









Impact of Mathematical Norms on Convergence of Gradient Descent Algorithms for Deep Neural Networks Learning

Linzhe Cai¹(✉) , Xinghuo Yu¹ , Chaojie Li² , Andrew Eberhard¹ ,
Lien Thuy Nguyen¹ , and Chuong Thai Doan¹ 

¹ School of Engineering, RMIT University, Melbourne, VIC 3000, Australia
s3548838@student.rmit.edu.au

² School of Electrical Engineering and Telecommunications, University of New South
Wales, Sydney, NSW 2052, Australia

Abstract. To improve the performance of gradient descent learning algorithms, the impact of different types of norms is studied for deep neural network training. The performance of different norm types used on both finite-time and fixed-time convergence algorithms are compared. The accuracy of the multiclassification task realized by three typical algorithms using different types of norms is given, and the improvement of Jorge's finite time algorithm with momentum or Nesterov accelerated gradient is also studied. Numerical experiments show that the infinity norm can provide better performance in finite time gradient descent algorithms and give strong robustness under different network structures.

Keywords: Infinity norm · Finite-time convergence · Norms equivalence · Deep neural network

1 Introduction

For a machine learning model, increasing the model complexity can effectively improve the learning ability. For models like neural networks, there are two obvious ways to increase complexity, one is to make the model wider and the other is to make the model deeper [1]. Shallow networks require exponentially increasing the number of units to achieve the same computational results compared with deep networks. Additionally, shallow networks need a good feature extractor that solves the selectivity-invariance dilemma [2], which can be avoided automatically when a deeper structure instead. From the perspective of topology, the transformation of a high-dimensional space by multiple activation functions

The authors were supported by the Australian Research Council (ARC) under Discovery Program Grant DP200101197.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
H. Aziz et al. (Eds.): AI 2022, LNAI 13728, pp. 131–144, 2022.
https://doi.org/10.1007/978-3-031-22695-3_10

makes the multi-classification problem linearly separable [3], thus the study of deep learning attracts more attention.

With the development of gradient descent-based algorithms, stochastic gradient descent (SGD) [4] provides a trade-off between accuracy and speed by modifying the size of the batch, while momentum [5] can help to meet dampen oscillation requirements by considering past velocity when updating. Nesterov accelerated gradient (NAG) [6] can further speed up the process by effectively looking ahead, the gradient of parameters in which with respect to the approximate future position instead of the current one. Other than modifying the direction, Adagrad [7] adapts the learning rate to parameters based on past gradients, reducing the learning rate when approaching the optimum. RMSprop [8] modifies the learning rate through dividing by an exponentially decaying average, solving the dramatically dropping problem. Adam [9] keeps both the adaptive learning rate like RMSprop and the direction adjustment like Momentum. However, most of them can only have asymptotic convergence, which means they cannot complete their learning within a reasonable time.

To solve the problem mentioned above, Recently, a series of algorithms appear to guarantee finite time convergence. Among them, Jorge first provides a kind of finite-time convergent learning algorithm, in particular, gradient flow (continuous gradient descent) through the gradient over the Euclidean distance (L_2 norm) of vectors [10]. After that, Wibisono gives a variant of which by adding a fraction on the Euclidean distance (q rescaled gradient flow) [11]. Besides, Romero and Benosman prove that it is indeed finite-time convergent [12]. Additionally, Garg proposes a fixed-time convergence algorithm that essentially splits the q -RGD into two parts [13]. Although a growing body of research has access to the mathematical norm on convergence, most of them only consider the Euclidean distance (L_2 norm) when rescaling the gradient flow. There is no study focusing on the effect of different types of norms with respect to convergence performance to the best of our knowledge.

This paper aims to study the impact of mathematical norms on the convergence of gradient flow for deep neural networks. Section 2 provides a review of different types of norms, the equivalence of norms, and convergence property. Section 3 gives numerical applications comparing different norms used on specific algorithms, and the potential improvement after involving momentum or NAG methods. Section 4 concludes.

2 Main Results

In this section, we first review the definition of mathematical norms and the most popular used norm types in Sect. 2.1, then give the equivalence of norms as well as the convergence property in Sect. 2.2. The qualitative analysis of different norms based on the expression of algorithms is given in Sect. 2.3, and the related works we used to compare in Sect. 3 are concluded in Sect. 2.4.

2.1 Mathematical Norms

Mathematically, a norm is a function from a vector space to the real numbers describing the distance from the origin, which is an abstract generalization of length [14]. According to the definition, a norm on a vector space \mathbb{R}^n is a real-valued function $\|\cdot\| : \mathbb{R}^n \rightarrow \mathbb{R}$ that meets the following properties [15]:

- Triangle Inequality: $\|x + y\| \leq \|x\| + \|y\|$ for all $x, y \in \mathbb{R}^n$.
- Absolute Homogeneity: $\|sx\| = |s| \|x\|$ for all $x \in \mathbb{R}^n$ and all scalars s .
- Positive Definiteness: for all $x \in \mathbb{R}^n$, if $\|x\| = 0$, then $x = 0$.

There are some typical types of norms given as follows [16]:

- L_1 norm (Taxicab norm): $\|x\|_1 := \sum_{i=1}^n |x_i|$.
- L_2 norm (Euclidean norm): $\|x\|_2 := \sqrt{x_1^2 + x_2^2 + \dots + x_n^2}$.

Both L_1 and L_2 norms are usually used as a regularization term to penalize large weights during logistic regression against the overfitting issue. While L_1 regularization penalizes the sum of the absolute values, L_2 regularization encourages the sum of the square of parameters to be small [17].

- L_p norm ($p \geq 1$): $\|x\|_p := (\sum_{i=1}^n |x_i|^p)^{1/p}$.

According to [[18], Theorem 3.5.4], L_p is a norm for $1 \leq p < \infty$. However, it will become a pseudo-norm for $0 < p < 1$, as it violates the triangle inequality property.

- L_∞ (Infinity Norm): $\|x\|_\infty := \max_i |x_i|$.

The infinity norm is essential for the limit of the L_p norm for $p \rightarrow \infty$. According to the expression of the L_p norm, we can figure out that the computation burden increases with the increase of the subscript of the norm symbol. However, after the functional limit operation, the computation of infinity norm as shown in the L_∞ norm only needs to iterate over through vector space once.

2.2 Equivalence of Norms

We recall from [[19], Definition 1.3] that two norms $\|\cdot\|_\alpha$ and $\|\cdot\|_\beta$ on a vector space \mathbb{R}^n are called equivalent if and only if there exist positive real numbers C and D such that for all $x \in \mathbb{R}^n$:

$$C \|x\|_\alpha \leq \|x\|_\beta \leq D \|x\|_\alpha. \quad (1)$$

A more precise relationship between different norms is obtained through Cauchy-Schwarz inequality and Hölder's inequality: for $p > r > 1$ on \mathbb{R}^n [20], we have

$$\|x\|_p \leq \|x\|_r \leq n^{1/r-1/p} \|x\|_p. \quad (2)$$

In particular,

$$\begin{aligned} \|x\|_2 &\leq \|x\|_1 \leq \sqrt{n} \|x\|_2, \\ \|x\|_\infty &\leq \|x\|_2 \leq \sqrt{n} \|x\|_\infty. \end{aligned} \quad (3)$$

According to [[19], Appendix A], the open subset of vector space \mathbb{R}^n defined by equivalent norms are the same, and the convergent sequences and their limits in \mathbb{R}^n defined by equivalent norms are the same. Similar statements are given in [21]: two finite-dimensional linear normed spaces with the same dimension are algebraically isomorphic and topologically homeomorphic. Thus, the convergence property is unchanged no matter what type of norm instead compared with the original algorithm under the Euclidean distance.

2.3 Different Norms Applications

As mentioned in Sect. 1, Jorge [10] and Wibisono [11] proposes finite-time convergence algorithms, Garg [13] provides a fixed-time convergence algorithm, and all of which are gradient flows involving the Euclidean norm (L_2 norm).

Jorge's finite-time convergence algorithm:

$$\frac{dw}{dt} = -\frac{\nabla_w J}{\|\nabla_w J\|_2}. \quad (4)$$

Wibisono's finite-time convergence algorithm:

$$\frac{dw}{dt} = -\zeta \frac{\nabla_w J}{\|\nabla_w J\|_2^{\frac{q-2}{q-1}}}, \quad (5)$$

where $q > 2$.

Garg's fixed-time convergence algorithm:

$$\frac{dw}{dt} = -C_1 \frac{\nabla_w J}{\|\nabla_w J\|_2^{\frac{p_1-2}{p_1-1}}} - C_2 \frac{\nabla_w J}{\|\nabla_w J\|_2^{\frac{p_2-2}{p_2-1}}}, \quad (6)$$

where $p_1 > 2$ and $1 < p_2 < 2$.

As all components in the vector share the same denominator, the relative size among different components in the vector has not changed. Thus, the back-propagation mechanism still works as the core of gradient descent is to figure out which component changes matter more.

According to the expressions of Eqs. (4), (5), and (6), all norms appear in the denominator of gradient flow and only have magnitude but without any direction. Thus, the potential step size when iteration operation will be inversely proportional to the relationship of the magnitude of different norms.

According to Eq. (3), the infinity norm obtains the smallest magnitude among all kinds of norms, which means it provides the largest potential step size after involving gradient flow. We can easily change the norm type in all these algorithms without changing the convergence property considering the convergence property discussed in Sect. 2.2.

2.4 Related Works

This section gives a brief review of the two most popular algorithms, stochastic gradient descent (SGD) [4] and Adam [9], which will be used as the benchmark for the first case study. Two direction adjustment methods, momentum [5] and Nesterov accelerated gradient (NAG) [6], are also introduced, which will be used to analyze the potential improvement for the second case study.

SGD [4] is an iterative approximation method calculated from a randomly selected subset ($50 < n < 256$) achieving faster iterations in trade for a lower convergence rate:

$$\theta = \theta - \eta \cdot \nabla_w J(\theta; x^{i:i+n}; y^{i:i+n}). \quad (7)$$

Momentum [5] accelerates SGD by adding a fraction ($\gamma < 1$) of the previous update vector, which obvious effect dampens oscillation when the gradient in one direction is larger than in others:

$$\begin{aligned} v_t &= \gamma \cdot v_{t-1} + \eta \cdot \nabla_w J, \\ \theta &= \theta - v_t. \end{aligned} \quad (8)$$

Nesterov [6] modifies the momentum one by calculating the gradient with respect to the estimated future position (moved by $\gamma \cdot v_{t-1}$) instead of the current one. The so-called look ahead essential considers the second derivative information of objective function:

$$\begin{aligned} v_t &= \gamma \cdot v_{t-1} + \eta \cdot \nabla_w J(\theta - v_t), \\ \theta &= \theta - v_t. \end{aligned} \quad (9)$$

Adam [9] computes adaptive learning rates, storing an exponentially decaying average (β_2) of past squared gradients and keeping another similar hyperparameter (β_1) on past gradients themselves, which is commonly considered fairly robust to hyperparameter selection:

$$\begin{aligned} m_t &= \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot \nabla_w J, \\ v_t &= \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot \nabla_w J \odot \nabla_w J. \end{aligned} \quad (10)$$

Additionally, bias correction is considered to offset the shift to the initial value at the beginning of the iteration:

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1 - \beta_1^t}, \\ \hat{v}_t &= \frac{v_t}{1 - \beta_2^t}, \end{aligned} \quad (11)$$

thus,

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} \cdot \hat{m}_t. \quad (12)$$

where ϵ is a smoothing term that avoids singularity.

3 Case Studies

As the Stanford vision and learning lab [22] summarizes that two recommended updates to use for CNN learning in visual recognition are either SGD with Nesterov momentum or Adam. Two case studies with quantitative analysis apply to comparing their performance with different norm-based finite-time algorithms. Section 3.1 compares the accuracy of different norms used in three typical algorithms under ResNet50 architecture, while Sect. 3.2 shows the improvement of Jorge’s finite-time involving momentum and Nesterov accelerated gradient (NAG) under a six-layers convolutional network.

3.1 Three Typical Algorithms Using Different Types of Norms

As one of the most popular used image classification databases, CIFAR100 [23] is considered as an example, and the 50-layer ResNet learning framework [24] is introduced to complete the task. We compare the performance of algorithms mentioned in Sect. 2.3 using different types of norms (L_1 , L_2 , L_3 , and L_∞) in each respectively. SGD [4] and Adam [9] are attached as the benchmark. The η and n for SGD in Eq. (7) are fixed at 0.01 and 128, while the η , β_1 , β_2 , and ϵ for Adam in Eqs. (10), (11), and (12) are 0.001, 0.9, 0.999, and 10^{-8} respectively.

Figure 1 gives the average value of training and testing accuracy of the CIFAR100 database under ResNet50 architecture using three different algorithms within different types of norms. According to Fig. 1a and 1b, Jorge’s finite-time algorithm [10] using the L_1 norm instead almost cannot converge under the same step size, while L_2 norm one obtains a reasonable convergent speed. Infinity norm one can obtain the best performance. When focusing on the training result given in Fig. 1a, the performance of SGD and Adam are between L_2 norm one and L_3 norm, while for the testing result given in Fig. 1b, Adam can beat the L_3 norm after 30 epochs. Additionally, the advantage from infinity one to L_3 norm is more obvious in testing accuracy compared with the training one.

Figures 1c and 1d give the accuracy using Wibisono’s finite time algorithm [11] within different types of norms, and the q given in Eq. (5) is chosen as 6. As we aim to figure out the improvement from the L_2 norm to the infinity one, only the consistency of the parameter chosen before and after changing the norm types is necessary, while the parameters are not necessarily obtained the best performance. When focusing on the training result given in Fig. 1c, the difference accuracy between L_2 norm one and Adam is relatively smaller compared with Jorge’s one Fig. 1a, while the difference accuracy between L_3 norm and infinity one is relatively larger. However, the improvement from L_2 norm to infinity one in Wibisono’s finite-time is less significant compared with Jorge’s one.

Figures 1e and 1f give the performance of the CIFAR100 database under ResNet50 architecture using Garg’s fixed-time algorithm [13] within different types of norms, and the p_1 and p_2 given in Eq. (6) are chosen as 3 and 1.5 respectively. Again, we only maintain the parameter consistency before and after changing the norm types but do not necessarily choose the optimal value. When

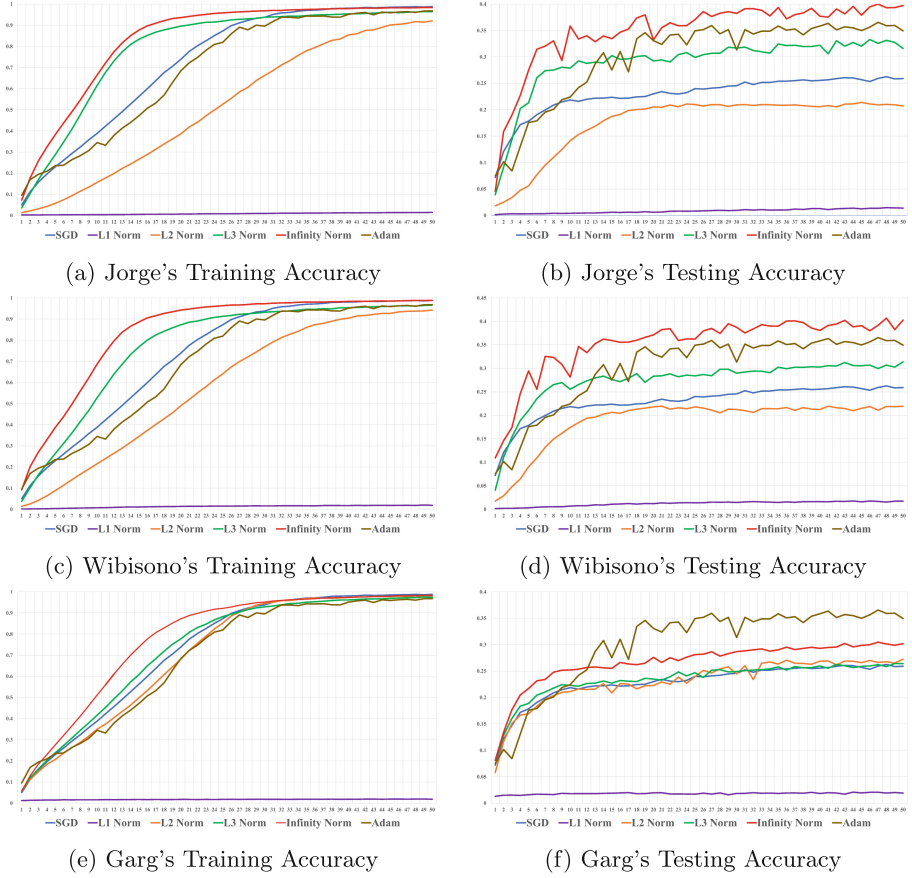


Fig. 1. ResNet50 CIFAR100 performance under different norms

Table 1. Improvement from L_2 norm to infinity one for different algorithms

	Statistics	Jorge	Wibisono	Garg	Jorge-v	Wibisono-v	Garg-v
$L_\infty - L_2$	Max	0.6109	0.5492	0.2441	0.2372	0.2039	0.0562
	Min	0.0596	0.0460	-0.0032	0.0274	0.0920	0.0179
	Median	0.3154	0.2383	0.0269	0.1732	0.1687	0.0338
	Mean	0.3209	0.2615	0.0739	0.1721	0.1636	0.0353
$\frac{L_\infty - L_2}{L_2}$	Max	721.43%	695.65%	51.29%	533.73%	528.45%	39.84%
	Min	6.87%	4.88%	-1.14%	61.81%	61.46%	7.65%
	Median	77.92%	46.96%	4.18%	86.99%	82.24%	15.73%
	Mean	169.30%	119.93%	16.64%	126.54%	108.38%	16.18%

focusing on the training result given in Fig. 1e, the performance of the L_2 norm can almost beat the Adam one, while Fig. 1f shows that the improvement from the L_2 norm to infinity one in Garg’s fixed-time algorithm is limited. Specifically, the performance of Garg’s fixed-time algorithm under almost all types of norms (excluding the L_1 one) is between SGD and Adam after 20 epochs, which means that the improvement room from L_2 norm to infinity one in Garg’s fixed-time algorithm is further compressed.

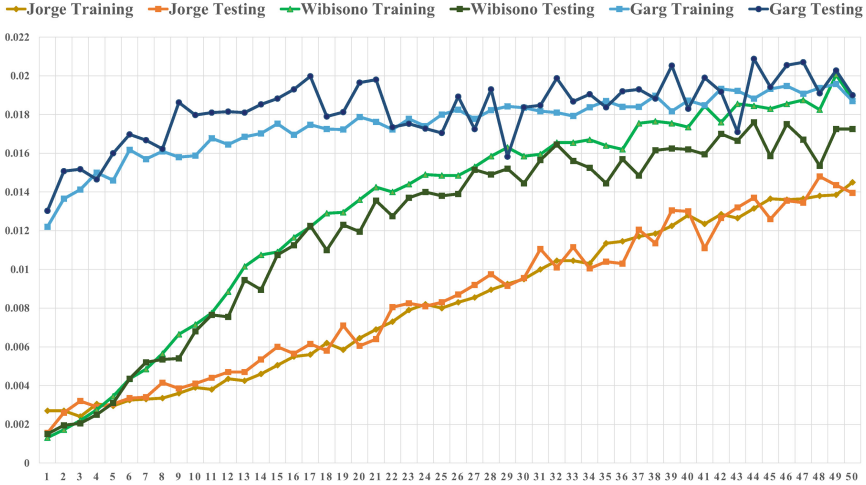


Fig. 2. Accuracy of CIFAR100 database using different algorithms under L_1 norm

According to Fig. 1, L_1 norm-based iteration provides the worst performance for the lowest accuracy for all three algorithms. To indicate they are convergent slower instead of cannot obtain the convergence property, Fig. 2 extracts the accuracy of different algorithms using the L_1 norm from Figure 1. The darker chroma of the same color represents the accuracy difference from training to testing. According to Fig. 2, all three algorithms can converge but with a slower step size, and the difference between the training database and the testing one is relatively small. Although Garg’s fixed-time algorithm has a slightly higher original accuracy, it provides the least improvement from the perspective of absolute numerical.

Table 1 gives statistical data on the differences from L_2 norm to infinity one at corresponding iteration times for different algorithms respectively. The absolute values indicate the absolute accuracy improvement from L_2 to infinity norm, while the relative values indicate the absolute differences over corresponding L_2 norm accuracy (percentage improvement). A positive value means that the accuracy of the infinity norm used on the algorithm is higher than Euclidean one, while the negative value implies that the L_2 norm may have higher accuracy at a specific iteration time. The algorithms without -v in the column express

the training accuracy, while the algorithms with -v in the column express the validation accuracy (testing datasets).

According to Table 1, the infinity norm used in all algorithms can have an improvement from its original one (L_2 norm). Among them, Garg’s fixed-time algorithm provides the smallest improvement, and Jorge’s finite time algorithm has slightly more improvement than Wibisono’s. Additionally, for a specific algorithm, the improvement in training datasets is always more obvious than which in testing one. The maximum improvement (7 times for Jorge’s and Wibisono’s training and (7 times for Jorge’s and Wibisono’s testing) usually appear in the first few iteration times, while the minimum differences (negative for Garg’s training accuracy) arise at the latest few steps.

In summary, the effects on different types of norms are obvious for Jorge’s finite-time algorithm, and the performance of which using infinity norm can surpass SGD and Adam for training and testing accuracy during the overall process. Although Wibisono’s finite-time algorithm with infinity norm also has similar accuracy, the dependency on the parameter chosen weakens its advantage.

3.2 Jorge’s Finite-Time Algorithm with Momentum and Nesterov

According to the brief review of gradient descent-based optimization given in Sect. 2.4, there are two mainstreams to refine an algorithm, namely iteration direction (eg. Momentum [5]) and adaptive learning rate (eg. RMSprop [8]). The finite time algorithm is essential one type of rescaled gradient flow, which means the improvement from the perspective of adaptive learning rate is already obtained. Thus, we are interested in whether the direction modification can further improve learning performance.

The second case study focuses on the improvement of Momentum [5] and Nesterov accelerated gradient (NAG) [6] methods used on finite time algorithms with different types of norms. The network structure considered in the case study is a six convolutional layers CNN (filter numbers 32, 32, 64, 64, 128, and 128) with batch normalization and dropout layers attached. To reduce the puzzle caused by the parameter chosen, Jorge’s finite-time algorithm is considered as an example.

Figure 3 gives the testing accuracy of the CIFAR100 database using the L_2 norm and L_3 norm with different fractions of momentum (γ in Eq. (8)) or Nesterov (γ in Eq. (9)) accelerate respectively. SGD is still considered as a benchmark. As for the infinity norm, the different performance among different fractions is too small to illustrate, more statistical details will give in Table 3.

According to Fig. 3, the difference between momentum (Figs. 3a, 3c, and 3e) and corresponding Nesterov (Figs. 3b, 3d, and 3f) under the same fraction value is not obvious. Besides, the improvement after involving momentum and Nesterov is outstanding on the L_2 norm-based Jorge’s finite-time algorithm as seen in Figs. 3c and 3d. However, the accelerated effect is reduced on the L_3 norm-based one as seen in Figs. 3e and 3f. The enhancement of SGD after adding momentum is between the L_2 norm and the L_3 norm.

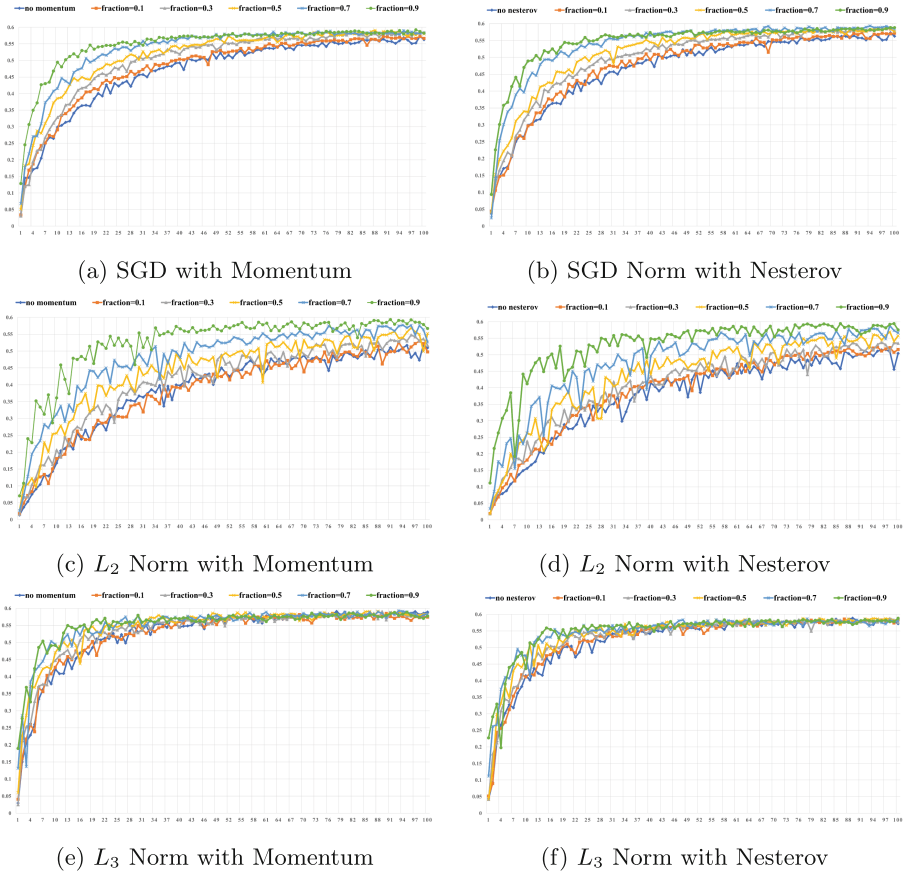


Fig. 3. CIFAR100 testing accuracy under six convolutional layer structure

Table 2. Improvement of momentum and Nesterov for different types of norms

	Statistic	SGD-M	SGD-N	L_2 -M	L_2 -N	L_3 -M	L_3 -N
$f_{0.9} - f_0$	Max	0.2216	0.2085	0.2772	0.2993	0.1634	0.1977
	Min	0.0113	0.0106	0.0567	0.0563	-0.0172	-0.0690
	Median	0.0646	0.0571	0.1349	0.1356	0.0071	0.0134
	Mean	0.0802	0.0773	0.1406	0.1499	0.0270	0.0277
$\frac{f_{0.9} - f_0}{f_0}$	Max	248.24%	154.20%	413.87%	534.09%	540.54%	428.37%
	Min	1.98%	1.86%	11.26%	11.01%	-2.97%	-25.89%
	Median	12.64%	11.48%	31.25%	32.02%	1.27%	2.38%
	Mean	24.71%	23.11%	56.46%	62.38%	12.61%	12.03%

Table 2 concludes the improvement from no momentum to 0.9 fractions (best performance under all subfigures) of both absolute and relative values for SGD, L_2 norm, and L_3 norm respectively. The algorithms with M in the column indicate adding momentum term, and the algorithms with N in the column indicate adding Nesterov term.

When we look at the absolute difference, while the improvement of the L_2 norm is almost double compared with which in SGD, the mean value of SGD is triple compared with the L_3 norm on average and quadruple compared with which median value. As for the relative value, the average improvement of SGD is only double compared with the L_3 norm, but the median value difference between them is five times. Again, the maximum improvement usually appears in the first few iteration times, while the minimum differences arise at the latest few steps. As the extreme value has relatively greater contingency, the statistical significance of which is weakened.

Although SGD, L_2 norm, and L_3 norm obtain the best performance when fractions equal to 0.9, the same fraction gives the worse performance when it comes to the infinity norm, while other fractions almost have no effect on it (as seen in Table 3), which may be caused by the radical acceleration in the same direction, as the increased dimensions almost doubled.

Table 3. Effect of momentum and NAG for infinity norm with different fraction

	Values	Statistic	$f = 0.1$	$f = 0.3$	$f = 0.5$	$f = 0.7$	$f = 0.9$
M	$f_x - f_0$	Mean	0.0042	0.0006	0.0046	-0.0192	-0.0254
		Median	0.0015	0.0003	0.0051	-0.0213	-0.0205
	$\frac{f_x - f_0}{f_0}$	Mean	1.33%	0.51%	0.88%	-3.16%	-5.04%
		Median	0.27%	0.06%	0.91%	-3.72%	-3.57%
N	$f_x - f_0$	Mean	0.0033	0.0005	-0.0017	-0.0006	-0.0143
		Median	0.0051	0.0009	-0.0022	-0.0017	-0.0098
	$\frac{f_x - f_0}{f_0}$	Mean	0.46%	0.11%	-0.22%	0.03%	-3.05%
		Median	0.89%	0.15%	-0.38%	-0.29%	-1.72%

Table 3 concludes the influence of momentum and Nesterov for infinity norm gradient flow under different fractions, where f_x represents the fraction value (γ) in Eqs. (8) and (9).

Among all fraction choices, 0.1 obtain the best performance while the improvement is still limited. The influence is negligible when the fraction is chosen between 0.3 and 0.5. When the fraction comes to 0.7, momentum shows negative effects and even worst when 0.9 is chosen. NAG effectively mitigates the negative effect of momentum by looking ahead effectively. When close to the optimum value, the gradient current time should be smaller than the previous one, and there is reason to believe that it will continue to be smaller, which justifies the radical acceleration deduction.

To have a more intuitive comparison, Fig. 4 concludes the best performance of each norm from Fig. 3 (L_2 norm and L_3 norm with 0.9 fraction Nesterov, SGD with 0.9 fraction momentum, and infinity norm without fraction).

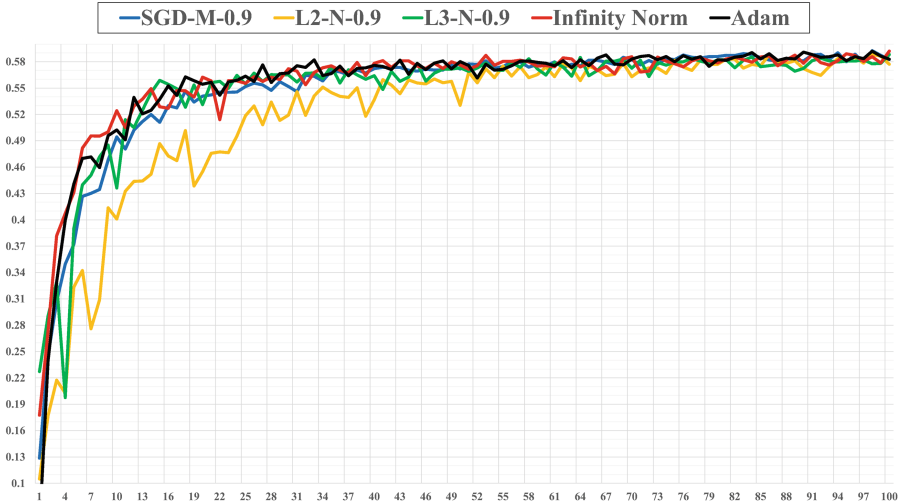


Fig. 4. Highest accuracy of CIFAR100 using different types of norms (Color figure online)

According to Fig. 4, the L_2 norm gives the worst performance (in yellow) while the infinity norm gives the best (in red), while the L_3 norm and the SGD are between the two mentioned above. Although the improvement of the L_2 norm after involving momentum is significant, it cannot surpass the infinity one, which can be imaged as an invisible ceiling existing (infinity norm gradient flow without momentum) no matter what type of norm choose. Thus, the less improvement of the L_3 norm given in Figs. 3e and 3f can be explained as the difference between the L_3 norm and the infinity one being small, thus there is no room for momentum and Nesterov to improve the performance. In other words, the better the performance without momentum or Nesterov acceleration, the less it can be improved through the dampens oscillation methods. Thus, the infinity norm used on Jorge’s finite-time algorithm can cover the benefits of Momentum without introducing the updated velocity in the past time, which saves computing costs.

The performance of Adam is also plotted in Fig. 4. Although the accuracy of Adam (in black) and infinity norm gradient flow (in red) is similar, there are no hyperparameters that needed to be adjusted (Jorge’s finite-time) related to the infinity norm gradient flow (INGF), and no memory requirement (no momentum or NAG need). Specifically, the average running time of Adam is 13.4% longer than which of the INGF (914.75 s and 806.62 s respectively) under the same GPU model (NVIDIA GeForce RTX 2080 Super with Max-Q Design

under CUDA 10.0 support). From that perspective, INGF is superior to Adam which needs not only adaptive learning rates (computational burden) but also history records (memory burden).

4 Conclusion

In this paper, the comparison of different types of norms used in finite-time convergence algorithms is obtained. Qualitative analysis after the equivalence of norms with the help of convergence property verifies the convergence rate. The performance of three typical algorithms using different types of norms is quantitatively analyzed for image classification using the CIFAR100 database under the ResNet50 architecture. Jorge's finite-time algorithm gives the maximum improvement after changing the Euclidean norm to the infinity one. The improvement of Jorge's finite-time algorithm with momentum and Nesterov is studied. Although the better original performance, the less improvement after momentum or Nesterov acceleration involving, infinity norm gradient flow (INGF) without momentum still keeps overwhelming superiority. Although INGF can not always be superior to Adam in accuracy, no hyper-parameters adjustment and no memory requirement of INGF can keep its favorable position in time-consuming compared with Adam. According to the results given in case studies, we have reason to believe that Jorge's finite-time algorithm with infinity norm can provide reliable performance (higher accuracy and less time) for CNN learning tasks, especially visual recognition.

References

1. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016)
2. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. *Nature* **521**(7553), 436–444 (2015)
3. Olah, C.: Neural networks, manifolds, and topology. Blog post (2014)
4. Bottou, L., Bousquet, O.: The tradeoffs of large scale learning. In: *Advances in Neural Information Processing Systems* 20 (2007)
5. Qian, N.: On the momentum term in gradient descent learning algorithms. *Neural Netw.* **12**(1), 145–151 (1999)
6. Nesterov, Y.: A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$. In: *Doklady an ussr*, vol. 269, pp. 543–547 (1983)
7. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* **12**(7) (2011)
8. Tieleman, T., Hinton, G.: Neural networks for machine learning. Technical report (2011). <http://www.cs.toronto.edu/tijmen/csc321/slides/lectureslideslec6.pdf>
9. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint [arXiv:1412.6980](https://arxiv.org/abs/1412.6980) (2014)
10. Cortés, J.: Finite-time convergent gradient flows with applications to network consensus. *Automatica* **42**(11), 1993–2000 (2006)
11. Wibisono, A., Wilson, A.C., Jordan, M.I.: A variational perspective on accelerated methods in optimization. *Proc. Natl. Acad. Sci.* **113**(47), E7351–E7358 (2016)

12. Romero, O., Benosman, M.: Finite-time convergence in continuous-time optimization. In: International Conference on Machine Learning, pp. 8200–8209. PMLR (2020)
13. Garg, K., Panagou, D.: Fixed-time stable gradient flows: applications to continuous-time optimization. *IEEE Trans. Autom. Control* **66**(5), 2002–2015 (2020)
14. Gradshteyn, I.S., Ryzhik, I.M.: Table of integrals, series, and products. Academic Press (2014)
15. Pugh, C.C.: Real Mathematical Analysis, vol. 2011. Springer, Cham (2002). <https://doi.org/10.1007/978-0-387-21684-3>
16. Weisstein, E.W.: Vector norm (2002). <https://mathworld.wolfram.com/>
17. Ng, A.Y.: Feature selection, l_1 vs. l_2 regularization, and rotational invariance. In: Proceedings of the Twenty-First International Conference on Machine learning, p. 78 (2004)
18. Wassermann, A.J.: Functional analysis (1999)
19. Conrad, K.: Equivalence of norms. In: Expository Paper, University of Connecticut, Storrs, heruntergeladen von, vol. 17, no. 2018 (2018)
20. Golub, G.H., Van Loan, C.F.: Matrix Computations. JHU Press, Baltimore (2013)
21. Gongqing, Z., Yuanqu, L.: Functional Analysis Lecture Notes. Peaking University Press (1990). (in Chinese)
22. Karpathy, A.: Cs231n convolutional neural networks for visual recognition (2017). cs231n.github.io. Dostopno na. <http://cs231n.github.io>
23. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
24. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)