RETHINKING THE PID OPTIMIZER FOR STOCHASTIC OPTIMIZATION OF DEEP NETWORKS

Lei Shi[1,2]*, Yifan Zhang*[1,2]*, Wanguo Wang*[3]*, Jian Cheng*[1,2,4] *and Hanqing Lu*[1,2]

[1]NLPR & AIRIA, Institute of Automation, Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China
[3]State Grid Intelligence Technology Co., Ltd.
[4]CAS Center for Excellence in Brain Science and Intelligence Technology
{lei.shi, yfzhang, jcheng, luhq}@nlpr.ia.ac.cn; wangwanguo03@163.com

ABSTRACT

Stochastic gradient descent with momentum (SGD-Momentum) always causes the overshoot problem due to the integral action of the momentum term. Recently, an ID optimizer is proposed to solve the overshoot problem with the help of derivative information. However, the derivative term suffers from the interference of the high-frequency noise, especially for the stochastic gradient descent method that uses minibatch data in each update step. In this work, we propose a complete PID optimizer, which weakens the effect of the D term and adds a P term to more stably alleviate the overshoot problem. To further reduce the interference of the high-frequency noise, two effective and efficient methods are proposed to stabilize the training process. Extensive experiments on three widely used benchmark datasets with different scales, i.e., MNIST, Cifar10 and TinyImageNet, demonstrate the superiority of our proposed PID optimizer on various popular deep neural networks.

Index Terms— SGD, PID, Optimizer, Deep Networks

1. INTRODUCTION

Recently, deep neural networks have shown the superiority in many areas [1, 2, 3] However, how to train deep neural networks with a large number of parameters is still a difficult problem, which requires a well-designed optimization method. Currently, the most widely used method is the SGD-Momentum optimizer. Different from the SGD optimizer that updates the parameter along the opposite direction of the gradient at current step, SGD-Momentum determines the update direction according to the moving average decay of the gradients in all of the historical steps. However, the historical information slows the response to the changes in the update direction, thus causes the parameter to continue to update in the wrong direction although the gradients in the current step already point in the correct direction. This problem is called the overshoot problem in the field of industrial control.

Recently, An et al. [4] proposed an integral-derivative (ID) optimizer to solve the problem. The ID optimizer utilizes the derivative (D) of the gradient in the current step to guide the update. The derivative can predict the variation tendency of the update direction, thus can respond in advance and reduce the effect of the overshoot problem. Nonetheless, the derivative is easily interfered by the high-frequency noise, which is a common phenomenon in stochastic gradient descent-based methods due to their use of minibatch data. This interference is especially severe in the initial stage of the training, where the direction of the gradient changes frequently thus its derivative also fluctuates sharply. The interference affects the stability of the training process, which slows the training speed and affects the performance of the final models.

In this work, we propose to weaken the effect of the derivative term and additionally add a proportional term (P) to form a complete PID optimizer. The P term is proportional to the value of the gradient at the current step, which can provide more flexibility for the optimizer to adjust the proportion of the current information and the past information. Increasing the proportion of the P term can also alleviate the overshoot problem by focusing more on the gradients of the current step. Compared with the derivative term, although it cannot predict the future trend, the P is more stable without the concern about the interference of the high-frequency noise. By adjusting the P, I and D to a suitable ratio, the complete PID optimizer always achieves better performance compared with traditional methods.

In addition, to further reduce the interference of the high-frequency noise for the D term, we propose two simple and effective improvements. One is called the incomplete differential, which adds a low-pass filter after the D term to reduce the fluctuation of the differential value. The other is called the adaptive differential output clamping, which adaptively restricts the amplitude of the derivative value based on the current value of the gradient. Both methods have been proven to be very effective for the training process in the ablation study experiments.

978-1-7281-1331-9/20/$31.00

To verify the superiority of our methods, extensive experiments are conducted on three widely used datasets with different scales: MNIST, Cifar10 and TinyImageNet. We compare the different optimizers with different deep neural networks, where our PID method achieves better performance in most of the cases.

The main contributions of our work lies in three aspects: (1) A complete PID optimizer that can take advantage of the proportional term, integral term and derivative term at the same time to help models converge to a better performance. (2) Two improvements, i.e., the incomplete differential and the adaptive differential output clamping, are designed specially to reduce the interference of the high-frequency noise for the D term. (3) With extensive experiments performed on three widely used datasets, our PID optimizer shows better performance in different deep neural networks than conventional methods.

2. RELATED WORK

2.1. Gradient descent optimization algorithms:

The mainstream optimizers for deep neural networks can be divided into two categories: (1) nonadaptive optimizers such as SGD [5], SGD-Momentum [6] and Nesterov's SGD-Momentum [7], which perform the same updating strategies for all of the parameters, and (2) adaptive optimizers that adaptively tune an individual learning rate for each parameter of the model based on their historical performance, such as AdaGrad [8], AdaDelta [9] and Adam [10]. Although the adaptive methods are very attractive, they usually do not perform as well as expected as illustrated in [11]. It is because the adaptive methods usually have poor generalization capabilities and may easily converge to the local optimum [12]. In this work, we only consider the nonadaptive methods.

2.2. PID controller:

Proportional-integral-derivative (PID) control offers the simplest and yet most efficient solution to many real-world control problems. It has been widely researched and has produced many variations since 1910 [13, 14, 15, 16]. It is formulated as

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t) \qquad (1)$$

where e denotes the difference between the target value and the current value, i.e., the error. u is the control variable and is adjusted according to the error. The complete PID controller contains three terms: a proportional (P) term, an integral (I) term and a derivative (D) term. K_p, K_i and K_d denote the three coefficients for the P, I and D terms, respectively. Note that to implement the PID controller, Eq. 1 is always transformed into a discrete manner:

$$u(t) = K_p e(t) + K_i \sum_{\tau=0}^{t} e(\tau) + K_d(e(t) - e(t-1)) \qquad (2)$$

where the integration is estimated by the summation and the derivative is estimated by the difference between the errors in the current step and the previous step.

3. REVISIT THE NONADAPTIVE OPTIMIZERS

3.1. SGD optimizer

SGD optimizer is the most basic algorithm for the optimization of deep neural networks, which updates the parameters in the opposite direction of the gradients of the objective function w.r.t the parameters. Its update rule is:

$$\theta_{t+1} = \theta_t - K_p g_t \qquad (3)$$

where t denotes the current update step. θ denotes the parameters of the model that are going to be updated. g denotes the backpropagated gradient of θ. K_p denotes the learning rate of SGD, which determines the size of the update in the current step. Compared with the Eq. 2, it is similar to the update rule of the the P controller when setting the K_i and K_d to 0. The error e in Eq. 2 corresponds to the gradient g in Eq. 3. The control output of PID corresponds to the update of the parameters in SGD. Therefore, the SGD optimizer can be viewed as the P optimizer.

3.2. SGD-Momentum optimizer

SGD-Momentum optimizer adds a momentum term to the original SGD optimizer, which accumulates the previous gradients and makes the decision based on the moving average decay of the historical gradients. The update rule of the SGD-Momentum optimizer can be formulated as follows:

$$\begin{aligned} v_{t+1} &= \alpha v_t + g_t \\ \theta_{t+1} &= \theta_t - K_i v_{t+1} \end{aligned} \qquad (4)$$

where v is the momentum item, which is the moving average decay over past gradients. α is the rate of moving average decay which focuses on the recent gradients. K_i denotes the learning rate of the SGD-Momentum optimizer. Rewrite the formulation by removing the item v:

$$\theta_{t+1} = \theta_t - K_i \sum_{i=0}^{t} \alpha^{t-i} g_i \qquad (5)$$

Actually, it is similar to the I controller when setting the K_p and K_d in Eq. 2 to 0. The only difference is the factor α of average decay used in momentum term. It is important because the SGD-Momentum uses only part of the data to update the

parameters in each step, thus the gradients are stochastic and the history gradients become increasingly useless for updating parameters in current step. Overall, the SGD-Momentum optimizer can be viewed as an I optimizer.

3.3. ID optimizer

An et al.[4] added an additional D term based on the SGD-Momentum optimizer, whose update rule is:

$$\begin{aligned} v_{t+1} &= \alpha v_t + g_t \\ u_{t+1} &= \alpha u_t + (1-\alpha)(g_t - g_{t-1}) \\ \theta_{t+1} &= \theta_t - K_i v_{t+1} - K_d u_{t+1} \end{aligned} \tag{6}$$

which can also be transformed into a single formula:

$$\theta_{t+1} = \theta_t - K_i \sum_{i=0}^{t} \alpha^{t-i} g_i - K_d \sum_{i=1}^{t} \alpha^{t-i}(1-\alpha)(g_i - g_{i-1}) \tag{7}$$

Note that An et al. [4] separate the g_t from the I term and claim that it is a PID optimizer. However, the complete PID controller has three freedoms on P, I and D. According to Eq. 7, there are only two freedoms matched with K_i and K_d and the proportion of P to I is not adjustable. Therefore, we argue that it is more similar to an ID optimizer rather than a PID optimizer. Moreover, the rates of moving average decay for both I and D in [4] are same, which is not reasonable.

4. OUR METHODS

4.1. PID optimizer

Instead of using a derivative term that is sensitive to the high-frequency noise, we propose a PI optimizer, which is a combination of a proportional term and an integral term:

$$\begin{aligned} v_{t+1} &= \alpha v_t + g_t \\ \theta_{t+1} &= \theta_t - K_p g_t - K_i v_{t+1} \end{aligned} \tag{8}$$

The PI optimizer can also overcome the overshoot problem with the help of the P term by focusing more on the gradient in the current step. However, the PI optimizer is more stable than the ID optimizer since there is no need to calculate the derivative. Additionally, we can take advantage of all three terms and obtain the complete PID optimizer:

$$\begin{aligned} v_{t+1} &= \alpha v_t + g_t \\ \Delta g_t &= g_t - g_{t-1} \\ u_{t+1} &= \Delta g_t \\ \theta_{t+1} &= \theta_t - K_p g_t - K_i v_{t+1} - K_d u_{t+1} \end{aligned} \tag{9}$$

where K_p,K_i and K_d are the ratio coefficients of the P term, I term and D term, respectively. By removing the v_t and u_t, Eq. 9 can be transformed into Eq. 10.

$$\theta_{t+1} = \theta_t - K_p g_t - K_i \sum_{i=0}^{t} \alpha^{t-i} g_i - K_d(g_t - g_{t-1}) \tag{10}$$

From the PID controller perspective, the P term of the optimizer is used to adjust the importance of the current direction of the gradient in the final decision. The I term is used to accumulate the history information, which can accelerate the training process when the parameter is optimized and heading in one direction in several consecutive batches. Additionally, when the training process is nearing the end, the I term can reduce the static error by considering more batches. The D term can estimate the future trend based on the current rate of change, which can respond in advance and make the training converge faster.

4.2. Incomplete differential PID optimizer

Based on the experiments, we found that the value of the D term has severe fluctuations during the training process, which seriously affects the stability of training and the final performance. This result is because the stochastic gradient decent algorithm uses a minibatch of data in each optimization step, which cannot represent the overall dataset and will cause the high-frequency noise in the optimization process. Since the D term is calculated based on the derivative, it is very sensitive to the noise. Inspired by the incomplete differential PID controller that is widely used in the field of industrial control, we add a low-pass filter (LPF) behind the derivative term to filter the high-frequency signals. It is equipped with adding a moving average decay on the D term, which considers the history value on the current step and calculates a weighted average on these values. Eq. 11 formulates the incomplete differential PID optimizer, where β is the rate of the moving average decay for D term.

$$\begin{aligned} v_{t+1} &= \alpha v_t + g_t \\ \Delta g_t &= g_t - g_{t-1} \\ u_{t+1} &= \beta u_t + \Delta g_t \\ \theta_{t+1} &= \theta_t - K_p g_t - K_i v_{t+1} - K_d u_{t+1} \end{aligned} \tag{11}$$

4.3. Adaptive differential output clamping

Although adding the low-pass filter can weaken the fluctuation problem, we found that the value of D term can sometimes be very large, especially in the beginning of the training process. This result is because that in the initial stage of the training process, the model is unstable and the difference in the gradient direction in two batches can be very large. To reduce the disturbance of these abnormal values, we propose an adaptive clamping method shown as follows:

$$
\begin{aligned}
&v_{t+1} = \alpha v_t + g_t \\
&\Delta g_t = g_t - g_{t-1} \\
&T = \eta |g_t| \\
&\gamma = \begin{cases} 0 & |\Delta g_t| > T \\ 1 & otherwise \end{cases} \\
&w_{t+1} = \gamma(\Delta g_t) + (1-\gamma)\frac{\Delta g_t}{|\Delta g_t| + \epsilon} T \\
&u_{t+1} = \beta u_t + w_{t+1} \\
&\theta_{t+1} = \theta_t - K_p g_t - K_i v_{t+1} - K_d u_{t+1}
\end{aligned} \tag{12}
$$

where $|\Delta g_t|$ denotes the absolute value of the current change in the gradient. If it is higher than a threshold T ($T \geq 0$), the Δg_t will be clamped to T with the same sign, i.e., $\frac{\Delta g_t}{|\Delta g_t|+\epsilon} T$. ϵ is a small value that is used to avoid dividing by 0. Since the amplitude of gradient will change in the training process, we adapt the clamping threshold based on the gradient in current step. In detail, we set $T = \eta |g_t|$, where η is a scale factor.

5. EXPERIMENTS

5.1. Datasets

MNIST is a handwriten digit (0-9) dataset, which contains 60,000 training samples and 10,000 test samples. The images in MNIST are 28×28 pixels and have only one channel. **Cifar10** contains 60,000 32×32 color images for classification. There are 10 classes with 6,000 images per class. It is split into a training set (50,000 images) and a test set (10,000 images). **TinyImageNet** has 120,000 64×64 color images, which has 200 classes with 600 images per class. There are a tatal of 100,000 images in the training set, 10,000 images in the validation set and 10,000 images in the test set. It is more difficult compared with Cifar10 and MNIST due to the increased number of classes and the more challenging images.

5.2. Visualization of the optimization trajectories

The Beale function is a widely used test function (also known as artificial landscapes) to evaluate optimization algorithms. Its object function is shown as Eq. 13. Its global minimum is $\mathbf{x} = (\mathbf{3}, \mathbf{0.5})$.

$$
\begin{aligned}
f(\mathbf{x}) =& (1.5 - x_1 + x_1 x_2)^2 + (2.25 - x_1 + x_1 x_2^2)^2 \\
&+ (2.625 - x_1 + x_1 x_2^3)^2
\end{aligned} \tag{13}
$$

Here, we apply different optimizers to optimize the Beale function and visualize their optimization trajectories on the contours of the loss surface, including the P optimizer (SGD), the I optimizer (SGD-Momentum), the PI optimizer, the ID optimizer, the PD optimizer and the PID optimizer. The initial point is $\mathbf{x} = (\mathbf{2}, \mathbf{2})$, and the training process ends in 1000 steps. We simply set $K_p = K_i = K_d = 0.0001$ for all the experiments, which may not be the best choice but can demonstrate key phenomenon.

Fig. 1 shows the result of the visualization: (1) For SGD, the distance between its end point and the target point is the largest. It is actually a P optimizer, which can only see the error in the current step. It can not utilize the past experience as well as the prediction for the future trend. Therefore, it has the lowest performance compared with the other optimizers. (2) SGD-Momentum, which is actually an I optimizer, can utilize the information of the gradient history to accelerate the convergence and reduce the variance and bias. It shows a faster convergence and a shorter distance from the end point to the optimum point. However, SGD-Momentum has the problem of overshoot, which is cycled with red dotted lines. Inside the area of the red cycle, although the gradient has already changed direction, the optimization direction cannot immediately turn due to the large momentum value accumulated in the previous steps. (3) Combining the P term and the I term, the PI optimizer achieves better performance. The existence of the P term reduces the affect of the overshoot problem of the I term, thus obtains shorter distance to the optimum point than SGD-Momentum in the same number of optimization steps. (4) We also make a comparison for the ID optimizer that equates to the method proposed in [4]. The result shows that the effect of the D term eases the overshoot problem. However, it is not as good as the PI optimizer according to the distance value. We suggest that it is because the D term is not as stable as the P term, which affect the optimization performance. (5) However, if we only combine the P term and the D term without the I term, the PD optimizer cannot obtain a satisfactory result. It demonstrates the importance of the I term, which is reasonable since the optimization is performed in a minibatch form. (6) Combing the P,I and D terms together, the final PID optimizer shows the best performance.

5.3. Experiments on MNIST

We trained a three-layer MLP on the MNIST dataset and showed the results in this section to validate the effectiveness of the proposed methods for training neural networks. Training details are the same for all the experiments and are shown in supplementary materials. Fig. 2 compares the curves of the validation accuracy for different nonadaptive optimization methods. The SGD-Momentum optimizer performs better than the SGD optimizer, which mainly owes to the effect of the momentum term. By adding the P term or the D term, the PI optimizer and the ID optimizer achieves better performance than the SGD-Momentum optimizer since they can ease the overshoot problem. The PI optimizer shows a little better performance than the ID optimizer, which confirms our argument that the P term is more suitable for the minibatch-based methods than the D term. Finally, the PID optimizer, which uses all the advantages of the three terms, achieves the

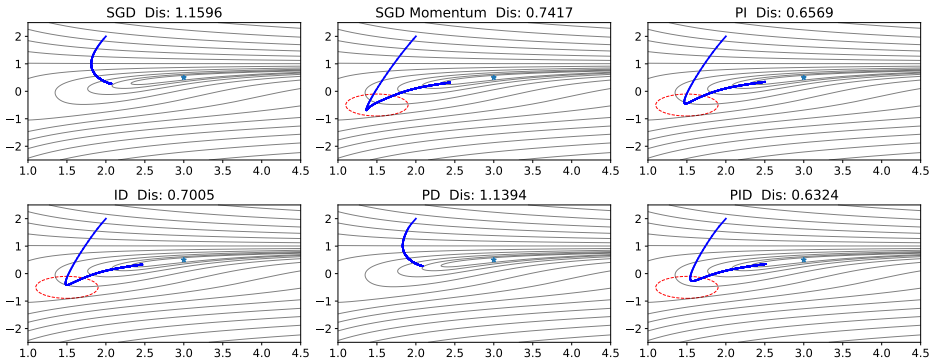

Fig. 1. Visualization of the trajectories (the blue line) on the contours of the loss surface of the Beale function for different optimization methods. The training process ends in 1000 steps. The pentagram represents the global minimum point of the Beale function. Dis represents the distance between the end point and the optimum point. The red circle emphases the overshoot problem.

Model	Depth/k	Params(M)	Error rates(%) on Cifar10/TinyImageNet					
			SGD	MSGD	MSGD*	PI	ID	PID
ResNet [17]	20/-	0.27	8.44/51.13	8.41/51.42	8.75/-	8.10/50.00	8.15/51.59	**7.98/49.67**
	110/-	1.70	7.01/48.72	6.57/46.45	6.43/-	6.06/45.26	6.44/45.65	**5.95/44.43**
PreActResNet [18]	110/-	1.7	6.63/45.04	6.26/44.76	6.37/-	5.92/43.52	6.16/44.48	**5.87/42.73**
	164/-	1.7	5.37/37.36	5.14/37.26	5.46/-	5.10/36.94	5.42/37.19	**4.75/36.85**
DenseNet-BC [19]	100/12	0.8	5.06/39.41	4.90/39.78	**4.51**/-	4.82/39.20	5.04/39.35	4.80/**38.79**
WRN [20]	16/8	11.0	5.20/38.70	4.93/38.43	**4.27**/-	4.92/37.41	5.07/37.77	4.82/**36.85**

Table 1. Test errors on Cifar10/TinyImageNet for different models using different optimization methods. MSGD represents the SGD-Momentum optimizer. The k of DenseNet denotes the growth rate of the network. The k of WRN denotes the widening factor of the network. $*$ indicates the results reported by the original papers. Bold results denote the best performance.

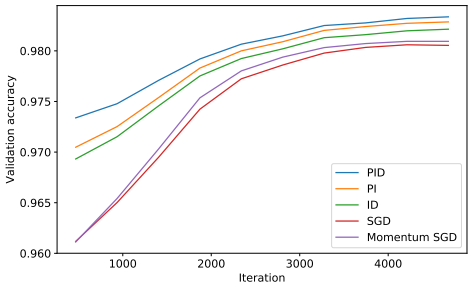

Fig. 2. The curves of the validation accuracy of MLP for various optimizers on MNIST.

best performance.

5.4. Comparison for deeper networks and larger datasets

To test the generality capability of our methods for deeper models and larger datasets, we select four types of models that are widely used in the image recognition field: ResNet [17], PreActResNet [18], DenseNet [19] and WRN [20]. They are trained in the same conditions using five types of optimizers on two larger datasets that have different scales: Cifar10 and TinyImageNet. The optimizers used for comparison includes the P (SGD) optimizer, the I (SGD-Momentum) optimizer, the PI optimizer, the ID optimizer and the PID optimizer. Training details are shown in supplementary materials. Tab. 1 shows that the results in two datasets are identical. The PI optimizer generally performs a little better than the ID optimizer under the same implementing conditions. The PI and PID optimizers generally perform better than the other optimizers, and the PID optimizer always shows the best performance. These results are also identical with the results in MNIST, but is obtained with deeper networks and larger datasets. This demonstrates the good generalization capability of our methods.

6. CONCLUSION

In this work, we propose a PID optimizer that can take advantage of the proportional term, integral term and derivative term at the same time to obtain optimal control of the optimization process. Two simple and effective strategies, i.e., the incomplete differential and the adaptive derivative output

clamping, are designed specially to reduce the interference of the high-frequency noise for the D term. Extensive experiments are performed on three datasets for various neural networks, and demonstrates the superiority and generality of our methods . Future works can focus on the adaptive PID that can change the K_p, K_i and K_d automatically based the past and present information. In addition, it is also worth exploring whether the hyperparameters of the PID optimizer can be set individually for each parameter of the model.

Acknowledgment

This work was supported by the State Grid Corporation Science and Technology Project (No.5200-201916261A-0-0-00).

7. REFERENCES

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems*, vol. 25, no. 2, pp. 1106–1114, 2012.

[2] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu, "Two-Stream Adaptive Graph Convolutional Networks for Skeleton-Based Action Recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 12026–12035.

[3] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu, "Skeleton-Based Action Recognition With Directed Graph Neural Networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019, pp. 7912–7921.

[4] Wangpeng An, Haoqian Wang, Qingyun Sun, Jun Xu, Qionghai Dai, and Lei Zhang, "A PID Controller Approach for Stochastic Optimization of Deep Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8522–8531.

[5] Léon Bottou, "Online learning and stochastic approximations," *Cambridge University Press*, pp. 9–42, 1998.

[6] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, 2013, pp. 1139–1147.

[7] Weijie Su, Stephen Boyd, and Emmanuel Candes, "A differential equation for modeling Nesterov's accelerated gradient method: Theory and insights," in *Advances in Neural Information Processing Systems*, 2014, pp. 2510–2518.

[8] John Duchi, Elad Hazan, and Yoram Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[9] Matthew D Zeiler, "ADADELTA: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[10] Diederik Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *International Conference for Learning Representations (ICLR)*, 2014.

[11] Ashia C. Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht, "The marginal value of adaptive gradient methods in machine learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4148–4158.

[12] Pratik Chaudhari and Stefano Soatto, "Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks," in *International Conference on Learning Representations*, 2018.

[13] John G. Ziegler and Nathaniel B. Nichols, "Optimum settings for automatic controllers," *trans. ASME*, vol. 64, no. 11, 1942.

[14] Kiam Heong Ang, Gregory Chong, and Yun Li, "PID control system analysis, design, and technology," *IEEE transactions on control systems technology*, vol. 13, no. 4, pp. 559–576, 2005.

[15] Kit-Sang Tang, Kim Fung Man, Guanrong Chen, and Sam Kwong, "An optimal fuzzy PID controller," *IEEE transactions on industrial electronics*, vol. 48, no. 4, pp. 757–765, 2001.

[16] Karl Johan Aström, Tore Hägglund, and Karl J. Astrom, "Advanced PID control," 2006.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep Residual Learning for Image Recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Identity Mappings in Deep Residual Networks," in *The European Conference on Computer Vision (ECCV)*, 2016, pp. 630–645.

[19] Gao Huang, Zhuang Liu, Kilian Q. Weinberger, and Laurens van der Maaten, "Densely Connected Convolutional Networks," *arXiv:1608.06993 [cs]*, Aug. 2016, arXiv: 1608.06993.

[20] Sergey Zagoruyko and Nikos Komodakis, "Wide Residual Networks," in *BMVC*, 2016.