Incremental PID Controller-Based Learning Rate Scheduler for Stochastic Gradient Descent

Zenghui Wang and Jun Zhang

***Abstract*— As we all know, the learning rate plays a vital role in deep neural network (DNN) training. This study introduces an incremental proportional-integral-derivative (PID) controller widely used in automatic control as a learning rate scheduler for stochastic gradient descent (SGD). To automatically calculate the current learning rate, we utilize feedback control to determine the relationship between training losses and learning rates, named incremental PID learning rates, which include PID-Base and PID-Warmup. The new schedulers reduce the dependence on the initial learning rate and achieve higher accuracy. Compared with multistep learning rates (MSLR), cyclical learning rates (CLR), and SGD with warm restarts (SGDR), incremental PID learning rates based on feedback control obtain higher accuracy on CIFAR-10, CIFAR-100, and Tiny-ImageNet-200. We believe that our methods can improve the performance of SGD.**

***Index Terms*— Feedback control, incremental proportional-integral-derivative (PID) controller, learning rate scheduler, stochastic gradient descent (SGD).**

I. Introduction

DEEP neural networks (DNNs) have become popular tools for computer vision (CV) [1], [2], [3] and natural language processing (NLP) [4], [5]. Usually, training a DNN on large-scale datasets is a time-consuming process. An optimizer with an appropriate learning rate is critical to accelerate training and improve accuracy. Deep learning optimizers can be categorized into two groups. One is hand-tuned learning rate optimizers, including stochastic gradient descent (SGD) [6], SGD with momentum (SGDM) [7], and Nesterov accelerated gradient (NAG) [8]. The other one is adaptive learning rate optimizers, such as AdaGrad [9], AdaDelta [10], and Adam [11]. Although adaptive learning rate optimizers have many advantages, hand-tuned learning rate optimizers are still the mainstream training approach for DNNs, considering their stability and accuracy. For example, state-of-the-art models such as VGG [12], ResNet [1], DenseNet [13], ResNeXt [2], SENet [3], RegNet [14], and RepVGG [15] are optimized by SGDM. WideResNet [16] is optimized by NAG and GoogleNet [17] use asynchronous SGD (ASGD) [7]. Besides, many researchers have also analyzed the reasons for the poor performance of adaptive learning rate optimizers in recent years. Reddi et al. [18] proved that Adam does not converge to the optimal solution even in a simple convex optimization problem. Wilson et al. [19] observed that the generalization performance optimized by adaptive methods is worse than SGD. Therefore, SGD with learning rate schedulers is still the mainstream for DNN training.

Many learning rate schedulers are used to tune optimizers. He et al. [1] used a lower learning rate of 0.01 for warmup training. Goyal et al. [20] proposed a gradual warmup that gradually increases the learning rate from a small to a large value to avoid training instability. The learning rate decay can help to advance the training process when neural networks suffer from stagnation. The multistep learning rate (MSLR) is a commonly used method that reduces learning rates at specified epochs. SGD with warm restarts (SGDR) [21] periodically restarts learning rates. Cyclical learning rates (CLR) [22] change learning rates cyclically between reasonable boundaries. However, these learning rate schedulers do not directly consider the characteristics of different networks. In particular, the internal relationship between updated learning rates and network training is not established. Therefore, we try to adopt feedback control to establish their relationship.

Control theory has inspired many studies of neural networks. Hopfield network [23] is a typical multiple-loop feedback system since the output values of the neurons are fed back. Xu et al. [24] adopted closed-loop control to improve the stability of generative adversarial networks (GANs). Proportional-integral-derivative (PID) controllers are widely used in industrial control. They adjust objective action using the current error, the integral of the past error, and the derivative of the error. A PID controller is incorporated into the SGD-based latent factor analysis model to achieve high-performance [25]. A derivative (D) term is introduced to SGDM to develop a PID optimizer [26], [27]. However, the D term suffers from oscillation and high-frequency noise. Some studies [28], [29] improve the stability of the PID optimizer by improving the D term. Wang et al. [30] used an integral-separated PI controller to solve the problem.

In this study, we use the incremental PID controller as a learning rate scheduler to adjust learning rates. The major contributions are summarized as follows.

1) The incremental PID controller is introduced to adjust the learning rates of SGD. Feedback control is used to establish the relationship between training losses and learning rates directly. Incremental PID learning rates,

Manuscript received 24 July 2021; revised 18 June 2022; accepted 3 October 2022. This work was supported by the National Natural Science Foundation of China under Grant 61872004. *(Corresponding author: Jun Zhang.)*

Zenghui Wang is with the Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, School of Electrical Engineering and Automation, Anhui University, Hefei 230601, China (e-mail: scholarzhwang@163.com).

Jun Zhang is with the School of Artificial Intelligence, Anhui University, Hefei 230601, China (e-mail: junzhang@ahu.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3213677.

Digital Object Identifier 10.1109/TNNLS.2022.3213677

including PID-Base and PID-Warmup, are proposed to reduce the dependence on initial learning rates and achieve higher accuracy.
2) We explain the superiority of the incremental PID learning rates. In addition, the hyperparameters of our methods have physical significance and directly reflect the updated learning rates.
3) The incremental PID learning rates are comprehensively evaluated on CIFAR-10, CIFAR-100, and Tiny-ImageNet-200. Compared with MSLR, CLR, and SGDR, experimental results demonstrate the effectiveness of the proposed methods.

The remainder of this article is organized as follows. Section II describes related works. Section III introduces the proposed incremental PID learning rates. Then, experimental results are presented in Section IV. Section V discusses learning rate schedulers and the PID optimizer. Finally, the conclusion is drawn in Section VI.

II. Related Works

In this section, we review the SGD optimizer, learning rate schedulers, and incremental PID controller.

A. SGD Optimizer

Training a neural network f_θ with parameters θ can be represented as minimizing the distance between the ground-truth label y_i and the predicted output $\hat{y}_i = f_\theta(x_i)$ by a loss function $\mathcal{L}$

$$\min_\theta \; F(\theta) \triangleq \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(y_i, \hat{y}_i) \tag{1}$$

where F is the objective function and x_i is the data point. SGD is an effective tool for high-dimensional optimization problems. It is expressed as

$$\theta_{t+1} = \theta_t - \alpha_t \nabla_\theta F(\theta_t) \tag{2}$$

where θ_t is the parameter at the tth iteration, α_t is the learning rate, and $\nabla_\theta F$ is the gradient of F. Since SGD is performed on a training sample, it introduces random noise that causes oscillation and slows down convergence. In practice, mini-batch SGD is usually used for training. Momentum is an effective method to accelerate SGD and suppress oscillation [7]. SGD with momentum (SGDM) is written as

$$\begin{aligned} m_t &= \beta m_{t-1} + (1-\beta)\nabla_\theta F(\theta_t) \\ \theta_{t+1} &= \theta_t - \alpha_t m_t \end{aligned} \tag{3}$$

where the momentum term β is usually set to 0.9. SGDM accelerates learning by accumulating the exponential moving average of past gradients and has excellent convergence characteristics. However, the drawback of these optimizers is that the learning rate needs to spend much time fine-tuning. Therefore, a learning rate scheduler is used to cooperate with SGD to train deep models.

B. Learning Rate Schedulers

Learning rate schedulers are crucial strategies to accelerate model convergence and achieve better performance. Learning rate decay and warmup are two common strategies. A large learning rate helps accelerate training or escape spurious local minima and then decays it to converge to a local minimum and avoid oscillation [31], [32], [33]. You et al. [34] also found that a large initial learning rate suppresses neural networks from memorizing noisy data while decaying the learning rate enhances the ability of neural networks to learn complex patterns. However, network weights are initialized randomly at the beginning of training, which hinders training convergence. Even worse, exploding gradient problem may happen if the initial learning rate is set too large. The learning rate warmup adopts a gentle manner to avoid network instability in the initial stage. Gotmare et al. [35] revealed that warmup can prevent deeper layers from training instability.

MSLR, CLR, and SGDR are commonly used learning rate schedulers. MSLR decays learning rates at specified epochs. The learning rate α_t of MSLR at the ith update is written as

$$\alpha_t = \alpha_0 \cdot \text{gamma}^i \tag{4}$$

where α_0 is the initial learning rate and *gamma* is the scaling factor. CLR allows learning rates to periodically change between the minimum learning rate boundary $\alpha_{\min}$ and the maximum learning rate boundary $\alpha_{\max}$ in a cycle [22]. It has three policies: *triangular*, *triangular2*, and *exp_range*. The *triangular* policy is defined as

$$\begin{cases} x = \left| \dfrac{i}{\text{stepsize}} - 2\left\lfloor 1 + \dfrac{i}{2\cdot\text{stepsize}} \right\rfloor + 1 \right| \\ \alpha_t = \alpha_{\min} + (\alpha_{\max} - \alpha_{\min}) \cdot \max(0, (1-x)) \end{cases} \tag{5}$$

where stepsize is the half the period or cycle length, $|\cdot|$ is the absolute value function, and $\lfloor\cdot\rfloor$ is the floor function. SGDR [21] restarts the learning rate as follows:

$$\alpha_t = \alpha_{\min} + \frac{1}{2}(\alpha_{\max} - \alpha_{\min})\left(1 + \cos\left(\frac{T_{\text{cur}}}{T_i}\pi\right)\right) \tag{6}$$

where T_{cur} counts how many epochs have been trained since the last restart, and T_i is the restart period. $T_i = T_0 \times T_{\text{mult}}$, here T_{mult} is a factor. CLR and SGDR have many key hyperparameters that require manual fine-tuning. More importantly, the learning rates predetermined by boundaries do not directly consider the characteristics of various network training. Therefore, these tricky methods bring inconvenience to model training.

C. Incremental PID Controller

Feedback control is used to establish the relationship between learning rates and network training. The PID control provides a simple and efficient solution to many real-world control problems [36]. It calculates the control amount u_t based on the proportional (P), integral (I), and derivative (D) terms of error e_t. The PID control rule is written as

$$u_t = K_P\left(e_t + \frac{1}{T_I}\int_0^t e_t dt + T_D\frac{de_t}{dt}\right) \tag{7}$$

where K_P is the proportional gain, T_I is the integral time constant, and T_D is the derivative time constant. The rule must be digitized for computer realization. Equation (7) is transformed into a difference equation and then becomes digital PID control. First of all, the integral and derivative terms are approximated as

$$\int_0^t e_t dt \approx \sum_{i=0}^{k} T e_i, \quad \frac{de_t}{dt} \approx \frac{e_k - e_{k-1}}{T} \tag{8}$$

where k is the sampling time and T is the sampling period. We have a positional PID controller by (7) and (8)

$$u_k = K_P\left(e_k + \frac{T}{T_I}\sum_{i=0}^{k} e_i + T_D\frac{e_k - e_{k-1}}{T}\right) \tag{9}$$

where u_k is the output at the kth sampling time. Next, u_{k-1} is written as

$$u_{k-1} = K_P\left(e_{k-1} + \frac{T}{T_I}\sum_{i=0}^{k-1} e_i + T_D\frac{e_{k-1} - e_{k-2}}{T}\right). \tag{10}$$

We get the PID increment Δu_k

$$\begin{aligned}\Delta u_k &= u_k - u_{k-1} \\ &= K_P(e_k - e_{k-1}) + K_I e_k + K_D(e_k - 2e_{k-1} + e_{k-2})\end{aligned} \tag{11}$$

where K_P is the proportional coefficient, $K_I = (K_P T/T_I)$ is the integral coefficient, and $K_D = ((K_P T_D)/T)$ is the derivative coefficient. At last, the incremental PID controller is defined as

$$u_k = u_{k-1} + \Delta u_k. \tag{12}$$

Compared with the positional PID controller, the incremental PID controller is highly suitable for tuning learning rates because it is only related to the last three errors and has a lower computational cost.

III. Incremental PID Learning Rates

In this section, we introduce an incremental PID controller to establish the relationship between training losses and learning rates. Two methods are proposed: PID-Base and PID-Warmup. We analyze the learning rate update process and hyperparameter roles and settings.

A. Basic Incremental PID Learning Rates

The learning rate is one of the most significant hyperparameters in training neural networks [37]. Usually, researchers attempt to obtain optimal performance by varying learning rate settings. The incremental PID controller can adjust learning rates automatically according to feedback errors. It is fundamentally different from MSLR, CLR, and SGDR.

First of all, we must construct appropriate errors for DNN training. The training loss is chosen to build feedback control due to its easy availability. By (1), the training loss at step t is defined as

$$\text{loss}_t = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_t(y_i, \hat{y}_i). \tag{13}$$

Algorithm 1 PID-Base

Input: K_P, K_D, α_0, training loss $loss$
Output: learning rate α

Initialize: $loss_1, loss_2, loss_3 \leftarrow 0$, $step \leftarrow 128$, $\alpha \leftarrow \alpha_0$, $\alpha_{\max} \leftarrow 10$, $\alpha_{\min} \leftarrow 10^{-8}$
for $iteration = 1, 2, 3 \ldots$ **do**
  $loss_1 \leftarrow loss_1 + loss$
  if $iteration \ \% \ step = 0$ **then**
    $loss_1 \leftarrow loss_1/step$
    $\Delta\alpha \leftarrow K_P(loss_1 - loss_2) + K_D(loss_1 - 2\ loss_2 + loss_3)$
    $\alpha \leftarrow \alpha + \Delta\alpha$
    if $\alpha > \alpha_{\max}$ **then**
      $\alpha \leftarrow \alpha_{\max}$
    else if $\alpha < \alpha_{\min}$ **then**
      $\alpha \leftarrow \alpha_{\min}$
    end if
    $loss_3 \leftarrow loss_2$
    $loss_2 \leftarrow loss_1$
    Set: $loss_1 \leftarrow 0$
    return α
  end if
end for

In control theory, the error is defined as the difference between the ideal and actual values. For deep learning, we define the error e_t as the difference between training loss and desired value 0. The definition can ensure that e_t is positive and avoid cumulative negative values. It is written as

$$e_t = \text{loss}_t - 0 = \text{loss}_t. \tag{14}$$

This is equivalent to adding an inverter in front of the incremental PID controller. It is a crucial step to simplify the controller parameters. According to (11), the learning rate increment $\Delta\alpha_t$ is written as

$$\begin{aligned}\Delta\alpha_t = K_P(\text{loss}_t - \text{loss}_{t-1}) + K_I\text{loss}_t \\ + K_D(\text{loss}_t - 2\text{loss}_{t-1} + \text{loss}_{t-2}).\end{aligned} \tag{15}$$

The $K_I\text{loss}_t$ term may cause $\Delta\alpha_t$ instability because loss_t cannot be determined before training. A large K_I also leads to exploding gradients. Hence, K_I is set to 0 to avoid harming the optimization process. The first and third terms can be smoothed by the average training losses. Finally, $\Delta\alpha_t$ is simplified as

$$\begin{aligned}\Delta\alpha_t = K_P(\text{loss}_t - \text{loss}_{t-1}) \\ + K_D(\text{loss}_t - 2\text{loss}_{t-1} + \text{loss}_{t-2}).\end{aligned} \tag{16}$$

By (12), the incremental PID learning rates α_t are defined as

$$\alpha_t = \alpha_{t-1} + \Delta\alpha_t. \tag{17}$$

According to (2) and (17), we get SGD with incremental PID learning rates

$$\theta_{t+1} = \theta_t - (\alpha_{t-1} + \Delta\alpha_t)\nabla_\theta F(\theta_t). \tag{18}$$

The incremental PID controller updates the current learning rate based on the last three feedback training losses. Since

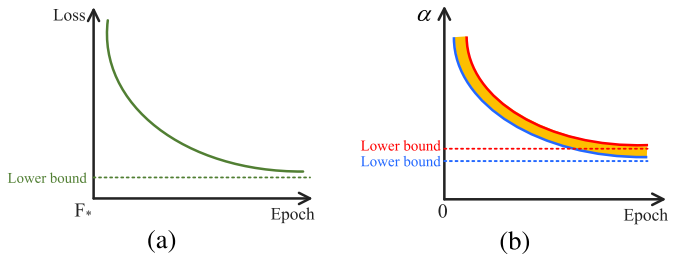

Fig. 1. (a) Assumed training loss curve. (b) Learning rates of PID-Base (yellow band) are sandwiched by two monotonically decreasing curves (red and blue curves).

training loss values vary widely from one batch to another, the noise from training losses and the update frequency of incremental PID learning rates must be considered. The step parameter is used to update learning rates smoothly by averaging training losses. The step means the learning rate is updated every step iteration. For example, the step is set to 128. CIFAR-10 [38] has 50 000 training images and the batch size is 128, then an epoch has $50\,000/128 = 391$ iterations. The learning rate update times is 391/step = 3. The loss_t is the average of step iterations. In addition, limiting the PID output is a common method in industrial control, and it can also be used for learning rates

$$\alpha_t = \begin{cases} \alpha_{\max}, & \alpha_t > \alpha_{\max} \\ \alpha_{\min}, & \alpha_t < \alpha_{\min}. \end{cases} \tag{19}$$

The basic incremental PID learning rate is named PID-Base, and Algorithm 1 is its pseudo-code. We also propose PID-Base with a gradual warmup, named PID-Warmup. We make the incremental PID learning rates perform multiple-step decay at specified epochs to further improve convergence.

B. Principle of Incremental PID Learning Rates

First, we briefly review the SGD convergence analysis proved by Bottou et al. [39], and the key assumptions are as follows.

Assumption 1 [39]: The objective function F: $\mathbb{R}^d \to \mathbb{R}$ is continuously differentiable and has Lipschitz continuous gradient with Lipschitz constant $L > 0$ such that, for all $\{\theta, \bar{\theta}\} \subset \mathbb{R}^d$

$$\left\|\nabla F(\theta) - \nabla F(\bar{\theta})\right\|_2 \le L\left\|\theta - \bar{\theta}\right\|_2.$$

Assumption 2 [39]: The objective function F is bounded below by a scalar. Add additional constraints on the first moment, second moment, and variance of the stochastic gradient $g(\theta_t, \xi_t)$, and ξ_t is the random variable. There exist scalars $\mu_G \ge \mu > 0$, $M \ge 0$, and $M_V \ge 0$ such that, for all $t \in \mathbb{N}$

$$\begin{aligned} \nabla F(\theta_t)^T \mathbb{E}_{\xi_t}[g(\theta_t, \xi_t)] &\ge \mu \|\nabla F(\theta_t)\|_2^2 \\ \left\|\mathbb{E}_{\xi_t}[g(\theta_t, \xi_t)]\right\|_2 &\le \mu_G \|\nabla F(\theta_t)\|_2 \\ \mathbb{E}_{\xi_t}\left[\|g(\theta_t, \xi_t)\|_2^2\right] - \left\|\mathbb{E}_{\xi_t}[g(\theta_t, \xi_t)]\right\|_2^2 &\le M + M_V \|\nabla F(\theta_t)\|_2^2. \end{aligned}$$

Under Assumption 2, the second moment of $g(\theta_t, \xi_t)$ satisfies

$$\mathbb{E}_{\xi_t}\left[\|g(\theta_t, \xi_t)\|_2^2\right] \le M + M_G \|\nabla F(\theta_t)\|_2^2 \tag{20}$$

where $M_G = M_V + \mu_G^2$. If $g(\theta_t, \xi_t)$ is an unbiased estimate of $\nabla F(\theta_t)$, then $\mu_G = \mu = 1$. If there is no noise in $g(\theta_t, \xi_t)$, then $M_G = 1$.

Assumption 3 [39]: The objective function F is strongly convex and there exists a constant $c > 0$ such that, for all $(\bar{\theta}, \theta) \in \mathbb{R}^d \times \mathbb{R}^d$

$$F(\bar{\theta}) \ge F(\theta) + \nabla F(\theta)^T(\bar{\theta} - \theta) + \frac{1}{2}c\left\|\bar{\theta} - \theta\right\|_2^2.$$

Hence, F has a unique minimizer $F_* := F(\theta_*)$, and F is also bounded below by F_*.

Finally, the SGD convergence is proven in Lemma 1.

Lemma 1 [39]: The expected optimality gap for SGD with a fixed learning rate $\bar{\alpha}$, $0 < \bar{\alpha} \le (\mu/LM_G)$ satisfies the inequality for all $t \in \mathbb{N}$

$$\lim_{t \to +\infty} \mathbb{E}[F(\theta_t) - F_*] \le \frac{\bar{\alpha} LM}{2c\mu}.$$

Furthermore, we analyze the incremental PID learning rates based on Bottou et al. [39]. By (16) and (17), loss_t, loss_{t-1}, and loss_{t-2} are initialized to 0, then α_t is rewritten as

$$\begin{aligned} \alpha_t &= \alpha_{t-1} + \Delta\alpha_t \\ &= \alpha_0 + \sum_{i=1}^{t} \Delta\alpha_i \\ &= \alpha_0 + K_P \text{loss}_t + K_D(\text{loss}_t - \text{loss}_{t-1}) \qquad (21a) \\ &= \alpha_0 + (K_P + K_D)\text{loss}_t - K_D \text{loss}_{t-1}. \qquad (21b) \end{aligned}$$

By Lemma 1, Assumption 3, (1), and (13), the training loss loss_t has the following:

$$\lim_{t \to +\infty} \text{loss}_t = F_*. \tag{22}$$

Theorem 1: Under Lemma 1, (21b), and (22), the learning rate α_t updated by the incremental PID learning rates converges to

$$\lim_{t \to +\infty} \alpha_t = \alpha_0 + K_P F_*.$$

Proof 1:

$$\begin{aligned} \lim_{t \to +\infty} \alpha_t &= \lim_{t \to +\infty} \left[\alpha_0 + (K_P + K_D)\text{loss}_t - K_D \text{loss}_{t-1}\right] \\ &= \lim_{t \to +\infty} \left[\alpha_0 + (K_P + K_D)F_* - K_D F_*\right] \\ &= \alpha_0 + K_P F_*. \end{aligned} \tag{23}$$

Thus, α_t converges to a value slightly greater than the initial learning rate.

Theorem 2: Under Lemma 1 and (23), for all $t \in \mathbb{N}$, the expected optimality gap for SGD with incremental PID learning rates satisfies

$$\lim_{t \to +\infty} \mathbb{E}\left[F(\theta_t) - \left(1 + \frac{K_P LM}{2c\mu}\right)F_*\right] \le \frac{\alpha_0 LM}{2c\mu}.$$

Therefore, decaying the initial learning rate can further improve model convergence. Next, we make assumptions about the training loss with step t to analyze the learning rate update process.

Assumption 4: The training loss loss_t is a monotonically decreasing function with step t such that, for $\forall x, y \in \mathbb{N}$, $x < y$, then $\text{loss}_x \ge \text{loss}_y$.

Assumption 5: The training loss loss_t is a bounded function, and there exists an upper bound $L_{\text{ub}} > 0$ and a lower bound.

Assumptions 4 and 5 are based on the optimization objective and SGD convergence analysis [39]. In practice, the training loss curve contains noise, but the main trend is decreasing. For a normal optimization process (no exploding gradients), the bounded training loss is also consistent with training intuition. Using Assumptions 4 and 5, we have

$$0 \leq \text{loss}_t \leq L_{\text{ub}}. \tag{24}$$

Theorem 3: Combining (21b) and (24), the learning rate α_t is sandwiched by two monotonically decreasing functions

$$\alpha_0 + (K_P + K_D)\text{loss}_t - K_D L_{\text{ub}} \leq \alpha_t \leq \alpha_0 + (K_P + K_D)\text{loss}_t.$$

The learning rates of the PID-Base are sandwiched by two monotonically decreasing curves. The assumed training loss curve is shown in Fig. 1(a). Fig. 1(b) shows that PID-Base is limited to the yellow band. Using (21b), we have the following:

$$\begin{aligned} \alpha_1 &= \alpha_0 + (K_P + K_D)\text{loss}_1 > \alpha_0 \\ \alpha_2 &= \alpha_0 + (K_P + K_D)\text{loss}_2 - K_D\text{loss}_1 < \alpha_1. \end{aligned} \tag{25}$$

The α_1 is greater than the initial learning rate α_0, then α_2 is smaller than α_1. An excessive learning rate may cause oscillation or exploding gradients, so the $K_D\text{loss}_{t-1}$ term can avoid these situations and improve training stability. An important attribute of the PID-Base is that the learning rate decays from a large value and then converges around the initial learning rate in a small oscillation. Modern deep models are usually trained with a large initial learning rate and then decay when models come to a training loss plateau. A large learning rate can accelerate training and help networks escape spurious local minima and then decays the learning rate to converge to a local minimum. Li et al. [40] proved that networks trained by a large initial learning rate with annealing can achieve better generalization performance than starting with a small learning rate. Weight decay [41] is an excellent technique to improve generalization performance. The incremental PID learning rates have an effect on weight decay. The objective function with L_2 regularization is expressed as

$$\tilde{F}(\theta_t) = F(\theta_t) + \frac{\lambda}{2}\|\theta_t\|_2^2 \tag{26}$$

where λ defines the rate of the weight decay per step. Next, the gradient of $\tilde{F}(\theta_t)$ is written as

$$\nabla_\theta \tilde{F}(\theta_t) = \nabla_\theta F(\theta_t) + \lambda\theta_t. \tag{27}$$

Equation (2) is rewritten as

$$\begin{aligned} \theta_{t+1} &= \theta_t - \alpha_t \nabla_\theta \tilde{F}(\theta_t) \\ &= \theta_t - \alpha_t(\nabla_\theta F(\theta_t) + \lambda\theta_t) \\ &= (1 - \lambda\alpha_t)\theta_t - \alpha_t \nabla_\theta F(\theta_t). \end{aligned} \tag{28}$$

Finally, combining (17) and (28), we have

$$\theta_{t+1} = (1 - \lambda\alpha_{t-1} - \lambda\Delta\alpha_t)\theta_t - (\alpha_{t-1} + \Delta\alpha_t)\nabla_\theta F(\theta_t). \tag{29}$$

Algorithm 2 PID-Warmup

Input: α_0, $epoch$, warmup epoch w, symmetry axis $axis$
Output: learning rate α

```
Initialize: offset ← 1, axis ← α_0
if epoch < w then
    α_PID-Base ← Algorithm 1: PID-Base
    α ← 2 × offset × axis − α_PID-Base
else
    α ← Algorithm 1: PID-Base
end if
return α
```

Therefore, the $\lambda\Delta\alpha_t$ term can affect weight decay. The attribute that PID-Base decays from a large learning rate has a regularization effect. It helps to improve generalization abilities and alleviate overfitting slightly.

We explain the incremental PID learning rates through a flowchart. Fig. 2 shows the connection between feedback control system and DNN training [27]. The gradient $\nabla_\theta F(\theta)$ is the incarnation of the error, SGDM is regarded as a controlled object, and backpropagation is considered as feedback. In the red part of Fig. 2, the incremental PID controller introduces a new feedback loop to tune learning rates. The training loss is used to automatically compute the current learning rate of SGDM, which is essentially different from those pre-calculated learning rate algorithms such as CLR and SGDR. Therefore, this loop can improve the robustness of network training and obtain higher accuracy.

C. Incremental PID Learning Rates With Gradual Warmup

All weight parameters are random values at the beginning of training. Warmup strategies are utilized to avoid training instability when using a larger learning rate, especially for those large or dynamic batch size settings. We propose incremental PID learning rates with a gradual warmup, named PID-Warmup, and it mainly comes from PID-Base. Fig. 3 shows the process of PID-Warmup updating learning rates. During the warmup phase, we let the learning rates updated by PID-Base be symmetric about the initial learning rate. The learning rates gradually increase in a non-linear way. PID-Warmup switches back to PID-Base after the warmup phase. The learning rates increase rapidly during the process of switching. However, our experiments show that the impact of this phenomenon is limited. Algorithm 2 is the pseudo-code of PID-Warmup. The parameter offset can change the symmetry axis during the warmup phase. We can increase the symmetry axis by setting offset > 1 and reduce it by setting $0 <$ offset < 1.

D. Roles and Settings of K_P and K_D

Although some control theories can be used for PID tuning, they are unsuitable for deep learning. For example, the Ziegler–Nichols rule [42] is a classical PID tuning method. The controlled object is usually assumed to be a first-order plus time delay model, but more research work is needed to verify whether deep learning follows the assumption. The rule also introduces some parameters, including the steady-state

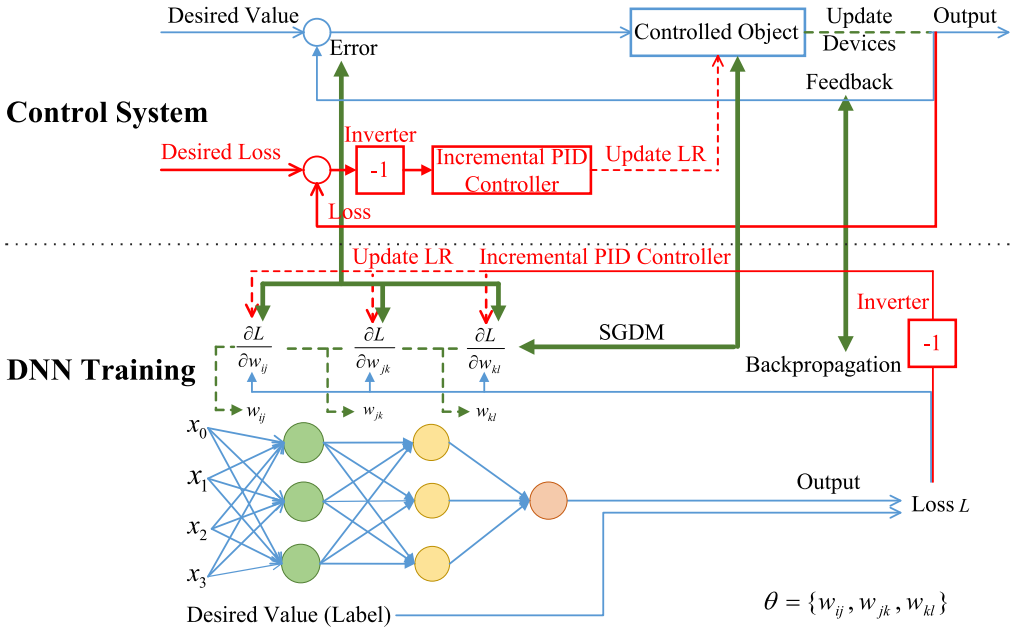

Fig. 2. Connection between feedback control system and DNN training [27]. The red part is incremental PID controller updates learning rates. "LR" denotes the learning rate.

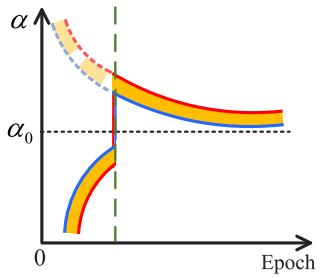

Fig. 3. Learning rates are updated by PID-Warmup.

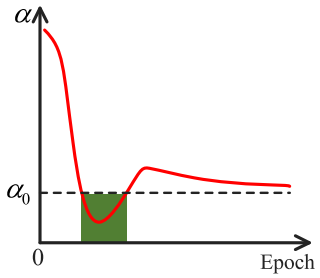

Fig. 4. In the green part, a large K_D causes an overshoot problem.

gain, time axis, maximum response rate, and time constant. Therefore, manually tuning K_P and K_D is a simple and effective method considering that there are various networks and datasets in the deep learning community.

We first discuss the role of K_P and K_D. In (15), K_I is set to 0 because it is difficult to estimate the training loss value before training a model. K_I also leads to an unstable learning rate increment $\Delta\alpha_t$ if K_I is unreasonable. For the $K_P \text{loss}_t$ term in (21a), K_P can quickly respond to training loss and update the current learning rate. It is the main source of accelerating network training and improving regularization. K_P can make networks perform better when the initial learning rate is too small. However, a large K_P causes the learning rate to be too large, which may cause SGD to fail to converge to local optima. The $K_D(\text{loss}_t - \text{loss}_{t-1})$ term in (21a) is susceptible to high-frequency noise from the training loss. In Algorithm 1, the step parameter can reduce the effect of high-frequency noise. The $K_D \text{loss}_{t-1}$ term in (21b) further indicates that a large K_D can cause $\alpha_t < \alpha_0$ easily and the updated learning rates are instability in a short time. As shown in the green part of Fig. 4, this phenomenon is called overshoot. A smaller K_D can alleviate the phenomenon. Therefore, K_D can improve the dynamic performance of learning rates and avoid the overshoot problem caused by unreasonable settings.

We further discuss strategies for setting K_P and K_D. Their relationship is defined as

$$K_D = \frac{K_P}{\varepsilon} \tag{30}$$

where ε is the scaling factor and is set to 5, 10, 20, and 50 to limit the harmful effect of the $K_D \text{loss}_{t-1}$ term in (21b). The main function of ε is to prevent a too large K_D from causing overshoot. Next, we provide a simple method to estimate K_P. PID-Base first decays from a large learning rate α_1. Equation (30) ensures that K_D is much smaller than K_P, so we ignore the influence of K_D such that

$$\alpha_1 \approx K_P \text{loss}_1 + \alpha_0 > \alpha_0. \tag{31}$$

Suppose the training loss interval is (0, 5], loss_1 is 5, and the initial learning rate is 0.1. The training loss interval is based on empirical observations for image datasets, and the initial learning rate is derived from [1], [3], [13], [14], [15]. If the K_P interval is [0.001, 0.01], then the α_1 interval

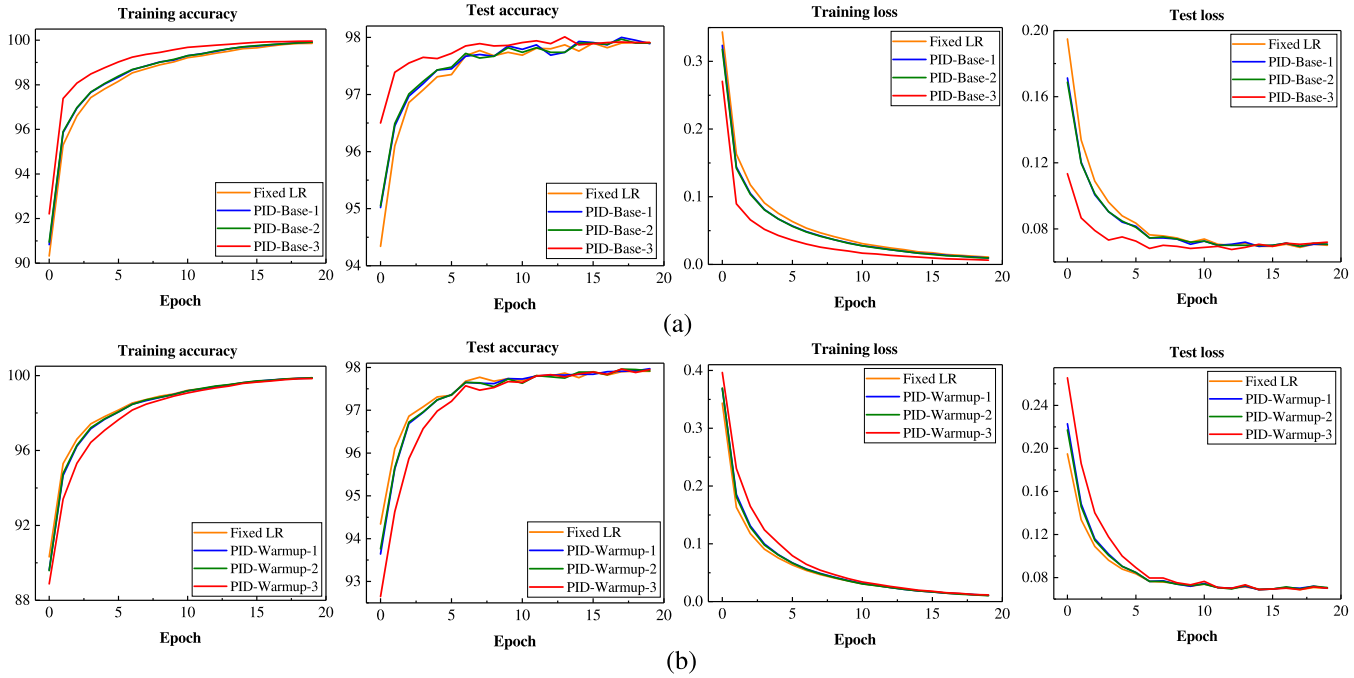

Fig. 5. (a) Trained by PID-Base. (b) Trained by PID-Warmup.

[0.105, 0.15] is consistent with parameter tuning experience. In Section IV-F, the parameter sensitivity experiments show that the classification results are not sensitive to K_D, and K_P has a wider value interval. Overall, tuning K_P and K_D have fine generality, physical significance, and interpretability.

IV. Experiments

We conduct experiments to demonstrate the effectiveness of PID-Base and PID-Warmup. First, an MLP is trained to explain the role of K_P and K_D on MNIST. Next, experimental results on CIFAR and Tiny-ImageNet-200 are reported. Finally, the ablation study and parameter sensitivity are discussed.

A. Datasets

MNIST [43] contains a training set of 60 000 examples and a test set of 10 000 examples of handwritten digits from 0 to 9. Each example is a 28 × 28 grayscale image.

CIFAR [38] consists of colored natural scene objects with resolution 32 × 32. CIFAR-10 and CIFAR-100 contain 10 classes and 100 classes, respectively. Both datasets are split into 50 000 training and 10 000 test images. For data augmentation, we use horizontal flips and random crops on the original image padded by 4 pixels on each side.

Tiny-ImageNet-200 [44] contains 100 000 training images and 10 000 testing images with 64 × 64 pixels. The dataset has 200 classes and each class has 500 training examples, 50 validation examples, and 50 test examples. We convert the resolution from 64 × 64 to 32 × 32 and use the same data augmentation as CIFAR.

TABLE I
Settings of K_P and K_D on MNIST

Group	K_P	K_D
Fixed LR	–	–
PID-Base-1	0.005	0.0005
PID-Base-2	0.005	0.005
PID-Base-3	0.05	0.005
PID-Warmup-1	0.005	0.0005
PID-Warmup-2	0.005	0.005
PID-Warmup-3	0.01	0.005

B. Results on MNIST

An MLP is trained to explain the influence of K_P and K_D. The MLP is with ReLU, 128 hidden neurons, and followed by the ten-way softmax output. We fix the random number seed and use SGD with a momentum of 0.9. The batch size is 64 and the fixed learning rate (Fixed LR) is 0.005. The K_P and K_D for different groups are listed in Table I.

In Fig. 5(a), the training loss is large at the beginning stage. PID-Base increases learning rates to accelerate network training and skip some local minima. The learning rate increase is mainly determined by K_P which can quickly respond to training losses. In Fig. 6(a), the learning rates decrease rapidly after the first update and then decay in a slow and small oscillating manner, and they finally converge to around the initial learning rate. This rapid decay benefits training convergence and avoids network instability caused by the random initialization of weight parameters. In both training and test, PID-Base converges faster than the Fixed LR group and achieves lower loss and higher accuracy. The learning rates of PID-Base-2 decrease sharply below the initial learning rate in the second update because K_D is set too large, which is caused by the overshoot problem. The partial enlargement

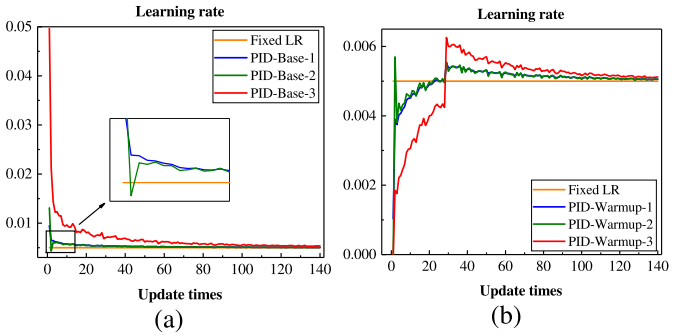

Fig. 6. MLP is trained by PID-Base and PID-Warmup on MNIST. (a) PID-Base. (b) PID-Warmup.

TABLE II
SETTINGS OF K_P AND K_D ON CIFAR

Model	Method	CIFAR-10		CIFAR-100	
		K_P	K_D	K_P	K_D
VGG-16-BN	PID-Base	0.005	0.0005	0.005	0.0005
	PID-Warmup	0.0005	0.00001	0.0005	0.00001
ResNet-18	PID-Base	0.005	0.001	0.005	0.001
	PID-Warmup	0.005	0.001	0.005	0.001
DenseNet-121	PID-Base	0.01	0.0005	0.01	0.001
	PID-Warmup	0.005	0.0005	0.005	0.0001

TABLE III
AVERAGE ACCURACY ON CIFAR

Model	Method	Accuracy (mean ± std %)	
		CIFAR-10	CIFAR-100
VGG-16-BN	MSLR	92.65 ± 0.11	70.73 ± 0.26
	CLR	91.58 ± 0.17	67.74 ± 0.28
	SGDR	92.53 ± 0.18	70.15 ± 0.12
	PID-Base	92.82 ± 0.24	**71.66 ± 0.17**
	PID-Warmup	**92.85 ± 0.04**	71.02 ± 0.25
ResNet-18	MSLR	94.83 ± 0.15	75.28 ± 0.13
	CLR	94.46 ± 0.07	74.50 ± 0.18
	SGDR	94.93 ± 0.14	75.39 ± 0.08
	PID-Base	95.00 ± 0.11	75.48 ± 0.11
	PID-Warmup	**95.04 ± 0.06**	**75.49 ± 0.08**
DenseNet-121	MSLR	94.56 ± 0.12	75.16 ± 0.15
	CLR	94.09 ± 0.10	74.41 ± 0.21
	SGDR	94.49 ± 0.10	74.92 ± 0.29
	PID-Base	**94.71 ± 0.07**	75.55 ± 0.13
	PID-Warmup	94.70 ± 0.11	**75.68 ± 0.12**

TABLE IV
CLASSIFICATION RESULTS OF VGG-16-BN WITH AN INITIAL LEARNING RATE OF 0.001 ON CIFAR

Model	Method	Accuracy (mean ± std %)	
		CIFAR-10	CIFAR-100
VGG-16-BN	MSLR	87.76 ± 0.20	61.14 ± 0.32
	PID-Base	**90.30 ± 0.10**	**64.90 ± 0.34**

of Fig. 6(a) shows the phenomenon. PID-Base-1 reduces K_D to solve the problem, which smoothes the updated learning rates and improves dynamic performance. Since MNIST is small-scale and the MLP is very simple, the benefits of dynamic performance are not demonstrated, and the training and test curves of PID-Base-1 and PID-Base-2 almost overlap.

During the warmup phase, the learning rates of PID-Warmup are obtained by PID-Base symmetric about the initial learning rate. The larger K_P is, the lower learning rates are obtained. The learning rates suddenly increase when PID-Warmup switches to PID-Base. The learning rate curves are shown in Fig. 6(b). The role of K_D is still the same. In Fig. 5(b), PID-Warmup does not show any advantages because the MNIST experiments are not suitable for warmup strategies.

C. Experiments on CIFAR

In this section, we illustrate the effectiveness of the proposed methods on CIFAR. All models are optimized by SGD with a momentum of 0.9 and the batch size is 128. VGG-16-BN is trained for 150 epochs, the initial learning rate of 0.01 is lowered by ten times at epochs 70 and 120, and weight decay is 0.0005. ResNet-18 and DenseNet-121 (growth rate $k = 12$) are trained for 300 epochs, the initial learning rate 0.1 is divided by ten times at epochs 150 and 230, and the weight decay is 0.0001. We also perform the same decay operations on the incremental PID learning rates to further improve convergence. For CLR, we set the *triangular* mode, $\alpha_{\min} = 0.001$, $\alpha_{\max} = 0.1$, and stepsize $= 2000$. In the VGG experiments, $\alpha_{\max}$ is set to 0.01 to be consistent with the initial learning rate. For SGDR, we set $T_0 = 50$ and $T_{\text{mult}} = 1$. We apply 5-epoch warmup to PID-Warmup. K_P and K_D are listed in Table II. The results of five runs are summarized in Table III.

In Table III, the incremental PID learning rates achieve the highest accuracy. Fig. 7 shows that CLR and SGDR have faster training acceleration but the final results are lower than our methods. The learning rate curves are shown in Fig. 8. PID-Base decays learning rates quickly in the initial stage and then updates learning rates in a small and slow oscillation. PID-warmup slowly increases learning rates during the warmup phase and then switches to PID-Base.

Furthermore, the initial learning rate is reduced to 0.001 to demonstrate the superiority of our methods. Since the warmup strategy is used to avoid training instability in the initial stage when the initial learning rate is set large [35], we only compare PID-Base and MSLR to illustrate the importance of the initial learning rate. As shown in Table IV, PID-Base is 2.54% and 3.76% higher than MSLR on CIFAR-10 and CIFAR-100, respectively. VGG is a plain network that easily causes vanishing or exploding gradients [1]. The learning rate of 0.001 is unsuitable for VGG, while PID-Base overcomes the poor learning rate by decaying from a large learning rate.

D. Classification on Tiny-ImageNet-200

Finally, ResNet-34, DenseNet-121 ($k = 32$), MobileNetV2 [45], and RegNetX-200MF [14] are trained on Tiny-ImageNet-200. All networks are optimized by SGD with a momentum of 0.9 and a weight decay of 0.0001. They are trained for 150 epochs with a batch size of 128, and the initial learning rate of 0.1 is lowered by ten times at epochs 70 and 110. Other settings are the same as CIFAR. We also use the *exp_range* mode of CLR, and the scaling factor is 0.99998. The symbol * denotes *exp_range*. We apply a 5-epoch warmup

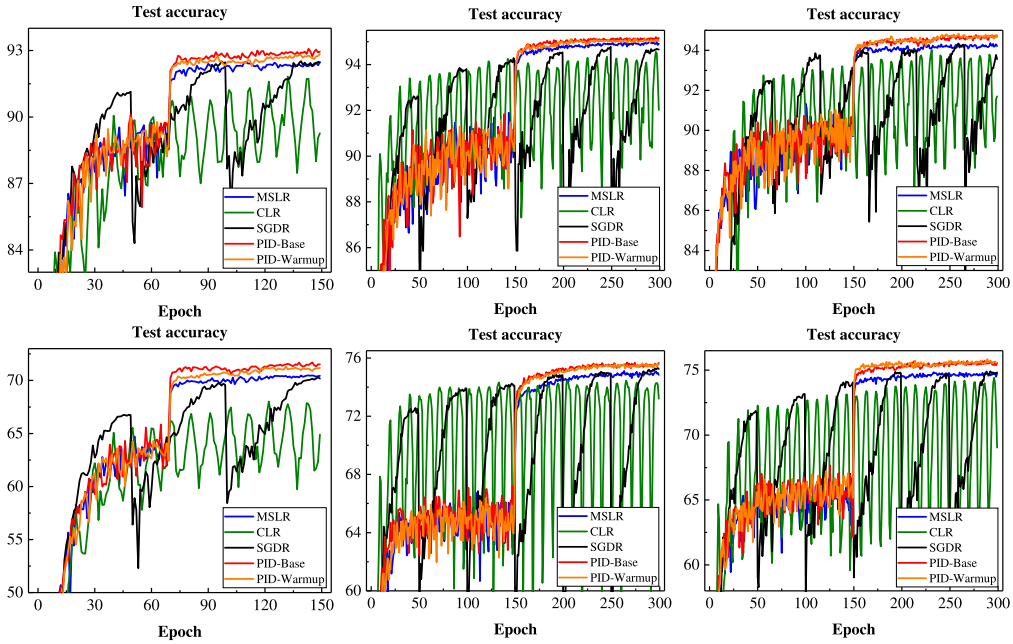

Fig. 7. Test accuracy on CIFAR. Top row: VGG-16-BN, ResNet-18, and DenseNet-121 on CIFAR-10. Bottom row: VGG-16-BN, ResNet-18, and DenseNet-121 on CIFAR-100.

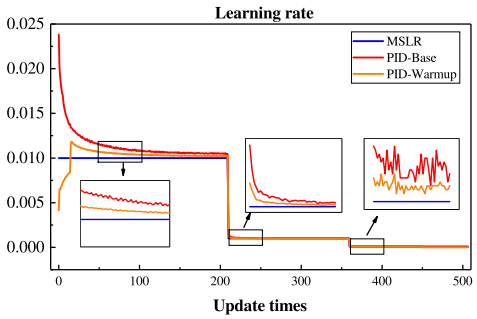

Fig. 8. Learning rate curves of VGG-16-BN on CIFAR-10.

TABLE V
SETTINGS OF K_P AND K_D ON TINY-IMAGENET-200

Model	Method	K_P	K_D
ResNet-34	PID-Base	0.01	0.0005
	PID-Warmup	0.005	0.0001
DenseNet-121	PID-Base	0.01	0.0005
	PID-Warmup	0.005	0.0001
MobileNetV2	PID-Base	0.001	0.0001
	PID-Warmup	0.001	0.0001
RegNetX-200MF	PID-Base	0.01	0.0005
	PID-Warmup	0.005	0.0001

TABLE VI
AVERAGE ACCURACY ON TINY-IMAGENET-200

Model	Method	Accuracy (mean ± std %)
ResNet-34	MSLR	56.43 ± 0.52
	CLR	54.84 ± 0.27
	CLR*	56.23 ± 0.33
	SGDR	55.85 ± 0.68
	PID-Base	**57.63 ± 0.12**
	PID-Warmup	57.32 ± 0.45
DenseNet-121	MSLR	59.74 ± 0.23
	CLR	58.73 ± 0.14
	CLR*	58.10 ± 0.55
	SGDR	59.55 ± 0.36
	PID-Base	60.39 ± 0.20
	PID-Warmup	**60.63 ± 0.19**
MobileNetV2	MSLR	57.30 ± 0.23
	CLR	56.25 ± 0.23
	CLR*	55.16 ± 0.11
	SGDR	56.33 ± 0.07
	PID-Base	57.58 ± 0.43
	PID-Warmup	**57.64 ± 0.17**
RegNetX-200MF	MSLR	57.19 ± 0.13
	CLR	55.81 ± 0.39
	CLR*	55.11 ± 0.47
	SGDR	56.23 ± 0.36
	PID-Base	**57.92 ± 0.20**
	PID-Warmup	57.74 ± 0.60

to PID-Warmup. Table V shows the K_P and K_D settings. The results of five runs are reported in Table VI.

Our methods achieve the highest accuracy in Table VI and Fig. 9. Although CLR and SGDR can significantly accelerate training, they require more time to fine-tune hyperparameters. Their experimental results rely on subjective experience due to the hyperparameters are not directly related to model training. However, our methods adjust learning rates through a feedback loop. As shown in Fig. 10, the control effect of the incremental PID controller for learning rates is consistent with CIFAR.

E. Ablation Study

We perform ablation experiments to demonstrate the effectiveness of K_P and K_D on Tiny-ImageNet-200. According to (21a), incremental P learning rates P_{α_t} and incremental D learning rates D_{α_t} are written as follows:

$$\begin{aligned} P_{\alpha_t} &= \alpha_0 + K_P \text{loss}_t \\ D_{\alpha_t} &= \alpha_0 + K_D(\text{loss}_t - \text{loss}_{t-1}). \end{aligned} \tag{32}$$

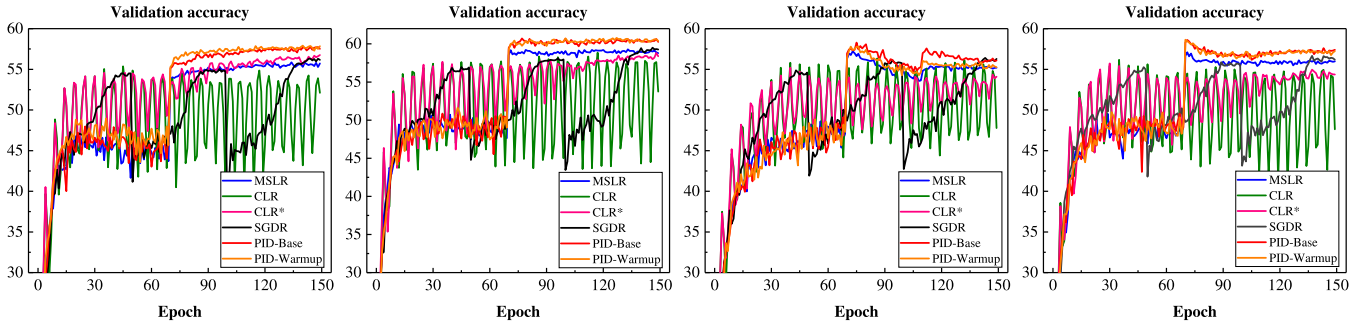

Fig. 9. From left to right: ResNet-34, DenseNet-121, MobileNetV2, and RegNetX-200MF on Tiny-ImageNet-200. The symbol * denotes *exp_range*.

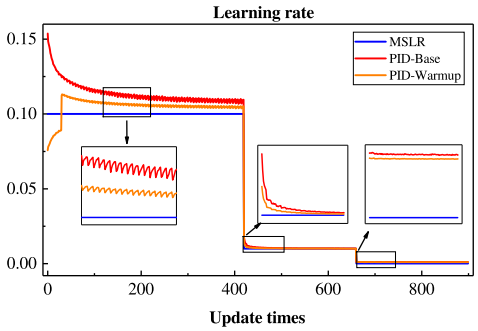

Fig. 10. Learning rate curves of DenseNet-121 on Tiny-ImageNet-200.

The classification results are reported in Table VII. K_P is more significant for performance gain than K_D. The average accuracy obtained by a single K_D is competitive with or slightly higher than the baseline method (MSLR), while a single K_P is significantly higher than MSLR. K_P can quickly respond to training losses, and K_D can respond to the dynamic change of training losses. Since the learning rate updated by P_{α_t} is higher than D_{α_t}, the accuracy of P_{α_t} is higher than D_{α_t}. P_{α_t} decays from a large learning rate, making networks skip some spurious local minima. Finally, the complete incremental PID learning rates can further improve the classification results.

F. Parameter Sensitivity

We test the sensitivity of K_P and K_D on Tiny-ImageNet-200. The average accuracy results under different K_P and K_D are shown in Fig. 11. Like the ablation study, K_P has a more significant impact on classification results than K_D. A larger K_P can significantly improve accuracy. The group average of K_P for PID-Base increases from 57.31% to 57.95% when K_P increases from 0.001 to 0.015. However, the group average of K_D remains around 57.70%. K_P can provide a larger performance gain in a wider value interval, and the experimental results are not sensitive to K_D. Similar experimental phenomena are also observed on PID-Warmup. Besides, the incremental PID learning rates outperform CLR, SGDR, and MSLR (only two results are lower than MSLR). Therefore, our methods are efficient and the hyperparameter tuning is intuitive.

TABLE VII
ABLATION STUDY OF THE PROPOSED METHODS ON TINY-IMAGENET-200

Model	Method	K_P	K_D	Accuracy (mean ± std %)
ResNet-34	MSLR	–	–	56.43 ± 0.52
	PID-Base	✓	✗	57.38 ± 0.10
		✗	✓	56.43 ± 0.07
		✓	✓	57.63 ± 0.12
	PID-Warmup	✓	✗	57.21 ± 0.17
		✗	✓	56.77 ± 0.26
		✓	✓	57.32 ± 0.45
DenseNet-121	MSLR	–	–	59.74 ± 0.23
	PID-Base	✓	✗	60.23 ± 0.38
		✗	✓	60.10 ± 0.34
		✓	✓	60.39 ± 0.20
	PID-Warmup	✓	✗	60.27 ± 0.09
		✗	✓	59.91 ± 0.46
		✓	✓	60.63 ± 0.19
MobileNetV2	MSLR	–	–	57.30 ± 0.23
	PID-Base	✓	✗	57.46 ± 0.25
		✗	✓	57.39 ± 0.37
		✓	✓	57.58 ± 0.43
	PID-Warmup	✓	✗	57.44 ± 0.10
		✗	✓	57.24 ± 0.16
		✓	✓	57.64 ± 0.17
RegNetX-200MF	MSLR	–	–	57.19 ± 0.13
	PID-Base	✓	✗	57.57 ± 0.40
		✗	✓	57.16 ± 0.20
		✓	✓	57.92 ± 0.20
	PID-Warmup	✓	✗	57.65 ± 0.15
		✗	✓	57.46 ± 0.37
		✓	✓	57.74 ± 0.60

TABLE VIII
RESULTS OF THE PID OPTIMIZER

Dataset	Model	Accuracy (%)
CIFAR-10	VGG-16-BN	NaN
	ResNet-18	94.93 ± 0.13 (−0.07, −0.11)
	DenseNet-121	94.18 ± 0.07 (−0.53, −0.52)
CIFAR-100	VGG-16-BN	72.70 ± 0.13 (+1.04, +1.68)
	ResNet-18	75.78 ± 0.15 (+0.30, +0.29)
	DenseNet-121	75.14 ± 0.19 (−0.41, −0.54)
Tiny-ImageNet-200	ResNet-34	57.80 ± 0.20 (+0.17, +0.48)
	DenseNet-121	58.50 ± 0.38 (−1.89, −2.13)
	MobileNetV2	52.94 ± 0.70 (−4.64, −4.70)
	RegNetX-200MF	56.54 ± 0.12 (−1.38, −1.20)

[1] The numbers in parentheses represent the difference in accuracy between PID optimizer and SGDM with PID-Base and PID-Warmup, respectively.
[2] Underline in parentheses indicates that PID optimizer has no statistical significance (t-test, $p > 0.05$) with the corresponding method.

V. DISCUSSION

The learning rates updated by MSLR, CLR, and SGDR do not reflect the training characteristics of different networks and

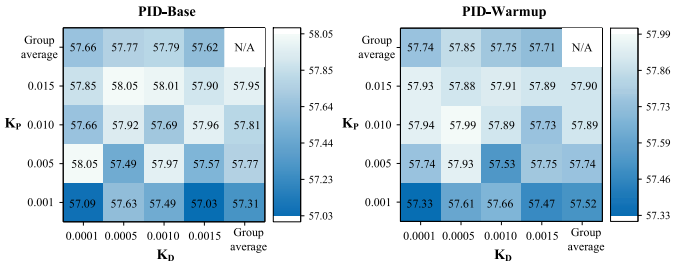

Fig. 11. Parameter sensitivity test for PID-Base and PID-Warmup.

datasets. Specifically, they calculate learning rates according to (4), (5), and (6). The learning rates predetermined before training cannot be adapted to various training processes, resulting in hyperparameter tuning heavily dependent on subjective experience. However, incremental PID learning rates are adaptive schedulers because our methods use feedback control to establish the relationship between training losses and the current learning rate. K_P can respond to errors and increase learning rates, and K_D can promote the dynamics of learning rates. Incremental PID learning rates automatically adapt to training patterns and improve accuracy. Besides, K_P and K_D have vivid physical meaning and intuitive interpretability, which reduces the difficulty of hyperparameter tuning.

Since SGDM can be regarded as a proportional-integral (PI) controller, a derivative (D) term of gradient $\nabla_\theta F(\theta)$ is introduced to develop a PID optimizer [27]

$$\text{PID optimizer} = \text{SGDM} + K_D[\nabla_\theta F(\theta_t) - \nabla_\theta F(\theta_{t-1})]. \quad (33)$$

Note that the PID optimizer is not strictly comparable to our proposed incremental PID learning rates because the PID optimizer is a new variant of SGD but our methods are learning rate schedulers. The gradient $\nabla_\theta F(\theta)$ is an incarnation of the error for the PID optimizer, while our methods treat training loss as an error. The computational complexity of the gradient-based error is higher than the training loss error. Owing to the high-dimensional characteristics of gradients, the integral and derivative of the PID optimizer do not have intuitive physical meaning. The DNN optimization is simplified to a second-order underdamped system with a one-step response as the parameter update [27]. However, our methods do not have these restrictive assumptions.

Furthermore, we test the performance of the PID optimizer. Its integral and derivative parameters are set to 5 and 10, respectively. The experimental results are reported in Table VIII. In most cases, SGDM with incremental PID learning rates outperforms the PID optimizer, and the t-test shows that our methods are significant (t-test, $p < 0.05$). In addition, the PID optimizer shows exploding gradients on VGG-16-BN training CIFAR-10. One possible reason is that the D term is disturbed by the high-frequency noise of gradients. In the previous research [27], the PID optimizer is better than SGDM. On the contrary, our experimental results show that incremental PID learning rates can improve the performance of SGDM.

VI. Conclusion

In this study, we propose an incremental PID controller-based learning rate scheduler for SGD, named incremental PID learning rates, including PID-Base and PID-Warmup. The incremental PID controller establishes the relationship between training losses and learning rates through feedback control. Compared with MSLR, CLR, and SGDR, K_P and K_D have excellent physical meaning and intuitive interpretability. Experimental results confirm that our methods can improve performance on CIFAR-10, CIFAR-100, and Tiny-ImageNet-200. These advantages reduce work in setting learning rates and make our methods practical tools for researchers who employ SGD to train deep networks.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[2] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5987–5995.
[3] J. Hu, L. Shen, and G. Sun, "Squeeze- and-excitation networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 7132–7141.
[4] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 5999–6009.
[5] J. Devlin, M.-W. Chang, K. Lee, and K. N. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1, 2019, pp. 4171–4186.
[6] L. Bottou, "On-line learning and stochastic approximations," in *On-Line Learning in Neural Networks*, no. 34, D. Saad, Ed. Cambridge, U.K.: Cambridge Univ. Press, 1999, pp. 9–42.
[7] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 2176–2184.
[8] Y. Nesterov, "A method for solving the convex programming problem with convergence rate $O(1/k^2)$," *Dokl. Akad. Nauk SSSR*, vol. 269, no. 3, pp. 543–547, 1983.
[9] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, Feb. 2011.
[10] R. Sutton, "Two problems with back propagation and other steepest descent learning procedures for networks," in *Proc. 8th Annu. Conf. Cognit. Sci. Soc.*, 1986, pp. 823–831.
[11] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, 2014, pp. 1–15.
[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–14.
[13] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2261–2269.
[14] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollar, "Designing network design spaces," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, p. 10.
[15] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun, "RepVGG: Making VGG-style convnets great again," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 13728–13737.
[16] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proc. Brit. Mach. Vis. Conf.*, 2016, pp. 1–15.
[17] C. Szegedy et al., "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.
[18] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of ADAM and beyond," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–23.
[19] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht, "The marginal value of adaptive gradient methods in machine learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 4149–4159.
[20] P. Goyal et al., "Accurate, large minibatch SGD: Training imagenet in 1 hour," 2017, *arXiv:1706.02677*.

[21] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–16.
[22] L. N. Smith, "Cyclical learning rates for training neural networks," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2017, pp. 464–472.
[23] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci. USA*, vol. 79, no. 8, pp. 2554–2558, 1982.
[24] K. Xu, C. Li, J. Zhu, and B. Zhang, "Understanding and stabilizing GANs' training dynamics using control theory," in *Proc. Int. Conf. Mach. Learn.*, vol. 1, 2020, pp. 10566–10575.
[25] J. Li, Y. Yuan, T. Ruan, J. Chen, and X. Luo, "A proportional-integral-derivative-incorporated stochastic gradient descent-based latent factor analysis model," *Neurocomputing*, vol. 427, pp. 29–39, Feb. 2021.
[26] W. An, H. Wang, Q. Sun, J. Xu, Q. Dai, and L. Zhang, "A PID controller approach for stochastic optimization of deep networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 8522–8531.
[27] H. Wang, Y. Luo, W. An, Q. Sun, J. Xu, and L. Zhang, "PID controller-based stochastic optimization acceleration for deep neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 12, pp. 5079–5091, Dec. 2020.
[28] L. Shi, Y. Zhang, W. Wang, J. Cheng, and H. Lu, "Rethinking the PID optimizer for stochastic optimization of deep networks," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2020, pp. 1–6.
[29] W. Tang, Y. Zhao, W. Xie, and W. Huang, "A novel adaptive PID optimizer of deep neural networks," in *Proc. Int. Conf. Neural Inf. Process.*, 2021, pp. 506–513.
[30] D. Wang, M. Ji, Y. Wang, H. Wang, and L. Fang, "SPI-optimizer: An integral-separated PI controller for stochastic optimization," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2019, pp. 2129–2133.
[31] Y. LeCun, I. Kanter, and S. A. Solla, "Second order properties of error surfaces: Learning time and generalization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 3, 1990, pp. 918–924.
[32] B. Kleinberg, Y. Li, and Y. Yuan, "An alternative view: When does SGD escape local minima?" in *Proc. 35th Int. Conf. Mach. Learn.*, vol. 80, 2018, pp. 2698–2707.
[33] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, "Snapshot ensembles: Train 1, get m for free," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–14.
[34] K. You, M. Long, J. Wang, and M. I. Jordan, "How does learning rate decay help modern neural networks?" 2019, *arXiv:1908.01878*.
[35] A. Gotmare, N. S. Keskar, C. Xiong, and R. Socher, "A closer look at deep learning heuristics: Learning rate restarts, warmup and distillation," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–15.
[36] K. H. Ang, G. Chong, and Y. Li, "PID control system analysis, design, and technology," *IEEE Trans. Control Syst. Technol.*, vol. 13, no. 4, pp. 559–576, Jul. 2005.
[37] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural Networks: Tricks of the Trade*. Berlin, Germany: Springer, 2012, pp. 437–478.
[38] A. Krizhevsky, "Learning multiple layers of features from tiny images," M.S. thesis, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, 2009.
[39] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Rev.*, vol. 60, no. 2, pp. 223–311, 2018.
[40] Y. Li, C. Wei, and T. Ma, "Towards explaining the regularization effect of initial large learning rate in training neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–12.
[41] S. J. Hanson and L. Y. Pratt, "Comparing biases for minimal network construction with back-propagation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 1, 1988, pp. 1–9.
[42] J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers," *Trans. ASME*, vol. 64, no. 11, pp. 759–765, 1942.
[43] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[44] Y. Le and X. Yang, "Tiny ImageNet visual recognition challenge," *CS 231N*, vol. 7, no. 7, p. 3, 2015.
[45] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

Zenghui Wang received the B.S. degree from Hefei University, Hefei, China, in 2019. He is currently pursuing the M.S. degree with the School of Electrical Engineering and Automation, Anhui University.

His current research interests include brain–computer interface, biosignal processing, and computer vision.

Jun Zhang received the B.S. degree from the Hefei University of Technology, Hefei, China, in 1995, the M.S. degree from the Institute of Intelligent Machine, Chinese Academy of Sciences, Hefei, in 2004, and the Ph.D. degree from the University of Science and Technology of China, Hefei, in 2007.

He was a Post-Doctoral Fellow with the University of Louisville, Louisville, KY, USA, from 2009 to 2011. He is currently serving as a Full Professor with the School of Artificial Intelligence, Anhui University, Hefei. He has published more than 100 publications. His current research interests include bioinformatics, cheminformatics, machine learning, and computer vision.