ is the right selection for preserving the knowledge and for training for a new task. With $r = 0.3$, the accuracies of the previous tasks are roughly 15.1% higher on average than the baseline without the accuracy loss for a new task.

C. MITIGATING CATASTROPHIC FORGETTING USING BOTH SB AND AH

SB and AH can be applied together for continual learning because the two methods are orthogonal. The SB is applied to Eq. (1-3) for the forward propagation and Eq. (4-6) for the backward propagation. The AH is applied to Eq. (8) for the weight update with Algorithm 3. We have found that using both SB and AH shows better performance for continual learning. Its evaluation is detailed in Section V.

V. EVALUATION

This section reports the performance of the proposed methods in terms of accuracy and execution time.

A. EXPERIMENTAL ENVIRONMENT

We implemented a multi-layer perceptron with the SB and AH using C language and evaluated its accuracy on an AMD 3970X 32-core machine with 64GiB main memory and GeForce RTX 3090 GPU. Permuted and Split MNIST benchmarks are used to evaluate the continuous learning capability; The MNIST [38] is a benchmark for classifying 10 different digits. Each task in the Permuted MNIST is created by the random permutation of the pixels in the images. Permuted MNIST is widely adopted [14]–[17], [39], [40], [41], [45] for the evaluation of continual learning methods. In this paper, we have created 10 different tasks by permutation. Split MNIST [14], [15], [17], [41], [45] is also typically used in the evaluation. It is constructed by separating 10-digit classifications into 5 different binary classification tasks. The proposed SB and AH were experimented in two continual learning scenarios [45]: Incremental domain learning and Incremental task learning; The former [14]–[16] means that the model with a single-headed output layer is trained for multi-domain problems. The latter [17], [45] means the model with a multi-headed output layer is trained for multiple tasks (one head for each task). Permuted and Split MNISTs are used in the incremental domain and task learnings, respectively.

We also implemented hardware accelerators for the evaluation of training time on an off-the-shelf machine. The ZCU102 FPGA board [46] is used as an equipment. It has Zynq UltraScale+ MPSoC [47] and a 4GB DDR4. The Zynq UltraScale+ is composed of the processing system (PS) and programmable logic (PL) sections: The PS has a quad-core Cortex-A53 processor operating at 1.5 GHz. The PL is configured with the hardware accelerator in our work. SDSoC 2019.1v [44], a CAD tool from Xilinx, is used for hardware and software implementation. It provides the capability of automating the system-level integration for C/C++/OpenCL code, targeting the Zynq programmable SoCs. The system-level integration includes the software-to-hardware translation, its device driver generation, and kernel creation; Users can specify software functions to be translated to hardware in SDSoC. We designed two C functions performing the forward propagation and the speculative backpropagation. Then, the C functions were translated to hardware by the SDSoC. Two directives are used to generate the target hardware: *#pragma async* to generate two different hardware operating in parallel and *#pragma pipeline* to create the pipelining for maximal throughput. The synthesized accelerator operates at 100MHz, and the S/W portion in the code is processed on the Cortex-A53 in the PS.

B. CATASTROPHIC FORGETTING

The proposed methods were implemented on a multi-layer perceptron with two hidden layers, and each layer has 400 neurons. As mentioned in Section IV, α, β, and r affect the accuracy of the previous task. The experiments were performed with $\alpha = 0.2$ and $\beta = 0.8$ for SB and with $r = 0.3$ for AH. Figure 3 shows the accuracies for Permuted MNISTs. The accuracies of all previous tasks with SB are higher than SGD. In case of the Handwritten, the task1's accuracy with SB is 91.1% as shown in Figure 3(a), after sequentially training from task 1 to task 4. It is 9.1% higher than the baseline, which is the SGD's accuracy (82%). In case of the Fashion in Figure 3(b), the task 1's accuracy with SB is 52.2%, which is 11.1% higher than the baseline (41.1% with SGD).

AH outperforms SB in the accuracies of the previous tasks. When it comes to the Handwritten, the task 1's accuracy with AH is 92% after sequentially training from task 1 to task 4. It is 0.9% higher than the SB. In case of the Fashion, the task 1's accuracy with AH is 9.2% higher compared to the

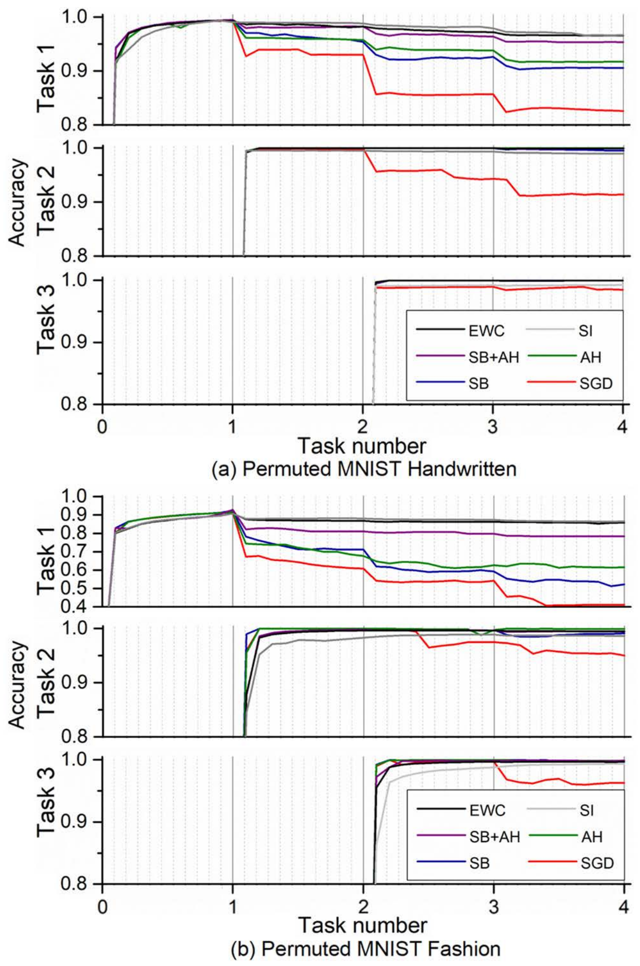

FIGURE 3. Accuracies of previous tasks for Permuted MNISTs with SB, AH, SB+AH, EWC and SI. Baseline is normal training with SGD in red.

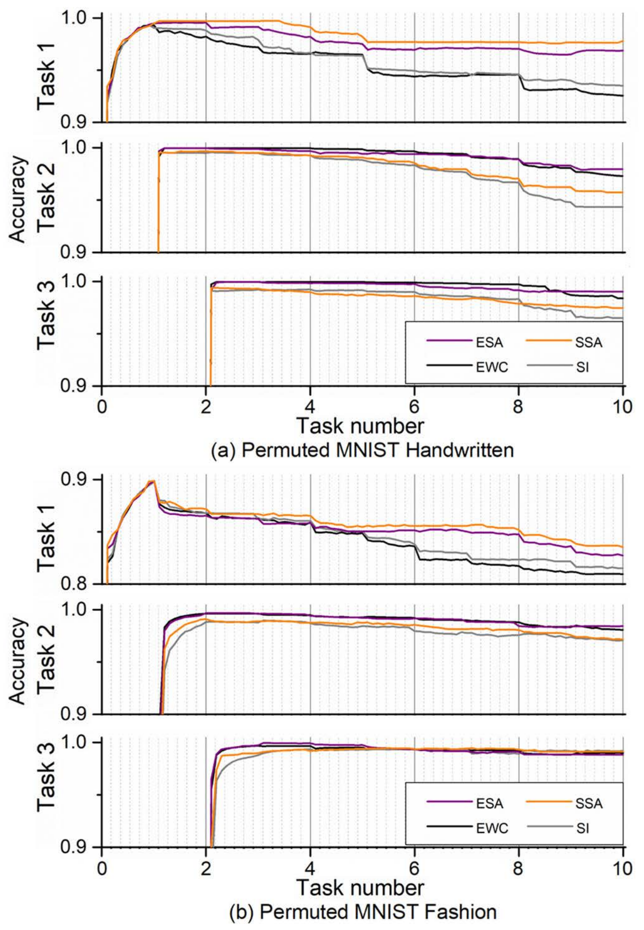

FIGURE 4. Accuracies of previous tasks for Permuted MNISTs with EWC, SI, ESA, and SSA (ESA = EWC+SB+AH, SSA = SI+SB+AH).

SB. The SB's performance on knowledge retention is not as powerful as AH. However, it enables the simultaneous operation of forward and backward propagations, shortening the training time. The training time is reported in Section V.C. with an off-the-shelf machine.

Combining SB and AH provides superior performance to each method alone. For the Handwritten, the task 1's accuracy with SB+AH is 95.1%, which is 4% and 3.1% better than SB and AH, respectively. For the Fashion, the task 1's accuracy with SB+AH is 78.7%, which is 26.5% and 17.3% higher than SB and AH, respectively. When comparing with EWC and SI, the training with SB+AH results in a slightly lower performance. On average, the accuracy of previous tasks for Handwritten is 0.9% and 1% lower than EWC and SI, respectively. For the Fashion, the accuracy is 2.3% and 2% lower than the one with EWC and SI, respectively.

We have applied and implemented SB and AH on top of EWC and SI, respectively. Figure 4 shows the accuracies of previous tasks with the EWC, SI, EWC+SB+AH (ESA), and SI+SB+AH (SSA) for the Permuted MNISTs. On average, the accuracies with ESA and SSA are 3.9% and 2.8% higher than the ones with EWC and SI, respectively. In case of the Handwritten in Figure 4(a), the task 1's accuracy with ESA is 96.9%, which is 4.4% better than the one with EWC. The accuracy with SSA is 97.8%, which is 4.3% higher than the one with SI. In case of the Fashion in Figure 4(b), the task 1's accuracy with ESA is 82.8%, which is 1.9% better than the one with EWC. The accuracy with SSA is 83.7%, which is 2.2% higher than the one with SI.

Figure 5 shows the accuracies with the EWC, SI, ESA, and SSA for the Split MNISTs. On average, the accuracies of the previous tasks with ESA and SSA are 1.4% and 1.7% better than EWC and SI, respectively. In case of the Handwritten in Figure 5(a), the accuracies of tasks 1, 2, and 3 with ESA are 99.3%, 99.5%, and 99.7%, which are 1.9%, 0.9%, and 1.1% higher than the ones with EWC, respectively. The accuracies with SSA are 99.8%, 99.4%, and 99.7%, which are 1.7%, 1.9%, and 0.9% superior to the ones with SI, respectively. In case of the Fashion in Figure 5(b), the accuracies of tasks 1, 2, and 3 with ESA are 99.5%, 98.4%, and 99.6%, which are 1.2%, 0.9%, and 1.2% higher than the ones with EWC. The accuracies with SSA are 99.7%, 98.4%, and 99.6%, which are 1%, 1.4%, and 0.7% superior to the ones with SI.

Our experiments reveal that continual learning ability is improved when applying SB and AH on top of EWC or SI.

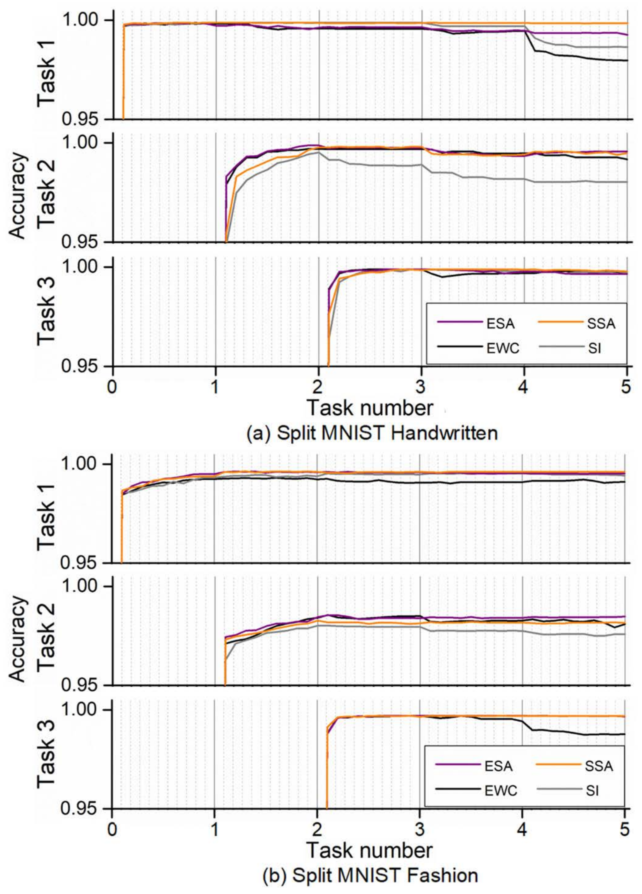

FIGURE 5. Accuracies of previous tasks for Split MNISTs with EWC, SI, ESA, and SSA. (ESA = EWC+SB+AH, SSA = SI+SB+AH).

EWC and SI focus on important weights for the previous task based on the past gradients. The SB and AH focus on the activation history and the Softmax outcomes of the past task. Considering both the previous task's gradients and activation history tends to help find important weights better for continual learning. The implemented scheme minimizes changes to important weights for previous tasks and trains the remaining weights for a new task according to the history of previous tasks.

C. H/W ACCELERATOR AND PERFORMANCE

We implemented four different hardware accelerators using SDSoC, to compare the training times on an off-the-shelf device called Zynq Ultrascale+: The first two are the baseline accelerators implementing EWC and SI, respectively. The other two are the parallel accelerators implementing ESA and SSA, respectively. The hardware designed in PL performs two operations; (a) Forward and backward propagation (b) weight selections based on EWC and SI. The other works are carried out in software, such as weight update and activation history accumulation. Table 5 shows the resource utilization and power consumption of the accelerators targeting the Zynq Ultrascale+. In terms of resource utilization, there are almost no differences between the baselines and the

TABLE 5. Hardware resource utilization and power consumption of training accelerators on zynq ultrascale+ [47].

Hardware Accelerators	*#BRAMs (18Kbit)*	*#DSPs (48E)*	*#FFs*	*#LUTs*	*Power Consumption*
Baseline accelerator for EWC	168.5 (18.48%)	41 (1.63%)	199,441 (36.38%)	144,355 (52.67%)	4.888W
Baseline accelerator for SI	200.5 (21.98%)	44 (1.75%)	217,084 (39.6%)	158,888 (57.97%)	4.995W
Accelerator for ESA	168.5 (18.48%)	41 (1.63%)	199,444 (36.38%)	144,362 (52.67%)	4.889W
Accelerator for SSA	200.5 (21.98%)	44 (1.75%)	217,087 (39.6%)	158,896 (57.97%)	4.998W

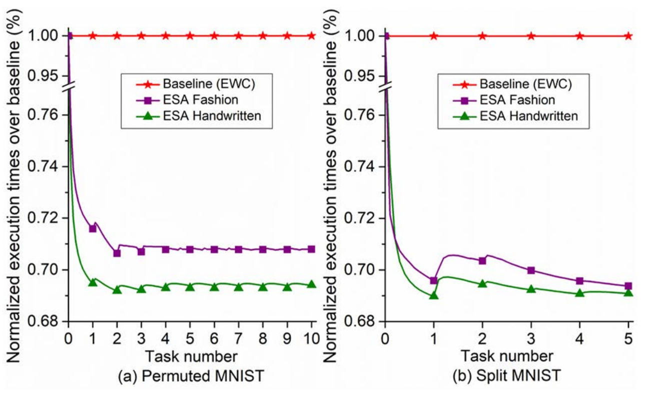

FIGURE 6. Normalized execution time of the parallel accelerator for ESA over the baseline. Baseline is an accelerator for EWC.

parallel accelerators. This is because the parallel accelerator is composed of roughly the same components as the baseline; Both accelerators should perform the two operations (forward and backward propagations) anyway. In tandem with the similar resource utilizations, the power consumption is also similar in both accelerators.

Figure 6 shows normalized execution times of the ESA accelerator over the baseline. With the Permuted MNIST Handwritten and Fashion in Figure 6(a), the training times were reduced by 1.30x and 1.29x, respectively. With the Split MNIST Handwritten and Fashion in Figure 6(b), the execution times were shortened by 1.31x and 1.308x over the baseline, respectively. Figure 7 shows normalized execution times of the SSA accelerator over the baseline. With the Permuted MNISTs in Figure 7(a), the training times were reduced by 1.228x and 1.208x. With the Split MNISTs in Figure 7(b), the execution times were shortened by 1.227x and 1.218x over the baseline.

The ESA accelerator provides a relatively higher performance improvement than SSA in training speed, as shown in Figures 6 and 7. In the parallel accelerators, the forward and

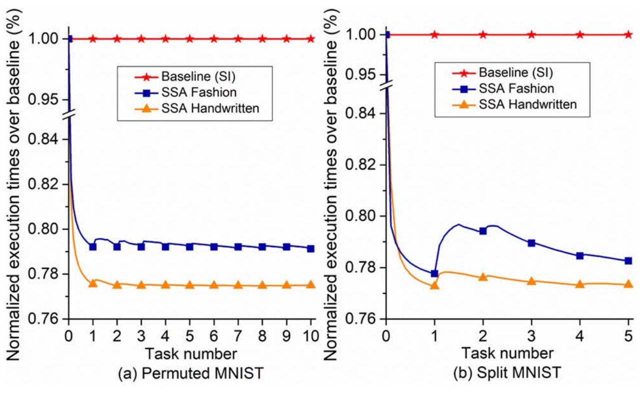

FIGURE 7. Normalized execution time of the parallel accelerator for SSA over the baseline. Baseline is an accelerator for SI.

backward computations are performed in parallel, and additional operations for EWC and SI are carried out sequentially afterward. The amount of the additional computations in ESA is much less than the one in SSA. To find important weights for the previous task, EWC computes the Fisher information metric at the end of training for a task, whereas SI computes the parameter regularization term at every training step of a task. The speedup over the baseline is decreased as the sequential part of the operations increases.

When training for task 2 at the first epoch, ESA and SSA training times are increased by about 1% and 2%, respectively, as shown in Figures 6(b) and 7(b). This is because the miss speculation rate in SB is increased. As mentioned in Section IV.A, the SB performs backpropagation again upon miss speculation, and it negatively affects the execution time. In the first epoch of training for task 2, the miss speculation rates of ESA and SSA are 18.8% and 36.6%, respectively. This is 14.2% and 29.4% higher than the miss rate in the last epoch of task 1 with ESA (4.6%) and SSA (7.2%). It is because the speculation in the first epoch is based only on the knowledge from the previous task. Thus, the speculation in the first epoch is not likely to be accurate for the current task. As the training epoch progresses, the speculation is more likely to be influenced by the current task based on Eq. (9) and is getting more accurate. It is reflected in the execution times at the last epochs shown in Figures 6 and 7.

VI. CONCLUSION

In this paper, we proposed two novel ideas of mitigating catastrophic forgetting for continual learning: SB and AH. The SB enables performing backpropagation based on past knowledge. The AH enables isolating important weights for the previous task based on the activation history. Our evaluation reveals the performance advantage of our scheme in terms of accuracy and training time. The SB+AH improves the accuracy of the previous tasks by 30.2%, compared to the normal training, for Permuted MNIST Fashion. When combined together with the state-of-the-arts in continual learning, our approach offers the improved accuracy. Compared with EWC and SI, the accuracies of the previous task with ESA and SSA were improved by up to 4.4% and 4.3%, respectively, for the Permuted MNIST Handwritten. The experiments with hardware accelerators report shorter execution time in training on an off-the-shelf device. The training times with ESA and SSA were reduced by 30% and 22% compared to EWC and SI, respectively, for the Split MNIST Fashion. In the future, we plan to extend our work to other deep learning models such as CNN and RNN.

REFERENCES

[1] C. S. Lee and A. Y. Lee, "Clinical applications of continual learning machine learning," *Lancet Digit. Health*, vol. 2, no. 6, pp. e279–e281, Jun. 2020.

[2] K. N. Vokinger, S. Feuerriegel, and A. S. Kesselheim, "Continual learning in medical devices: FDA's action plan and beyond," *Lancet Digit. Health*, vol. 3, no. 6, pp. e337–e338, Jun. 2021.

[3] M. Chatterjee. (2019). *Top 20 Applications of Deep Learning in 2021 Across Industries*. [Online]. Available: https://www.mygreatlearning.com/blog/deep-learning-applications/#healthcare

[4] Towards Data Science. (May 7, 2019). *Tesla's Deep Learning at Scale: Using Billions of Miles to Train Neural Networks*. Accessed: Apr. 16, 2021. [Online]. Available: https://towardsdatascience.com/teslas-deep-learning-at-scale-7eed85b235d3

[5] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of Learning and Motivation*, vol. 24. New York, NY, USA: Academic, 1989, pp. 109–165.

[6] R. Ratcliff, "Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions," *Psychol. Rev.*, vol. 97, no. 2, p. 285, 1990.

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

[8] A. Robins, "Catastrophic forgetting, rehearsal and pseudorehearsal," *Connection Sci.*, vol. 7, no. 2, pp. 123–146, 1995.

[9] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "ICaRL: Incremental classifier and representation learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2001–2010.

[10] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," 2017, *arXiv:1706.08840*.

[11] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu, "Large scale incremental learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 374–382.

[12] J. Serra, "Overcoming catastrophic forgetting with hard attention to the task," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4548–4557.

[13] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 2935–2947, Dec. 2018.

[14] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner, "Variational continual learning," 2017, *arXiv:1710.10628*.

[15] M. Farajtabar, "Orthogonal gradient descent for continual learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 3762–3773.

[16] K. James, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 13, pp. 3521–3526, Mar. 2017.

[17] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3987–3995.

[18] C. Farabet, B. Martini, B. Corda, P. Akselrod, E. Culurciello, and Y. LeCun, "NeuFlow: A runtime reconfigurable dataflow processor for vision," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2011, pp. 109–116, doi: 10.1109/CVPRW.2011.5981829.

[19] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "EIE: Efficient inference engine on compressed deep neural network," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Architecture (ISCA)*, Jun. 2016, pp. 243–254. [Online]. Available: http://ieeexplore.ieee.org/document/7551397/

[20] Y.-H. Chen, J. Emer, and V. Sze, ''Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks,'' *IEEE Micro*, vol. 44, no. 3, pp. 367–379, Jun. 2016, doi: 10.1145/3007787.3001177.

[21] Y. Chen, T. Chen, Z. Xu, N. Sun, and O. Temam, ''DianNao family: Energy-efficient hardware accelerators for machine learning,'' *Commun. ACM*, vol. 59, no. 11, pp. 105–112, Oct. 2016. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3013530.2996864

[22] (2019). *EdgeTPU*. [Online]. Available: https://cloud.google.com/edge-tpu/

[23] (Oct. 2019). *Cornami Achieves Unprecedented Performance at Lowest Power Dissipation for Deep Neural Networks*. [Online]. Available: https://cornami.com/1416-2/

[24] P. Teich, ''Tearing apart Google's TPU 3.0 AI coprocessor,'' Next Platform, U.K., Tech. Rep., May 2018.

[25] E. Medina and E. Dagan, ''Habana labs purpose-built AI inference and training processor architectures: Scaling AI training systems using standard Ethernet with Gaudi processor,'' *IEEE Micro*, vol. 40, no. 2, pp. 17–24, Mar. 2020, doi: 10.1109/MM.2020.2975185.

[26] L. Gwennap, ''Habana offers Gaudi for AI training,'' Microprocessor Rep., Habana Labs, USA, Jun. 2019. [Online]. Available: https://habana.ai/wp-content/uploads/2019/06/ Habana-Offers-Gaudi-for-AI-Training.pdf

[27] R. Merritt. (Jul. 2018). *Baidu Accelerator Rises in AI*. [Online]. Available: https://www.eetimes.com/baidu-accelerator-rises-in-ai/

[28] C. Duckett. (Jul. 2018). *Baidu Creates Kunlun Silicon for AI*. [Online]. Available: https://www.zdnet.com/article/ baidu-creates-kunlun-silicon-for-ai/

[29] D. Schor. (Nov. 2017). *The 2,048-Core PEZY-SC2 Sets a Green500 Record—WikiChip Fuse*. [Online]. Available: https://fuse.wikichip.org/news/191/the-2048-core-pezy-sc2-sets-a-green500-record/

[30] L. Gwennap, ''Tenstorrent scales AI performance: Architecture leads in data-center power efficiency,'' Microprocessor Rep., Tenstorrent, USA, Apr. 2020. [Online]. Available: https://www.tenstorrent.com/wp-content/uploads/2020/04/Tenstorrent-Scales-AI-Performance.pdf

[31] A. Reuther, P. Michaleas, M. Jones, V. Gadepally, S. Samsi, and J. Kepner, ''Survey of machine learning accelerators,'' in *Proc. IEEE High Perform. Extreme Comput. Conf. (HPEC)*, Sep. 2020, pp. 1–12.

[32] (Oct. 2019). *Cornami Achieves Unprecedented Performance at Lowest Power Dissipation for Deep Neural Networks*. [Online]. Available: https://cornami.com/1416-2/

[33] V. Mehta, ''Performance estimation and benchmarks for real-world edge inference applications,'' in *Proc. Linley Spring Processor Conf.* Linley Group, 2020. [Online]. Available: https://www.youtube.com/watch?v=PS3BjfzhYGo

[34] D. Piyasena, S.-K. Lam, and M. Wu, ''Accelerating continual learning on edge FPGA,'' in *Proc. 31st Int. Conf. Field-Program. Log. Appl. (FPL)*, Aug. 2021, pp. 294–300.

[35] D. Piyasena, S.-K. Lam, and M. Wu, ''Edge accelerator for lifelong deep learning using streaming linear discriminant analysis,'' in *Proc. IEEE 29th Annu. Int. Symp. Field-Program. Custom Comput. Mach. (FCCM)*, May 2021, p. 259.

[36] S. Park and T. Suh, ''Speculative backpropagation for CNN parallel training,'' *IEEE Access*, vol. 8, pp. 215365–215374, 2020.

[37] Y. LeCun and C. Cortes. *MNIST Handwritten Digit Database*. Accessed: Oct. 17, 2020. [Online]. Available: http://yann.lecun.com/exdb/mnist/

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

[39] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang, ''Overcoming catastrophic forgetting by incremental moment matching,'' 2017, *arXiv:1703.08475*.

[40] H. Ritter, A. Botev, and D. Barber, ''Online structured Laplace approximations for overcoming catastrophic forgetting,'' 2018, *arXiv:1805.07810*.

[41] H. Shin, J. K. Lee, J. Kim, and J. Kim, ''Continual learning with deep generative replay,'' 2017, *arXiv:1705.08690*.

[42] S. Farquhar and Y. Gal, ''Towards robust evaluations of continual learning,'' 2018, *arXiv:1805.09733*.

[43] D. P. Kingma and J. Ba, ''Adam: A method for stochastic optimization,'' 2014, *arXiv:1412.6980*.

[44] V. Kathail, J. Hwang, W. Sun, Y. Chobe, T. Shui, and J. Carrillo, ''SDSoC: A higher-level programming environment for Zynq SoC and ultrascale+ MPSoC,'' in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, Feb. 2016, p. 4.

[45] Y.-C. Hsu, Y.-C. Liu, A. Ramasamy, and Z. Kira, ''Re-evaluating continual learning scenarios: A categorization and case for strong baselines,'' 2018, *arXiv:1810.12488*.

[46] *Zynq UltraScale+ MPSoC ZCU102 Evaluation Kit*. Accessed: Nov. 21, 2021. [Online]. Available: https://www.xilinx.com/products/boards-and-kits/ek-u1-zcu102-g.html

[47] *Zynq UltraScale+ MPSoC*. Accessed: Dec. 13, 2021. [Online]. Available: https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html

[48] *Training Deep Net on 14 Million Images by Using a Single Machine*. Accessed: Jan. 9, 2022. [Online]. Available: https://mxnet-tqchen.readthedocs.io/en/latest/tutorials/imagenet_full.html