applied sciences

MDPI

Article

AdaCB: An Adaptive Gradient Method with Convergence Range Bound of Learning Rate

Xuanzhi Liao *, Shahnorbanun Sahran, Azizi Abdullah and Syaimak Abdul Shukor

Center for Artificial Intelligence Technology, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia (UKM), Bangi 43600, Selangor, Malaysia
* Correspondence: p98708@siswa.ukm.edu.my

Abstract: Adaptive gradient descent methods such as Adam, RMSprop, and AdaGrad achieve great success in training deep learning models. These methods adaptively change the learning rates, resulting in a faster convergence speed. Recent studies have shown their problems include extreme learning rates, non-convergence issues, as well as poor generalization. Some enhanced variants have been proposed, such as AMSGrad, and AdaBound. However, the performances of these alternatives are controversial and some drawbacks still occur. In this work, we proposed an optimizer called AdaCB, which limits the learning rates of Adam in a convergence range bound. The bound range is determined by the LR test, and then two bound functions are designed to constrain Adam, and two bound functions tend to a constant value. To evaluate our method, we carry out experiments on the image classification task, three models including Smallnet, Network IN Network, and Resnet are trained on CIFAR10 and CIFAR100 datasets. Experimental results show that our method outperforms other optimizers on CIFAR10 and CIFAR100 datasets with accuracies of (82.76%, 53.29%), (86.24%, 60.19%), and (83.24%, 55.04%) on Smallnet, Network IN Network and Resnet, respectively. The results also indicate that our method maintains a faster learning speed, like adaptive gradient methods, in the early stage and achieves considerable accuracy, like SGD (M), at the end.

Keywords: deep learning; adaptive gradient descent methods; stochastic gradient descent; convergence-range bound; image classification

Citation: Liao, X.; Sahran, S.; Abdullah, A.; Shukor, S.A. AdaCB: An Adaptive Gradient Method with Convergence Range Bound of Learning Rate. *Appl. Sci.* **2022**, *12*, 9389. https://doi.org/10.3390/app12189389

Academic Editor: Christos Bouras

Received: 20 July 2022
Accepted: 16 September 2022
Published: 19 September 2022

1. Introduction

Deep learning methods have yielded significant success in several complicated applications, such as image classification [1–3] and object detection [4–7]. It is essential to train a deep convolutional neural network with an excellent optimizer to obtain considerable accuracy and convergence speed. Stochastic gradient descent (SGD) [8] is one of the primary and dominant optimizers that performs well across many applications due to its simplicity [9,10]. However, SGD scales the gradients uniformly in all directions with a constant learning rate. This strategy leads to a slow training process and sensitivity to tuning learning rates. Recent works proposed various adaptive gradient methods to achieve faster convergence by computing individual learning rates for different gradients to address this problem. Examples of these adaptive methods include AdaGrad [11], RMSProp [12], and Adam [13]. These methods use the exponential moving average (EMA) [14] of the past square gradients to adjust the learning rates. Particularly, Adam is one of the famous adaptive gradient methods, and it has become the alternative optimizer after SGD(M) across many deep learning frameworks because of its rapid training speed [15,16]. Despite their popularity, there is still a performance gap between the adaptive gradient methods with well-tuned SGD (M) [17].

However, recent theories have shown that Adam suffers from non-convergence issues and weak generalization [15,18]. An empirical study showed that the parameters of Adam update unstably; its second moment could be out of date, resulting in producing

inappropriate learning rates based on the changes in gradients [19]. The second moment of Adam in training has been found to produce extreme learning rates; the extreme or abnormal learning rates encountered during training cause poor generalization performance of Adaptive methods [20].

Reddi et al. [18] proposed a variant of Adam called AMSGrad to solve the shortcomings of Adam as mentioned above. Although the authors provide a theoretical guarantee of convergence, AMSGrad only shows better results on training data. However, a study further indicated that AMSGrad does not show noticeable improvement over Adam [21].

Another work to fix the problems of Adam is called AdaBound [20]. AdaBound borrows the truncating theory of gradient clipping [22] to constrain the unreasonable learning rates of Adam by two hand-designed asymptotical functions: the upper bound function $\eta_u = final\_lr * (1 + \frac{1}{(1-\gamma)*t})$ and the lower bound function $\eta_l = final\_lr * (1 - \frac{1}{(1-\gamma)*t+1})$. These two functions tend to a constant value that transforms Adam to SGD (M) gradually. The experiments in [20] demonstrate that AdaBound maintains a rapid training speed, like Adam, in the early and achieves a considerable generalization, like SGD (M), in the end. Additionally, the parameters of bound functions are not sensitive and default parameters are recommended for all classification tasks. In contrast, the experiments reported in [23] showed that AdaBound does not outperform Adam and others, the poor performance of AdaBound could be caused by the default parameters because the same bound are recommended for all classification task. Savarese [24] showed that the bound functions strongly affect the optimizer's behavior and might require careful tuning.

A reasonable learning rate range refers to the reasonable learning rates that a current task can converge, the maximum value of this range is considered the largest learning rate that does not cause the disconvergence of training criteria [25]. However, AdaBound is empirically hand-designed for its bounds, which are independent of tasks. So, it remains a problem to AdaBound that its bounds are able to accurately and completely truncate the unreasonable learning rates or not during training, because the unreasonable learning rates of a certain task are not stated explicitly. Therefore, it is imperative to specify the reasonable convergence range of Adam for a certain task to accurately distinguish the unreasonable learning rates, and design the bound functions according to reasonable convergence range to achieve a smooth transformation from Adam to SGD (M) without any unreasonable learning rates.

To address the problem of AdaBound, we proposed a new variant of the adaptive gradient descent method with a convergence range bound, which is called AdaCB. The major contributions of this paper are summarized as follows:

1. The LR range test [26] is used to accurately determine the reasonable convergence learning rate range for Adam instead of empirically hand designed, which can distinguish the unreasonable learning rates for a certain task.
2. Inspired by the clipping method, the learning rates of Adam are constrained by two asymptotical functions: upper bound and lower bound. The upper bound is the asymptotical function that starts from the maximum value of the convergence range and gradually approaches the global fixed learning rate. The lower bound is the asymptotical function that starts from 0, and gradually approaches the global fixed learning rate.
3. We evaluate our proposed algorithm for the image classification task. Smallnet [27], Network IN Network [28], and Resnet [29] are trained on CIFAR10 [30] and CIFAR100 [30] datasets. Experiment results show that our proposed method behaves like an adaptive gradient method with a faster convergence speed, and achieves considerable results like SGD (M).

The rest of our paper is organized as follows. In Section 2, we review the background of traditional optimizers and adaptive gradient optimizers. In Section 3, we introduce our proposed AdaCB. In Section 4, we conduct experiments to verify the effectiveness of our method. In Section 5, we discuss the advantages and disadvantages of our proposed optimizer. In Section 6, we summarize the paper.

2. Related Works

Training neural networks are about tuning the weights with the lowest loss function by learning algorithms (gradient descent methods) to produce an accurate model for classification or prediction [31]. In general, there are two types of gradient descent methods: stochastic gradient descent methods and adaptive gradient descent methods.

2.1. Stochastic Gradient Descent (SGD)

SGD is the primary and dominant algorithm for CNN models. The SGD updates the weights with a constant learning rate, which is shown in the following Equation (1). The θ_t stands for weights at the step t and θ_{t+1} stands for the updated weights after that step t. The g_t denotes the gradients for corresponding weights at the step t and α stands for the global learning rate, which is often set from 0.1, 0.01, and 0.001. SGD takes a long time for training and is sensitive to learning rates. For the sake of clarifying, the variables of θ_{t+1}, θ_t, g_t, and t stand for updated weights, current weights, current gradients, and current step respectively for the rest of the equations.

The general updating rule for weights in SGD:

$$\theta_{t+1} = \theta_t - \alpha \times g_t \tag{1}$$

2.2. SGD with Momentum

Inspired by the idea of momentum in physics, Poyak et al. [32] proposed a variant of SGD called SGD with momentum, shortly named SGD (M). The cumulative momentum speed v_t is introduced to affect the updating direction and v_{t-1} denotes the last corresponding indicator. The basic idea is that the direction and magnitude of the previous gradients contribute to the step size in the current step. As a result, the training speed is slightly improved.

The general updating rule for weights in SGD (M):

$$\theta_{t+1} = \theta_t + v_t \tag{2}$$

Calculating the cumulative momentum speed:

$$v_t = \gamma \times v_{t-1} - \alpha \times g_t \tag{3}$$

In Equations (2) and (3), the γ is the momentum term that determines the influence of previous gradients on the current update, which is set as close to 1 as possible. The α is a global learning rate like vanilla SGD.

2.3. AdaGrad

However, SGDs scale the gradients in all directions with a constant learning rate, ignoring the sight of the difference between each parameter is not ideal. Therefore, the adaptive gradient descent methods are proposed. In such a situation, the parameter with the critical update should take a more significant step in the gradient direction than the parameter with a less critical update. In other words: to speed up the learning process, every parameter to be updated should have its learning rate in every step.

The first try is AdaGrad. This method calculates the individual learning rates for different parameters, given by a constant value dividing the root of the sum of the square of the past gradients from step 0 to t. The updating rule is defined as following Equation (4).

The general updating rule for weights in AdaGrad:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\sum_1^t g_t^2 + \varepsilon}} \times g_t \tag{4}$$

where the α is a constant value, which is given by 0.001 in practice, and epsilon ε is 1×10^{-8} to avoid the zero of the denominator. The adaptive learning rates are determined by

$\frac{\alpha}{\sqrt{\sum_1^t g_t^2+\varepsilon}}$. However, the denominator is the sum of squared gradients, which can lead to a monotonic increase. Therefore, the learning rates decrease monotonically, leading to slow updating or non-updating circumstances.

2.4. RMSProp

To address the problem of AdaGrad, RMSProp is developed. Instead, RMSProp uses an exponential moving average of square gradients, which is donated to v_t, and v_{t-1} is the last corresponding indicator. As a result, the denominator is not monotony increasing. Equations (5) and (6) show the updating rules. The momentum term of EMA γ is given by 0.9. The constant value α is 0.001 in practice, and the epsilon ε is 1×10^{-8} to avoid the zero of the denominator.

The general updating rule for weights in RMSProp:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{v_t} + \varepsilon} \times g_t \tag{5}$$

Calculating the EMA indicator of square gradients:

$$v_t = \gamma \times v_{t-1} + (1 - \gamma) \times g_t^2 \tag{6}$$

2.5. Adam

The Adaptive Moment Estimation (Adam) algorithm is another adaptive gradient method proposed by Kingma [13], which is widely used in CNNs [33]. It is an extension of RMSprop. Like RMSprop, an Exponential Moving Average of squared gradients v_t is used, and v_{t-1} stands as the last corresponding indicator. Additionally, an Exponential Moving Average of gradients m_t is included, and m_{t-1} is the last corresponding indicator. This new value resembles the average direction of the gradients to avoid non-updating cases caused by zero value of gradient. The authors also employ two bias corrections for these two EMA indicators, which are denoted as $\hat{m}_t$ and $\hat{v}_t$. As a result, they could obtain proper estimations for both denominator and gradients in the early training stage. The formal notations are shown as follows Equations (7) to (11). The variables of β_1 and β_2, which determine the memory time of that two EMA indicators. The authors suggest standard values of β_1 for 0.9, and β_2 for 0.999. The value of α is set as 0.001 in practice, and ε is 1×10^{-8}.

The general updating rule for weights in Adam:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \varepsilon} \times \hat{m}_t \tag{7}$$

Calculating the EMA indicator of gradients:

$$m_t = \beta_1 \times m_{t-1} + (1 - \beta_1) \times g_t \tag{8}$$

Calculating the bias correction for EMA indicator of gradient:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{9}$$

Calculating the EMA indicator of square gradients:

$$v_t = \beta_2 \times v_{t-1} + (1 - \beta_2) \times g_t^2 \tag{10}$$

Calculating the bias correction for EMA indicator of square gradient:

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{11}$$

2.6. AMSGrad

In the recent paper "On the convergence of Adam and beyond" [18], the authors indicated that Adam might suffer from non-convergence issues. Long-term memory should be emphasized to reduce the influence of non-informative gradients and eliminate unreasonable large learning rates. Therefore, the AMSGrad is proposed.

The AMSGrad differs at two points from the Adam algorithm. The first difference is that the AMSGrad only uses the EMA indicator of gradients m_t and EMA predictor of square gradients v_t, and omits the bias corrections for both $\hat{m}_t$ and $\hat{v}_t$. Note that m_{t-1} and v_{t-1} are the last corresponding EMA indicators. The second difference concerning Adam is that the denominator calculation is replaced by taking the maximum value $\hat{v}_t$ between the last indicator v_{t-1} and the current indicator v_t. In this way, a sort of long-term memory is incorporated into large gradients, which results in a non-increasing step size and avoids the pitfalls of Adam. However, the denominator increases monotonically because only the biggest variable is concerned, leading the slow updating or non-updating circumstances like AdaGrad. The formal notations of AMSGrad are described as follows Equations (12) to (15). The parameter setting is the same as Adam, where α is 0.001, β_1 is 0.9, and β_2 is 0.999. The epsilon ε is 1×10^{-8}.

The general updating rule for weights in AMSGrad:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \varepsilon} \times m_t \tag{12}$$

Calculating the EMA indicator of gradients:

$$m_t = \beta_1 \times m_{t-1} + (1 - \beta_1) \times g_t \tag{13}$$

Calculating the EMA indicator of square gradients:

$$v_t = \beta_2 \times v_{t-1} + (1 - \beta_2) \times g_t^2 \tag{14}$$

Taking the maximum operation for the denominator:

$$\hat{v}_t = max(v_{t-1}, v_t) \tag{15}$$

2.7. AdaBound

AMSGrad designed a strategy of non-increasing learning rates to tackle extreme learning rate problems. However, recent work has pointed out AMSGrad does not show evident improvement over Adam [21]. Luo et al. [20] speculate that Adam's extremely large and small learning rates are likely to account for weak generalization ability. To this end, they proposed AdaBound which implemented a gradual transformation from Adam to SGD (M) by employing dynamic bounds to clip extreme ones. However, AdaBound only tackles the extreme learning rates at the end. AdaBound is described in the following Equations (16) to (23).

The general updating rule for weights in AdaBound:

$$\theta_{t+1} = \theta_t - \hat{\eta}_t \times \hat{m}_t \tag{16}$$

Calculating the EMA indicator of gradients:

$$m_t = \beta_1 \times m_{t-1} + (1 - \beta_1) \times g_t \tag{17}$$

Calculating the bias correction for EMA indicator of gradient:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{18}$$

Calculating the EMA predictor of square gradient:

$$v_t = \beta_2 \times v_{t-1} + (1 - \beta_2) \times g_t^2 \tag{19}$$

Calculating the bias correction for EMA predictor of square gradient:

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{20}$$

Setting the lower bound function for Adam:

$$\eta_l = final\_lr \times (1 - \frac{1}{(1-\gamma) \times t + 1}) \tag{21}$$

Setting the upper bound function for Adam:

$$\eta_u = final\_lr \times (1 + \frac{1}{(1-\gamma) \times t}) \tag{22}$$

Constraining the learning rates of Adam in these two bounds:

$$\hat{\eta}_t = Clip(\frac{\alpha}{\sqrt{\hat{v}_t} + \varepsilon}, \eta_l, \eta_u) \tag{23}$$

where the η_l is the lower bound and η_u is the upper bound, and $final\_lr$ is considered as a global constant learning rate to the end. The value γ controls the transition speed from Adam to SGD (M). The authors do the grid search on these two parameters, and they are not sensitive to accuracy. The γ and the $final\_lr$ are given to 0.999 and 0.1 for all tasks, respectively. The rest of the parameters are the same as Adam, which mention in the last Section 2.5. In this setting, the bound range is empirically hand-designed without data information, which is an exhausting way. As a result, it is unknown whether bounds of AdaBound can properly truncate the unreasonable learning rates, because the unreasonable learning rates of a certain task are not state clearly. Intuitively, AdaBound uses the same bounds for all tasks because parameters are not sensitive, which may hurt the generalization performance of the algorithm [23].

2.8. Differences and Similarities between Algorithms

The algorithms described above are considered the adaptive version of SGD because some terms or techniques are mutually used. The general updating rules of them can be described in the following Equation (24).

$$\theta_{t+1} = \theta_t - learning\_rate \times gradient \tag{24}$$

Most algorithms can be described as a uniquely used combination of a small number of building blocks in Equations (25) to (28), and they vary in some terms, as shown in Table 1.

Table 1. Differences and similarities between gradient descent algorithms.

Algorithm	Learning_Rate	Gradient
SGD	0.001, 0.01, 0.1	g_t
AdaGrad	$\frac{0.001}{\sqrt{\sum_1^t g_t^2}+\varepsilon}$	g_t
RMSProp	$\frac{0.001}{\sqrt{v_t}+\varepsilon}$	g_t
Adam	$\frac{0.001}{\sqrt{\hat{v}_t}+\varepsilon}$	$\hat{m}_t$
AMSGrad	$\frac{0.001}{\sqrt{max(v_t,v_{t-1})}+\varepsilon}$	m_t
AdaBound	$Clip(\frac{\alpha}{\sqrt{\hat{v}_t}+\varepsilon}, \eta_l, \eta_u)$	$\hat{m}_t$

Calculating the EMA indicator of gradient:

$$m_t = \beta_1 \times m_{t-1} + (1-\beta_1) \times g_t \tag{25}$$

Calculating the EMA indicator of square gradient:

$$v_t = \beta_2 \times v_{t-1} + (1-\beta_2) \times g_t^2 \tag{26}$$

Calculating the bias correction for EMA indicator of gradient:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{27}$$

Calculating the bias correction for EMA indicator of square gradient:

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{28}$$

3. Proposed Algorithm

Inspired by Adam and Adabound, we propose a new optimizer with convergence range-bound named AdaCB. Firstly, the convergence range of Adam for a particular task is specified by the learning rate range test (LR test). Secondly, two bound functions named upper bound and lower bound are designed to clip the learning rates of Adam in this convergence range. The upper bound starts from the maximum value of the convergence range and approaches the global fixed learning rate asymptotically; the lower bound starts from zero value and approaches the global fixed learning rate asymptotically. In this setting, AdaCB behaves like Adam in the early stage and gradually transforms to SGD in the end, thus, extreme learning rates are considered during the whole stage because the bounds are specified.

3.1. Specify the Convergence Range for Adam

LR test is a helpful way to determine the reasonable convergence range of learning rate for a new model or dataset. LR test wraps the update part of an optimizer to train a model for several epochs and makes the learning rate gradually increase from small to large ones, then the acceptable range learning rate can be approximately estimated by the curve of the loss function. When the loss keeps decreasing, it means the current learning rate is acceptable, when the loss starts rising, it means the current learning rate is the proper largest learning rate the model can be converged.

Adam automatically adjusts the learning rates for different parameters by the formula $\frac{\alpha}{\sqrt{\hat{v}_t}+\varepsilon}$, which is always changed during training. We wrap the update part of Adam into the LR test as an individual value and make the value increase from small to large. When the loss starts rising, this current learning rate can be considered the maximum learning rate at which the model can be converged. The global fixed learning rate can be considered

as the proper learning rate that the model can be converged stably. A global fixed learning rate is recommended as the common SGD (M) learning rate in which one order lower than the learning rate where loss is minimum [26].

For example, we use the Smallnet [27] architecture to perform the LR test wrapped with Adam on the CIFAR10 dataset [30] and generate the curve of the loss function, which is shown in the following Figure 1. As we can see in Figure 1, the loss starts increasing when the current learning is approximately 0.5, so 0.5 is considered the maximum value that this particular task can be converged. The SGD (M) learning rate is commonly chosen from {0.1, 0.01, 0.001} [34,35], according to principle of choosing a global fixed learning rate [26], the global fixed value can be 0.01.

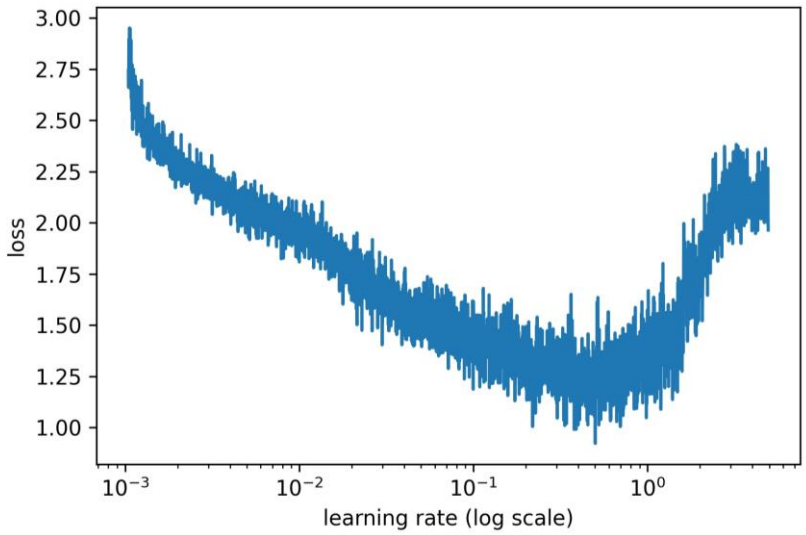

Figure 1. The curve of loss functions by learning rate range test on the CIFAR10 dataset (Smallnet). The x-axis is learning rate, and the y-axis is training loss.

3.2. Design the Bound Functions

Similar to AdaBound, the clipping operation $clip$ is employed on a reasonable learning rate range of Adam obtained by the above Section 3.1, which can be demonstrated in the following Equation (29). $\frac{\alpha}{\sqrt{\hat{v}_t}+\varepsilon}$ is donated as the learning rates of Adam, η_l is the lower bound function that starts from 0 to and gradually approaches the global fixed learning rate, and η_u is the upper bound function that starts from the maximum value of convergence range to the global fixed learning rate.

Clipping the learning rates of Adam in convergence range:

$$Clip(\frac{\alpha}{\sqrt{\hat{v}_t}+\varepsilon}, \eta_l, \eta_u) \tag{29}$$

3.2.1. The Design of Upper Bound

The linear and exponential properties are the common and widely used mechanisms to solve some mathematical problems in practical [36–40]. To tackle the extreme learning rates of Adam during whole training and achieve a smooth transformation from Adam to SGD, the upper bound is designed to be an asymptotical function from the maximum value of convergence range to the global fixed value. In this section, we define the two toy functions for the upper bound η_u, which can be linear and exponential according to basic math definitions [41]. The upper bound functions are shown in the following Equations (30) and (31).

Linear asymptotic of upper bound:

$$\eta_u = max_{lr} - (max_{lr} - glo_{lr}) \times Minimum\left(\frac{t}{T \times M}, 1\right) \tag{30}$$

Exponential asymptotic of upper bound:

$$\eta_u = max_{lr} \times \left(\frac{glo_{lr}}{max_{lr}} \right)^{Minimum(\frac{t}{T \times M}, 1)} \tag{31}$$

where the max_{lr} is the largest learning rate that a task can be converged, the glo_{lr} is the global fixed learning rate that a task can be converged stably, the T is the total training step of a particular task, and t is the current step, M is the certain moment that Adam completely transforms to SGD, and $Minimum$ is the clipping operation to take the minimum value 1 to avoid the range explosion. For the explanation of M, when the M is defined as 0.5, it takes 50 percent of training time to complete the asymptotical process from the largest learning rate to the global fixed learning rate. By this design principle, the upper bound can start from the largest learning rate of the convergence range and gradually approach the fixed learning rate at a certain moment, then $\eta_u = glo_{lr}$.

3.2.2. The Design of Lower Bound

For the lower bound, we design an asymptotical function that starts from zero and gradually approaches the upper bound. However, the upper bound gradually approaches a global fixed value, so the lower bound gradually approaches a global fixed value too. As a result, the lower bound depends on the upper bound, which varies for a particular task that enhances the flexibility of the range. Regarding the exponential design principle [41], the initial value and denominator cannot be zero. Therefore, in this section, we only define the linear function for the lower bound η_l, which is shown in Equation (32).

Linear asymptotic of lower bound:

$$\eta_l = Minimum\left(\frac{t}{T \times M}, 1 \right) \times \eta_u \tag{32}$$

where t is the current training step and T is the total training step, M is the moment that Adam completely transforms to SGD (M), and $Minimum$ is the clipping operation to take the minimum value 1 to avoid the range explosion. By this design principle, the lower bound can start from zero value and gradually approach a global fixed value at a certain moment, then $\eta_l = \eta_u = glo_{lr}$.

3.3. Algorithm Overview for AdaCB

Based on the above analysis, this subsection demonstrates our proposed algorithm named AdaCB, which considers tackling the extreme learning rates during the whole training stage using specified bounds. The details of AdaCB are illustrated in Algorithm 1. The max_{lr} and glo_{lr} can be found by the LR test mentioned in the last Section 3.1. The T is the total training step, which is automatically calculated by $(SampleSize/BatchSize) * Epoch$. The M is a certain moment when Adam completely transforms into SGD (M). For example, if we set it $M = 0.5$, it takes 50 percent of the total training time to complete the transformation between Adam and SGD (M). The β_1, β_2 and α are initial parameters of Adam. The $clip$ is the constraint operation in a range and the η_l and η_u are bound functions designed in the last Sections 3.2.1 and 3.2.2.

In this setting, the AdaCB behaves like Adam in the early stage, and it gradually becomes SGD (M) in the end as two bounds tend to a fixed value. Additionally, the extreme learning rates are tackled during whole training because the bound ranges are specified particularly.

Algorithm 1 AdaCB

Parameters: max_{lr}, glo_{lr}, T, M (parameters of our proposed algorithm)
Initialize: $\beta_1 = 0.9, \beta_2 = 0.999, \alpha = 0.001$ (initial parameters for Adam)
1:set $m_0 = 0, v_0 = 0, t = 0$
2:for $t = 0$ to T do
3: $g_t = \nabla f_t(x_t)$ (calculating the gradients)
4: $m_t = \beta_1 \times m_{t-1} + (1 - \beta_1) \times g_t$ (calculating EMA indicator of gradients)
5: $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$ (calculating bias correction of EMA indicator of gradients)
6: $v_t = \beta_2 \times v_{t-1} + (1 - \beta_2) \times g_t^2$ (calculating EMA indicator of square gradients)
7: $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$ (calculating bias correction of EMA indicator of square gradients)
6: $\hat{\eta}_t = Clip(\frac{\alpha}{\sqrt{\hat{v}_t}+\varepsilon}, \eta_l, \eta_u)$ (clipping the learning rates in the convergence range)
8: $\theta_{t+1} = \theta_t - \hat{\eta}_t \times \hat{m}_t$ (updating the weights)
9:ends for

4. Experiments

In this section, we validate our optimizer AdaCB on the image classification task and compare it to the other three state-of-the-art optimizers, including SGD (M), Adam, and AdaBound. The details of the optimizers are presented in previous Section 2. All the experiments are implemented using Keras with the backend support of Tensorflow, while GeForce GTX 1060 6 GB GPU is utilized to accelerate the computations. Three CNN architectures with various depths and widths are used respectively to train on CIFAR10 and CIFAR100 datasets in a total of 100 epochs, which include Smallnet [27], Network IN Network [28], and 18 layers of Resnet [29]. The batch size is set to 128. We report the average accuracy over five runs for each optimizer across different CNN architectures and datasets.

Data augmentation is a popular regularization technique used to expand the dataset size. However, it is difficult to isolate the impact of the data augmentation from the methodology [42,43]. To exclude complex data expansion that affected the final results, we only divide the original images by 255, and no data augmentation is used. Please note that we are only testing the performance of the optimizers, and we are not testing the effect of the regularization technique.

4.1. Hyper Parameter Tuning

The setting of hyper parameters has a great impact on the performance of the optimization algorithm. Here we describe how we adjust them. We follow Luo et.al. [20] and implement a logarithmical grid search on a large space of learning rate for SGD (M) and take the best results. We set the recommended parameters for another adaptive gradient method. The hyper parameters of each algorithm are adjusted and summarized as follows:

- SGD (M): we roughly tune the learning rate for SGD (M) on a logarithmic scale from {10, 1, 0.1, 0.01, 0.001} and then fine-tune it. We set the momentum parameter to the default value of 0.9. As a result, we set the learning rate of SGD (M) as 0.01 for CIFAR10 and CIFAR100 under Smallnet, Network IN Network, respectively; and 0.1 as the learning rate for CIFAR10 and CIFAR100 under Resnet18.
- Adam: we set the recommended parameters for Adam, where $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$.
- AdaBound: we set the recommended parameters for AdaBound, where $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $final\_lr = 0.1$, and $\gamma = 0.999$.
- AdaCB: We use the same hyper parameter settings for Adam. The max_{lr} and glo_{lr} are determined by the LR test, which is mentioned in Section 3.1. As a result, we set the $max_{lr} = 0.5$ and the $glo_{lr} = 0.01$ on CIFAR10 and CIFAR100 under Smallnet; we set the $max_{lr} = 0.9$ and $glo_{lr} = 0.01$ on CIFAR10 and CIFAR100 under Network IN Network; we set the $max_{lr} = 5$ and $glo_{lr} = 0.1$ on CIFAR10 and CIFAR100 under Resnet18. The total training step is $(SampleSize/BatchSize) * Epoch$. For more, we

will conduct an empirical study to obtain the proper value for transformation speed and asymptotical behavior of the bound function.

4.2. Empirical Study on Behaviors of Bound Function and Transformation Speed

In the previous Section 3.2.1, we propose a linear and exponential asymptotic function for the upper bound and a parameter for transformation speed. To investigate the proper behaviors of bound function and transformation speed M, we conduct a quick analysis with the Smallnet on CIFAR10 dataset, in which the moment M is chosen in {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1}. The results are shown in Table 2. The experiment results show that linear behavior is better than exponential behavior intuitively, and the best result is obtained when $M = 0.6$. For the sake of clarifying, we use linear behavior for the upper bound function and $M = 0.6$ in the rest of the experiments.

Table 2. AdCB with different asymptotic behavior and transformation speed.

M	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Linear	0.7966	0.8189	0.8162	0.8233	0.8274	**0.8299**	0.8243	0.8262	0.8295	0.8272
exponential	0.7849	0.8006	0.8144	0.8220	0.8243	0.8191	0.8217	0.8254	0.8247	**0.8281**

Note: The best accuracy with proper asymptotic behavior and transformation speed is in bold.

4.3. Results of Smallnet on CIFAR10 and CIFAR100

Figures 2 and 3 show the learning curve of testing accuracy for each gradient descent method on CIFAR10 and CIFAR100 under Smallnet architecture. We find that Adam has a faster convergence speed in the early stage, but it finally produces the worst test accuracy. Despite of slow convergence speed for SGD (M) in the early stage, SGD (M) gradually outperforms Adam and AdaBound with increasing training steps. AdaBound has a comparable convergence speed with SGD (M), but the testing accuracy is worse than SGD (M). Our proposed algorithm has a comparable convergence speed with Adam, and it achieves the highest testing accuracy. The detail of testing accuracy for each optimizer on CIFAR10 and CIFAR100 under Smallnet architecture is shown in Table 3. According to Table 3, AdaCB achieves the best accuracy over five runs, followed by SGD (M), AdaBound, and Adam.

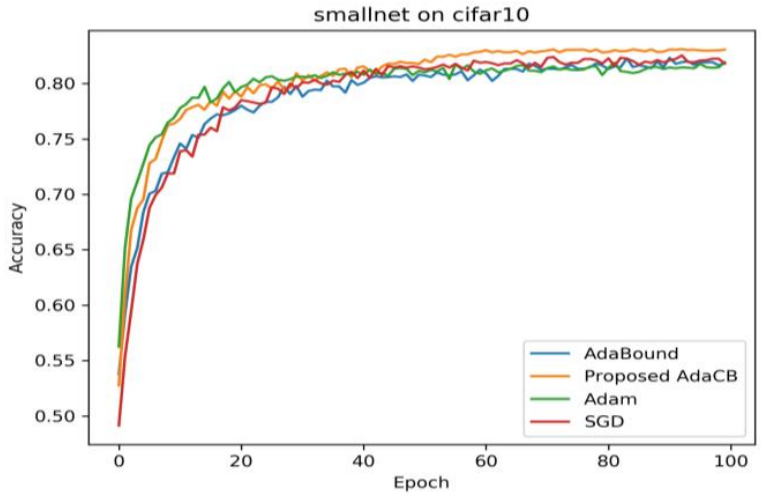

Figure 2. The learning curve of testing accuracy on CIFAR10 under Smallnet.

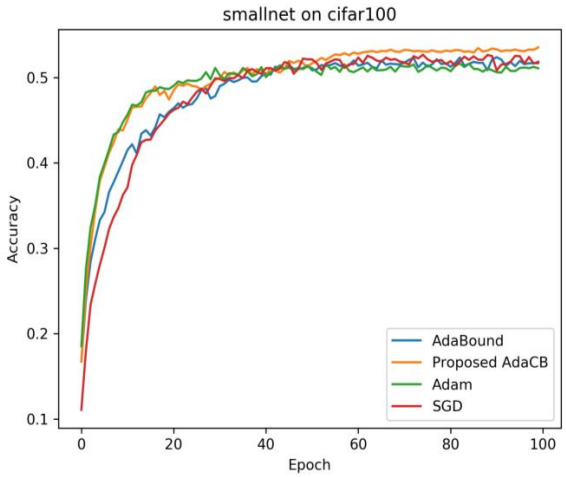

Figure 3. The learning curve of testing accuracy on CIFAR100 under Smallnet.

Table 3. The performance of optimizers on cifar10 and cifar100 under smallnet.

Optimizer	CIFAR10	CIFAR100
AdaBound	82.39% ± 0.13%	52.44% ± 0.3%
AdaCB	**82.76% ± 0.23%**	**53.29% ± 0.19%**
Adam	81.95% ± 0.37%	51.35% ± 0.28%
SGD(M)	82.47% ± 0.24%	52.76% ± 0.16%

Note: The best accuracy on cifar10 and cifar100 under smallnet is in bold.

4.4. Results of Network in Network on CIFAR10 and CIFAR100

As we can see in Figure 4, AdaBound has the worst convergence speed in the early stage and produces the worst testing accuracy on CIFAR10 under Network IN Network architecture. SGD (M) has a better convergence speed than AdaBound, and it finally beat AdaBound and Adam. AdaCB is as faster as Adam in the early stage and achieves the highest testing accuracy in the end. From Figure 5, we can find that Adam has the fastest convergence speed in the early stage and the worst testing accuracy in the end. AdaCB is second only to Adam in convergence speed in the early stage, and it finally achieves the best testing accuracy on CIFAR100 under Network IN Network architecture. Adabound has a comparable convergence speed with SGD (M) before 20 epochs, and SGD (M) wins Adabound eventually. Table 4 demonstrates the performance of each optimizer on CIFAR10 and CIFAR100 under Network IN Network.

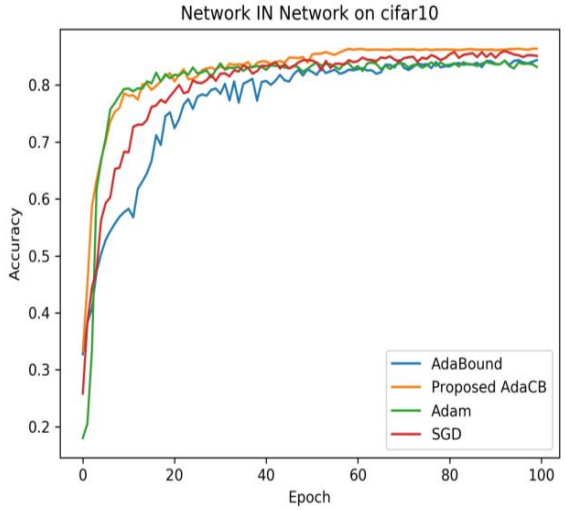

Figure 4. The learning curve of testing accuracy on CIFAR10 under NIN.

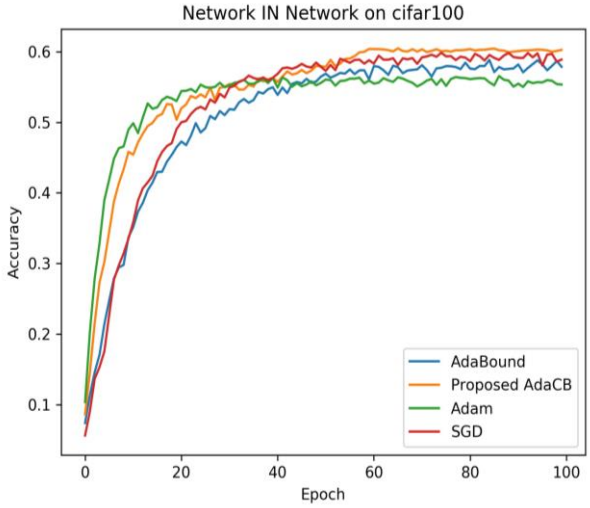

Figure 5. The learning curve of testing accuracy on CIFAR100 under NIN.

Table 4. The performance of optimizers on cifar10 and cifar100 under NIN.

Optimizer	CIFAR10	CIFAR100
AdaBound	84.32% ± 0.34%	58.31% ± 0.36%
AdaCB	**86.24%± 0.22%**	**60.19%± 0.25%**
Adam	84.55% ± 0.37%	56.41% ± 0.47%
SGD(M)	85.74% ± 0.26%	58.86% ± 0.6%

Note: The best accuracy on cifar10 and cifar100 under NIN is in bold.

4.5. Results of Resnet18 on CIFAR10 and CIFAR100

As we can see in Figures 6 and 7, the testing accuracy of Adam is fluctuating on CIFAR10 and CIFAR100 under Resnet18 architecture. Adabound produces a smooth curve of testing accuracy, but its testing accuracy is the worst. AdaCB achieves the best testing accuracy, followed by SGD (M), Adam, and AdaBound. The results are shown in Table 5.

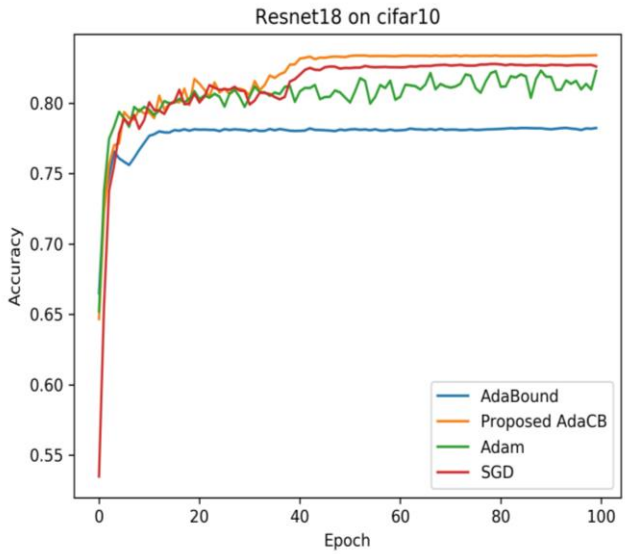

Figure 6. The learning curve of testing accuracy on CIFAR10 under Resnet18.

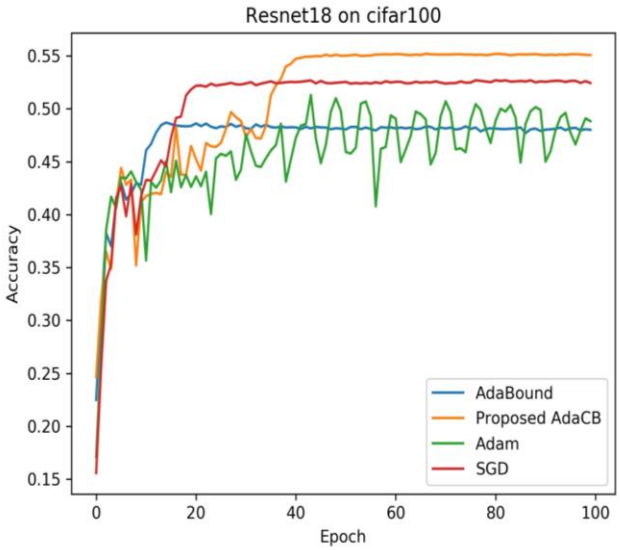

Figure 7. The learning curve of testing accuracy on CIFAR100 under Resnet18.

Table 5. The performance of optimizers on cifar10 and cifar100 under Resnet18.

Optimizer	CIFAR10	CIFAR100
AdaBound	78.72% ± 0.4%	48.53% ± 1%
AdaCB	**83.24%± 0.29%**	**55.04%± 0.4%**
Adam	82.32% ± 0.51%	51.54% ± 0.16%
SGD(M)	82.42% ± 0.73%	51.97% ± 0.54%

Note: The best accuracy on cifar10 and cifar100 under Resnet18 is in bold.

5. Discussion

The experimental outcomes obtained from the last section have shown that AdaCB outperformed the other optimizers. Compared to Adam, our proposed method employs the two bounds to clip the learning rates of Adam, so the effect of unreasonable learning rates is alleviated during training. Additionally, compared to AdaBound, the bound functions of our method are designed based on reasonable convergence range obtained by the LR test instead of empirically hand designed, which can accurately truncate the unreasonable learning rates for a certain task. Moreover, our proposed method is faster than SGD (M) with convergence speed, because our method generates the feature of adaptive learning rates in the early stage. AdaCB integrates both advantages of the adaptive gradient descent method and SGD (M), which can behave like Adam in the early stage with faster convergence speed and become SGD (M) in the end with considerable accuracy. However, our method also has shortcomings. Our method relies on the LR test to pre-specify the convergence bound range, which takes a bit of time to train the current task with a few epochs to calculate the starting point and end point of the convergence range. In future work, we will investigate to determine the convergence bound range adaptively rather than pre-specify with the LR test, so more time can be saved.

6. Conclusions

In this paper, we proposed a new optimizer called AdaCB, which employs the convergence range bound on their effective learning rates. We compare our proposed method with Adam, AdaBound, and SGD (M) on CIFAR 10 and CIFAR 100 datasets across Smallnet, Network IN Network, and Resent. The outcomes indicate that our method outperforms other optimizers on CIFAR10 and CIFAR100 datasets with accuracies of (82.76%, 53.29%), (86.24%, 60.19%), and (83.24%, 55.04%) on Smallnet, Network IN Network and Resnet, respectively. Moreover, the proposed optimizer enjoys the convergence speed of Adam and generates considerable results like SGD (M).

Author Contributions: Conceptualization, X.L.; methodology, X.L.; software, X.L.; validation, X.L., S.S., A.A., and S.A.S.; formal analysis, X.L., S.S., A.A., and S.A.S.; investigation, X.L., S.S., A.A., and S.A.S.; resources, X.L., S.S., A.A., and S.A.S.; data curation, X.L., S.S., A.A., and S.A.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Universiti Kebangsaan Malaysia (UKM), grant nos. GPK-4IR-2020-022 and TAP-K008010.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The selected dataset in this study is available on this link: https://www.cs.toronto.edu/~kriz/cifar.html (accessed on 20 March 2022).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Shi, C.; Zhang, X.; Sun, J.; Wang, L. Remote Sensing Scene Image Classification Based on Self-Compensating Convolution Neural Network. *Remote Sens.* **2022**, *14*, 545. [CrossRef]
2. Ye, A.; Zhou, X.; Miao, F. Innovative Hyperspectral Image Classification Approach Using Optimized CNN and ELM. *Electronics* **2022**, *11*, 775. [CrossRef]
3. Bansal, M.; Kumar, M.; Sachdeva, M.; Mittal, A. Transfer learning for image classification using VGG19: Caltech-101 image data set. *J. Ambient Intell. Humaniz. Comput.* **2021**, *12*, 1–12. [CrossRef] [PubMed]
4. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788. [CrossRef]
5. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767.
6. Zhao, Z.Q.; Zheng, P.; Xu, S.T.; Wu, X. Object Detection With Deep Learning: A Review. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 3212–3232. [CrossRef]
7. Zhang, N.; Wei, X.; Chen, H.; Liu, W. FPGA implementation for CNN-based optical remote sensing object detection. *Electronics* **2021**, *10*, 282. [CrossRef]
8. Robbins, H.; Monro, S. A Stochastic Approximation Method. *Ann. Math. Stat.* **1951**, *22*, 400–407. [CrossRef]
9. Ding, J.; Ren, X.; Luo, R.; Sun, X. An adaptive and momental bound method for stochastic learning. *arXiv* **2019**, arXiv:1910.12249.
10. Gupta, N.; Gupta, S.K.; Pathak, R.K.; Jain, V.; Rashidi, P.; Suri, J.S. Human activity recognition in artificial intelligence framework: A narrative review. *Artif. Intell. Rev.* **2022**, *55*, 4755–4808. [CrossRef]
11. Duchi, J.; Hazan, E.; Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* **2011**, *12*, 2121–2159.
12. Tieleman, T.; Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA Neural Netw. Mach. Learn.* **2012**, *4*, 26–31.
13. Kingma, D.P.; Ba, J.L. Adam: A method for stochastic optimization. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015—Conference Track Proceedings, San Diego, CA, USA, 7–9 May 2015.
14. NIST/Sematech Engineering Statistics Handbook. National Institute of Standards and Technology. 2001. Available online: https://www.itl.nist.gov/div898/handbook (accessed on 12 June 2022).
15. Wilson, A.C.; Roelofs, R.; Stern, M.; Srebro, N.; Recht, B. The marginal value of adaptive gradient methods in machine learning. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017.
16. Sinha, N.; Karjee, P.; Agrawal, R.; Banerjee, A.; Pradhan, C. COVID-19 Recommendation System of Chest X-Ray Images Using CNN Deep Learning Technique with Optimizers and Activation Functions. In *Studies in Computational Intelligence*; Springer: Berlin/Heidelberg, Germany, 2022; Volume 963. [CrossRef]
17. Chen, L.; Bei, L.; An, Y.; Zhang, K.; Cui, P. A Hyperparameters automatic optimization method of time graph convolution network model for traffic prediction. *Wirel. Netw.* **2021**, *27*, 4411–4419. [CrossRef]
18. Reddi, S.J.; Kale, S.; Kumar, S. On the convergence of Adam and beyond. In Proceedings of the 6th International Conference on Learning Representations, ICLR 2018—Conference Track Proceedings, Vancouver, BC, Canada, 30 April–3 May 2018.
19. Shazeer, N.; Stern, M. Adafactor: Adaptive learning rates with sublinear memory cost. In Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholm, Sweden, 10–15 July 2018.
20. Luo, L.; Xiong, Y.; Liu, Y.; Sun, X. Adaptive gradient methods with dynamic bound of learning rate. *arXiv* **2019**, arXiv:1902.09843.
21. Chen, X.; Liu, S.; Sun, R.; Hong, M. On the convergence of a class of Adam-type algorithms for non-convex optimization. In Proceedings of the 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, 6–9 May 2019.
22. Pascanu, R.; Mikolov, T.; Bengio, Y. On the difficulty of training recurrent neural networks. In Proceedings of the 30th International Conference on Machine Learning, ICML 2013, New Orleans, LA, USA, 6–9 May 2013.

23. Savarese, P.; McAllester, D.; Babu, S.; Maire, M. Domain-independent dominance of adaptive methods. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 19–25 June 2021. [CrossRef]
24. Savarese, P. On the Convergence of AdaBound and its Connection to SGD. *arXiv* **2019**, arXiv:1908.04457.
25. Bengio, Y. Practical recommendations for gradient-based training of deep architectures. *Neural Netw. Tricks Trade* **2012**, *7700*, 437–478. [CrossRef]
26. Smith, L.N. Cyclical learning rates for training neural networks. In Proceedings of the 2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017, Santa Rosa, CA, USA, 24–31 March 2017. [CrossRef]
27. Qiu, S.; Xu, X.; Cai, B. FReLU: Flexible Rectified Linear Units for Improving Convolutional Neural Networks. In Proceedings of the International Conference on Pattern Recognition, Beijing, China, 20–24 August 2018; Volume 2018-August. [CrossRef]
28. Lin, M.; Chen, Q.; Yan, S. Network in network. In Proceedings of the 2nd International Conference on Learning Representations, Banff, AB, Canada, 14–16 April 2014; pp. 1–10.
29. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognitio, Las Vegas, NV, USA, 26 June–1 July 2016; Volume 2016-December. [CrossRef]
30. Krizhevsky, A. *Learning Multiple Layers of Features from Tiny Images*; Technical report; Science Department, University of Toronto: Toronto, ON, Canada, 2009.
31. Dogo, E.M.; Afolabi, O.J.; Nwulu, N.I.; Twala, B.; Aigbavboa, C.O. A Comparative Analysis of Gradient Descent-Based Optimization Algorithms on Convolutional Neural Networks. In Proceedings of the International Conference on Computational Techniques, Electronics and Mechanical Systems, Belgaum, India, 21–22 December 2018. [CrossRef]
32. Polyak, B.T. Some methods of speeding up the convergence of iteration methods. *USSR Comput. Math. Math. Phys.* **1964**, *4*, 1–17. [CrossRef]
33. Bharanidharan, N.; Rajaguru, H. Dementia MRI image classification using transformation technique based on elephant herding optimization with Randomized Adam method for updating the hyper-parameters. *Int. J. Imaging Syst. Technol.* **2020**, *31*, 1221–1245. [CrossRef]
34. Yang, X. Kalman optimizer for consistent gradient descent. In Proceedings of the ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing—Proceedings, Toronto, ON, Canada, 6–11 June 2021; Volume 2021-June. [CrossRef]
35. Li, J.; Yang, X. A Cyclical Learning Rate Method in Deep Learning Training. In Proceedings of the 2020 International Conference on Computer, Information and Telecommunication Systems, Hangzhou, China, 5–7 October 2020. [CrossRef]
36. Alagic, M.; Palenz, D. Teachers Explore Linear and Exponential Growth: Spreadsheets as Cognitive Tools. *J. Technol. Teach. Educ.* **2006**, *14*, 633–649.
37. Gohivar, R.K.; Yadav, S.K.; Koirala, R.P.; Adhikari, D. Study of artifacts in thermodynamic and structural properties of Li–Mg alloy in liquid state using linear and exponential models. *Heliyon* **2021**, *7*, e06613. [CrossRef]
38. Khan, M.F.; Hasan, M.G.; Quddoos, A.; Fügenschuh, A.; Hasan, S.S. Goal programming models with linear and exponential fuzzy preference relations. *Symmetry* **2020**, *12*, 934. [CrossRef]
39. Kumar, R.S.V.; Kumar, R.N.; Sowmya, G.; Prasannakumara, B.C.; Sarris, I.E. Exploration of Temperature Distribution through a Longitudinal Rectangular Fin with Linear and Exponential Temperature-Dependent Thermal Conductivity Using DTM-Pade Approximant. *Symmetry* **2022**, *14*, 690. [CrossRef]
40. Oguejiofor, G.C. Modeling of linear and exponential growth and decay equations and testing them on pre- and post-war-coal production in nigeria: An operations research approach. *Energy Sources Part B Econ. Plan. Policy* **2010**, *5*, 116–125. [CrossRef]
41. Inigo, M.; Jameson, J. College Mathematics for Everyday. Available online: https://math.libretexts.org/Bookshelves/Applied_Mathematics/Book%3A_College_Mathematics_for_Everyday_Life_(Inigo_et_al) (accessed on 28 May 2022).
42. Hou, S.; Liu, X.; Wang, Z. DualNet: Learn Complementary Features for Image Recognition. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; Volume 2017-October. [CrossRef]
43. Murthy, V.N.; Singh, V.; Chen, T.; Manmatha, R.; Comaniciu, D. Deep Decision Network for Multi-class Image Classification. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; Volume 2016-December. [CrossRef]