Journal of Data Acquisition and Processing
ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.778188

AN EXAMINATION OF PARTIAL DERIVATIVES IN THE CONTEXT OF MACHINE LEARNING, SUCH AS IN BACKPROPAGATION AND TRAINING NEURAL NETWORKS

Ghorbanzadeh Mohammad[1], Majid Hameed Hadi[2]
[1]Department of Mathematics, Imam Reza International University, Mashhad, Iran
[2]Engineering of communication system, Imam Reza International University, Mashhad, Iran

Abstract
Partial derivatives play a critical role in the field of machine learning, specifically in the context of backpropagation and training neural networks. This paper provides an in-depth examination of the concept of partial derivatives and their applications in machine learning. We summarize the key findings of this paper, which include the use of partial derivatives in gradient computation, activation functions, normalization techniques, optimization algorithms, and regularization techniques. Additionally, we discuss the challenges associated with the use of partial derivatives, such as the vanishing gradient problem and numerical instabilities during backpropagation. Our analysis highlights the importance of choosing appropriate activation functions, normalization techniques, and optimization algorithms to enhance the efficiency and accuracy of partial derivative computations. Furthermore, we explore the impact of newer activation functions and regularization techniques on neural network performance. The insights provided in this review paper can assist researchers and practitioners in designing and implementing more effective machine learning models.

1. Introduction

Partial derivatives are crucial for neural network training in machine learning, with backpropagation being a widely used method [1]. The algorithm estimates the error gradient concerning each network weight using an error measure, such as mean squared error, to reduce the discrepancy between anticipated and actual output. The gradient, a vector of partial derivatives, is used to adjust the network's weights [2]. Gradient descent updates the weights using the chain rule of calculus [3].

1.1. Motivation for the paper

Despite the importance of partial derivatives in backpropagation, there has been little investigation into their properties within machine learning. Understanding their behavior can lead to better optimization algorithms and improved neural network performance. This paper aims to examine partial derivatives in machine learning, particularly in backpropagation and neural network training, to provide insights for designing more effective algorithms.

1.2. Background on partial derivatives and machine learning

Partial derivatives are used in machine learning, particularly for neural network training, to enhance algorithm efficiency. The backpropagation technique relies on partial derivatives to calculate the error gradient and update network weights [4]. They are also employed in neural network design, regularization methods, and optimization algorithms, making it essential to understand their role for effective machine learning [5].

1.3. Problem Statement

Neural networks require a complex and poorly understood training procedure, with backpropagation being a key method that adjusts weights via partial derivatives. Many practitioners lack a thorough understanding of the underlying principles and techniques, leading to suboptimal performance and interpretability difficulties. This paper aims to provide a comprehensive analysis of partial derivatives in machine learning, particularly in backpropagation training, to improve neural network performance and interpretability.

1.4. Overview of the paper

This paper investigates the role of partial derivatives in machine learning, focusing on neural network training using backpropagation. The paper covers the definition and applications of partial derivatives, their importance in backpropagation, and optimization techniques that alter them to improve performance. It also discusses various applications in machine learning, such as speech recognition, image recognition, natural language processing, and robotics. The paper concludes with the main findings, limitations, and future research directions.

2. Partial Derivatives in Machine Learning

Backpropagation, a prevalent neural network training algorithm, relies on partial derivatives to optimize weights during training [1]. Using the chain rule, backpropagation calculates partial derivatives of the loss function concerning each weight, enabling weight updates that minimize the loss. Other techniques, like gradient descent and stochastic gradient descent, also use partial derivatives to iteratively update model parameters [6].

Partial derivatives play a role in feature selection, model interpretation, and optimization techniques. In general, optimization and interpretation of machine learning models depend on partial derivatives, which this article explores in the context of backpropagation and neural network training [5].

2.1. Definition of partial derivatives

Partial derivatives are essential in calculus and mathematical analysis, measuring how a function changes as its inputs change one at a time. In machine learning, they are vital for training neural networks using backpropagation [7].

$$\partial f/\partial xi = lim\ (h \rightarrow 0)\ [f(x1, x2, ..., xi + h, ..., xn) - f(x1, x2, ..., xi, ..., xn)] / h$$

where h is a small positive number.

While keeping all other inputs constant, the partial derivative calculates how responsive the function f is to changes in the input variable xi. The partial derivative is used in machine learning to calculate the gradient of a loss function with respect to a neural network's weights, which are updated during backpropagation.

Computing partial derivatives can be challenging for complex neural networks; numerical methods like finite differences or automatic differentiation can approximate them.

2.2. Applications of partial derivatives in machine learning

Partial derivatives are crucial for training neural networks using backpropagation, which adjusts weights and biases during training, and regularization methods like L1, L2, and dropout regularization [1, 8, 9]. They also optimize hyperparameters and analyze neural networks, such as in sensitivity analysis to identify important features or detect biases.

Overall, partial derivatives are fundamental in machine learning, and understanding their role is essential for developing new algorithms and improving existing ones [9].

2.3. Importance of partial derivatives in backpropagation

Backpropagation heavily relies on partial derivatives, using the chain rule of calculus to compute the gradients of the loss function concerning the neural network's parameters [2]. Accurate computation of partial derivatives is crucial for successful neural network training, as even small errors can accumulate over time, causing suboptimal convergence or failure.

Partial derivatives in backpropagation can also be affected by activation function selection, such as the vanishing gradient problem with sigmoid activation functions [10]. In conclusion, understanding the significance of partial derivatives and their computation is essential for effectively training deep learning models.

3. Backpropagation and Training Neural Networks

Training neural networks using backpropagation requires careful selection of the loss function, activation function, optimization algorithm, and hyperparameters. Enhancing generalization and convergence can be achieved through regularization, dropout, and batch normalization [11]. Adaptive learning rates, momentum, and second-order methods have improved the backpropagation algorithm's efficiency, leading to deep neural networks capable of solving complex problems [12].

3.1. Introduction to backpropagation

Backpropagation, based on partial derivatives, calculates the gradient of the loss function concerning a neural network's weights and biases. This allows for gradient descent optimization of the network's parameters. Backpropagation iteratively updates weights and biases, minimizing the loss function [2]. The algorithm, introduced by Rumelhart, Hinton, and Williams in 1986 [2], has become fundamental for training deep neural networks. Techniques like batch normalization [12] have significantly improved its efficiency.

3.2. Algorithm of backpropagation

Backpropagation has two parts: the forward pass and the backward pass. The forward pass processes input data through the network, comparing output to the desired output. The backward pass computes partial derivatives of the loss function concerning weights and biases, updating them using optimization methods like gradient descent [1], [9]. This iterative technique enables efficient training of deep neural networks [2], [1], [11].

3.3. Mathematical explanation of backpropagation

For a feedforward neural network with L layers, backpropagation is expressed as follows:

A. Randomly initialize the network's weights and biases.

B. Calculate the network output for each training example using forward propagation.

C. Compute the error using a loss function.

D. Compute the partial derivatives of the loss function concerning the network's weights and biases.

E. Adjust the biases and weights based on the partial derivatives and a learning rate.

F. Repeat steps 2-5 for a fixed number of epochs or until convergence.

Partial derivatives in step 4 are computed recursively using the chain rule [1].

- For the output layer:

$$\frac{\partial L}{\partial z^{[L]}} = \frac{\partial}{\partial z^{[L]}} \mathcal{L}(\hat{y}, y)$$

$$\frac{\partial L}{\partial a^{[L]}} = \frac{\partial L}{\partial z^{[L]}} \frac{\partial z^{[L]}}{\partial a^{[L]}}$$
$$\frac{\partial L}{\partial w^{[L]}} = a^{[L-1]} (\frac{\partial L}{\partial z^{[L]}})^T$$
$$\frac{\partial L}{\partial b^{[L]}} = \frac{\partial L}{\partial z^{[L]}}$$

- For hidden layers:

$$\frac{\partial L}{\partial z^{[l]}} = (\frac{\partial L}{\partial z^{[l+1]}})(W^{[l+1]})^T \odot g'(z^{[l]})$$
$$\frac{\partial L}{\partial a^{[l]}} = \frac{\partial L}{\partial z^{[l]}} \frac{\partial z^{[l]}}{\partial a^{[l]}}$$
$$\frac{\partial L}{\partial w^{[l]}} = a^{[l-1]} (\frac{\partial L}{\partial z^{[l]}})^T$$
$$\frac{\partial L}{\partial b^{[l]}} = \frac{\partial L}{\partial z^{[l]}}$$

where $\mathcal{L}(\hat{y}, y)$ is the loss function, $\hat{y}$ is the predicted output of the network, y is the actual output, $z^{[l]}$ is the pre-activation of layer l, $a^{[l]}$ is the activation of layer l, $w^{[l]}$ and $b^{[l]}$ are the weights and biases of layer l, g is the activation function, and $\odot$ represents element-wise multiplication.

These partial derivatives are used to update the weights and biases in step 5 of the algorithm. The learning rate determines the step size of the update and can have a significant impact on the convergence of the algorithm.

These derivatives are crucial for updating weights and biases in step 5. The learning rate determines the step size of the update and impacts the algorithm's convergence.

3.4. Importance of partial derivatives in backpropagation

Partial derivatives are essential in backpropagation, directing the optimization process and enabling the training of neural networks. Without partial derivatives, the algorithm couldn't modify the network's weights to minimize the discrepancy between expected and actual outputs. Techniques like batch normalization and regularization also rely on partial derivatives [13], [6]. Thus, understanding partial derivatives and their role in backpropagation is crucial for developing successful deep learning models.

4. Optimization Techniques for Backpropagation with Modified Partial Derivatives

Efficient deep neural network training relies on optimizing the backpropagation process. Regularization techniques such as L1, L2, and dropout are commonly used to prevent overfitting. Other optimization techniques include batch normalization, momentum-based optimization, adaptive learning rate algorithms, and second-order optimization methods. Appropriate weight initialization techniques also improve the training process [14].

Several ways to optimize backpropagation involve modifying partial derivatives:

4.1. Different Activation Functions:

Activation functions like ReLU, leaky ReLU, and maxout impact partial derivatives computed in backpropagation. These functions result in sparse activations, enabling more efficient

backpropagation. The derivatives of ReLU and leaky ReLU are computationally efficient to calculate [15], [16].

Expectation: Functions like ReLU, leaky ReLU, or maxout can lead to faster convergence, improved accuracy, and more stable neural network training.

Manifestations: Studies suggest that activation functions like ReLU, leaky ReLU, and maxout positively impact backpropagation during neural network training [17], [18]. Gal et al. (2014) [19] found that using ReLU in latent variable and sparse Gaussian process regression models can enhance computational effectiveness. The choice of activation function significantly affects neural network performance during backpropagation, with recent functions often leading to better training results.

4.2. Normalization Techniques:

Batch normalization (BN) and layer normalization (LN) optimize backpropagation by altering partial derivatives. BN normalizes layer activations across the entire batch, reducing internal covariate shift [12]. LN normalizes layer activations across the features dimension instead of the batch dimension [20]. Both BN and LN improve neural network performance in tasks like image classification and machine translation.

Expectation: Normalization techniques can improve training stability, reduce learning rate dependence, and result in faster convergence and better network accuracy.

Manifestation: References support normalization techniques' ability to improve training stability, reduce learning rate dependence, and enhance network accuracy [12], [20], [21], [22], [23]. Empirical evidence shows that neural networks perform better with normalization techniques like BN and LN. New parameters (gamma and beta) introduced during normalization help further optimize the training process and improve network performance.

4.3. Gradient Clipping:

Large partial derivatives can cause numerical instabilities in backpropagation. Gradient clipping restricts gradient magnitudes to a pre-defined maximum value, stabilizing the backpropagation process [1], [24], [25], [26].

Expectation: Gradient clipping stabilizes backpropagation, prevents numerical instabilities, and improves network stability and accuracy.

Manifestation: Large partial derivatives can cause instabilities, leading to slower training and lower accuracy. Gradient clipping stabilizes backpropagation by restricting gradient magnitudes. Studies support gradient clipping's role in improving deep neural network stability [27], [12], [28]. Zhang et al. (2021) [29] found that gradient clipping can improve generalization performance but cautioned its potential to cause overfitting in some cases.

4.4. Adaptive Learning Rates:

Stochastic gradient descent (SGD) utilizes adaptive learning rates to enhance convergence speed and precision. These rates modify the step size based on the objective function's properties. Adam, Adagrad, RMSProp, and Adadelta are popular adaptive learning rate algorithms.

Partial derivatives of the loss function can inform adaptive learning rate algorithms by providing second-order information like the Hessian matrix, which is used to adjust the learning rate. [30], [31] The mathematical formulation can be found in Kingma and Ba (2014) [32] and Duchi et al. (2011) [33].

Expectation: Adaptive learning rates can accelerate network convergence, reduce dependence on learning rate hyperparameters, and increase network stability and accuracy.

Manifestation: Adaptive learning rate algorithms like Adam, Adagrad, RMSProp, and Adadelta enhance neural network convergence and accuracy by adaptively modifying learning rates. They use gradient magnitudes and second-order information like the Hessian matrix. Partial derivatives inform the algorithms and help improve network performance. This is supported by Ruder (2016) [5], Reddi et al. (2019) [34], and Zeiler (2012) [30].

4.5. Momentum and Nesterov Momentum:

Momentum and Nesterov momentum optimize SGD to speed up deep neural network training. Momentum accelerates SGD convergence by incorporating previous gradients, while Nesterov momentum predicts future positions to adjust gradient computation.

Partial derivatives of the loss function are essential for momentum and Nesterov momentum, informing the gradient's direction and objective function's rate of change [35, 36]. The mathematical expressions can be found in Polyak (1964) [37] and Sutskever et al. (2013) [28].

Expectation: Momentum and Nesterov momentum can quicken convergence, reduce oscillations, and enhance network stability and accuracy.

Manifestation: Implementing momentum and Nesterov momentum in optimization methods accelerates convergence and enhances neural network performance. Modifying partial derivatives is crucial for these techniques. References supporting this include Qian (1999) [36], Sutskever et al. (2013) [28], Nesterov (2003) [38], and Dozat (2016) [39].

4.6. Regularization Techniques:

Regularization techniques prevent deep neural network overfitting by modifying partial derivatives of the loss function. L2 regularization (weight decay) and dropout are popular regularization techniques.

Regularization aims to improve network generalization by balancing overfitting and underfitting. Tuning regularization parameters is essential for achieving the desired regularization level.

Expectation: Regularization techniques should improve network generalization but excessive regularization may result in underfitting.

Manifestation: Regularization techniques prevent overfitting by changing partial derivatives and encouraging smaller weights. This leads to a smoother decision boundary and less overfitting. References supporting this include Morgan et al. (1989) [41], Bishop and Nasrabadi (2006) [13], Srivastava et al. (2014) [9], Goodfellow et al. (2016) [1], and Hinton et al. (2012) [40].

In conclusion, optimizing backpropagation through partial derivative alterations involves modifying activation functions, normalization techniques, learning rates, regularization techniques, and employing optimization techniques like gradient clipping and momentum.

5. Applications of Partial Derivatives in Machine Learning

Partial derivatives are crucial in machine learning, especially in optimizing neural networks through backpropagation. This thesis explores their applications in speech recognition, image recognition, natural language processing, robotics, and more.

5.1. Image recognition

Reference [42] introduces neural networks and their fundamentals, including backpropagation and partial derivatives. It highlights their role in adjusting the network's weights to improve performance.

Reference [43] covers digital image processing techniques, such as using partial derivatives for edge detection and image segmentation, including neural network-based approaches.

Source [4] reviews the evolution of deep neural networks in speech recognition, image recognition, and natural language processing. It discusses how backpropagation and partial derivatives are used in training deep neural networks.

Reference [1] is a comprehensive book on deep learning, discussing partial derivatives in backpropagation and optimizing deep neural networks for various applications, including image recognition.

5.2. Natural language processing

Yahui Chen's Master's thesis [44] investigates Convolutional Neural Networks (CNNs) for phrase categorization in Natural Language Processing. She explains how partial derivatives and backpropagation are used to compute gradients for model optimization.

Reference [45] proposes a method to separate semantics and syntax from phrase embeddings using pre-trained language models. The authors employ partial derivatives to calculate each token's contribution to a sentence's overall meaning, outperforming previous methods.

Zhang and Wallace (2015) [46] present a sensitivity analysis of CNNs for sentence classification in NLP. They compute partial derivatives of the output with respect to each input word to quantify how the output changes. Their analysis shows CNNs can learn important features without explicit feature engineering and offers practical guidance for practitioners.

5.3. Speech recognition

Vesel et al. [47] optimize neural network parameters for speech recognition using partial derivatives. They propose the Minimum Phone Error (MPE) training criterion, which optimizes accuracy on a sequence level. The authors modify backpropagation to compute partial derivatives of the cost function, which optimizes the MPE metric and network parameters.

Graves et al. [48] introduce Connectionist Temporal Classification (CTC), a loss function enabling RNNs to map variable-length sequences without explicit segmentation. CTC sums over all alignments between input and output sequences and backpropagates the error gradient through the summed probabilities, improving RNN performance for speech recognition tasks.

Chan et al. [49] suggest the "Listen, Attend, and Spell" (LAS) architecture for conversational speech recognition. The LAS model combines a bidirectional RNN with an attention mechanism, focusing on crucial input segments. The authors investigate gradient computation techniques for LAS training, which require partial derivatives of the loss function concerning model parameters.

Goswami et al. [50] present DeepONet, a method fusing deep learning with partial differential equations (PDEs) to address speech recognition challenges. The approach enhances the DeepONet model's precision by altering partial derivatives in the backpropagation process.

5.4. Robotics

Mnih et al. [51] utilize deep reinforcement learning for robot control, proposing a neural network architecture that plays Atari games at human-level performance. Training requires partial derivatives in backpropagation and optimization methods affecting learning.

Peters et al. [52] use partial derivatives to optimize policy functions in reinforcement learning for humanoid robots. They employ the natural policy gradient approach, calculating the predicted reward's gradient concerning policy parameters, accelerating convergence and enhancing learning stability.

Gu et al. [53] propose a method for robotic reinforcement learning problems, extending Deep Q-Network (DQN) by predicting state transitions to speed up learning. The model-based approach involves training a neural network, computing partial derivatives of the loss function, and updating the model.

Bagnell and Schneider [54] introduce Policy Gradient with Parameter-Based Exploration (PGPE) for autonomous helicopter control. PGPE updates policy parameters based on the gradient of expected rewards concerning parameters.

5.5. Other fields

The study [55] improves meta-RL generalization by incorporating a learned objective function. The objective function captures task similarity, enabling better generalization. Gradients of the objective function are used to modify partial derivatives in the backpropagation algorithm.

The paper [57] proposes "physics-informed deep learning" (PIDL), combining deep learning and physics knowledge to solve PDEs. PIDL produces accurate, robust solutions for PDEs with limited data and outperforms traditional data-driven methods.

Several studies have examined partial derivatives in machine learning aspects, including optimization algorithms [34], reinforcement learning policy optimization [58, 59], unsupervised learning [61], and alternative techniques for computing partial derivatives in neural networks [62]. Other studies explore partial derivatives in reinforcement learning and generative models [63, 64].

In summary, partial derivatives are crucial in backpropagation and neural network training. Studying these references and exploring advancements in the field can deepen the understanding of partial derivatives' importance in machine learning and backpropagation.

6. Challenges and Future Directions

Partial derivatives are vital in machine learning algorithms but pose challenges and research opportunities. Vanishing or exploding gradients in deep neural networks hinder effective training with backpropagation. Techniques like residual connections, skip connections, and normalization layers show promise, but further research is required to understand the causes and solutions.

The trade-off between model expressiveness and interpretability is another challenge. Complex models like deep neural networks achieve high accuracy but are harder to interpret, while simpler models like linear regression or decision trees may be more interpretable but less powerful. Moreover, research on partial derivatives in reinforcement learning and other machine learning types beyond supervised learning is needed.

Despite the importance of partial derivatives, challenges remain. Addressing these challenges can advance the field of machine learning.

6.1. Limitations of current research

Current research on partial derivatives in machine learning has limitations. One issue is the reliance on the chain rule of calculus for computing partial derivatives in backpropagation. This

approach isn't always applicable in complex network architectures or non-Euclidean input/output spaces, requiring new mathematical tools and frameworks.

Neural networks' sensitivity to initialization and hyperparameters complicates model comparison. More robust techniques for training and optimizing hyperparameters are needed.

Data bias and fairness are crucial, as partial derivatives can amplify biases, making them hard to detect and correct. Ethical, transparent, and unbiased algorithms are necessary for future research.

6.2. Future research directions

Future research directions include investigating higher-order derivatives for faster convergence and better generalization, though this is computationally expensive and requires new mathematical tools.

Exploring partial derivatives in other machine learning methods, such as decision trees and support vector machines, can reveal recurring concepts and advance overall knowledge.

Partial derivatives' application to reinforcement learning presents challenges, especially in high-dimensional state and action spaces. New techniques and algorithms are required to improve reinforcement learning efficiency and effectiveness.

Lastly, developing interpretable and explainable models is vital. Understanding and explaining complex models using partial derivatives and other mathematical tools can lead to more reliable and robust machine learning systems.

In conclusion, the study of partial derivatives in machine learning offers numerous open questions and challenges. Addressing these can deepen our understanding and expand potential applications.

7. Conclusion

Partial derivatives are essential in machine learning, especially in backpropagation and neural network training. This paper provided an overview of partial derivatives, their applications, and related challenges.

7.1. Summary of key findings Key findings include:

1. Backpropagation employs partial derivatives to compute loss function gradients concerning network weights.
2. Various activation functions have pros and cons regarding partial derivative computations.
3. Batch and layer normalization reduce internal covariate shift, improving partial derivative computation efficiency.
4. Optimization algorithm choice impacts neural network training speed and accuracy.
5. The vanishing gradient problem occurs in deep networks when small partial derivatives hinder weight updates.
6. Regularization techniques like dropout can prevent overfitting and improve generalization.

7. Recent activation functions (e.g., ReLU, leaky ReLU, maxout) often yield faster, more stable training. BN and LN enhance performance in tasks like image classification and machine translation.
8. Adaptive learning rate algorithms (e.g., Adam, Adagrad, RMSProp, Adadelta) improve convergence by adjusting learning rates based on gradient magnitudes and second-order information. Gradient clipping prevents numerical instabilities due to large partial derivatives.
9. Momentum and Nesterov momentum techniques, using partial derivatives, reduce optimization oscillations for faster convergence.
10. Partial derivatives enhance neural network performance, leading to more accurate predictions and improved models.

7.2. Implications for future research Future research avenues include:

1. Examining activation functions' impact on partial derivative computation efficiency and neural network performance.
2. Developing optimization algorithms addressing high-dimensional optimization challenges.
3. Tackling the vanishing gradient problem in deep networks with more effective partial derivative computation techniques.
4. Assessing regularization techniques' benefits for neural network generalization performance.
5. Investigating normalization techniques' influence on partial derivative computation efficiency and neural network performance.

This paper summarized partial derivatives' application in machine learning, focusing on backpropagation and neural network training. Understanding the challenges and opportunities with partial derivatives can help develop more efficient and effective neural network training and optimization techniques.

References

[1] Goodfellow, Ian, et al. Deep learning. Vol. 1. Cambridge: MIT press, 2016

[2] Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors." nature 323.6088 (1986): 533-536.

[3] Bishop, Christopher M. *Neural networks for pattern recognition*. Oxford university press, 1995.

[4] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." nature 521.7553 (2015): 436-444

[5] Ruder, Sebastian. "An overview of gradient descent optimization algorithms." arXiv preprint arXiv:1609.04747 (2016)

[6] Nielsen, Michael A. Neural networks and deep learning. Vol. 25. San Francisco, CA, USA: Determination press, 2015

[7] Stewart, James, Daniel K. Clegg, and Saleem Watson. *Calculus: early transcendentals*. Cengage Learning, 2020.

[8] Hastie, Trevor, et al. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. New York: springer, 2009

[9] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." *The journal of machine learning research* 15.1 (2014): 1929-1958.

[10] Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010.

[11] Bengio, Yoshua, Ian J. Goodfellow, and Aaron Courville. "Deep learning." Nature 521.7553 (2015): 436-444.

[12] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." International conference on machine learning. pmlr, 2015.

[13] Bishop, Christopher M., and Nasser M. Nasrabadi. Pattern recognition and machine learning. Vol. 4. No. 4. New York: springer, 2006.

[14] LeCun, Y. A., et al. "Efficient BackProp. G. Montavon et al.(Eds.): NN: Tricks of the Trade, 2nd edn., LNCS 7700." (2012): 9-48.

[15] Glorot, Xavier, Antoine Bordes, and Yoshua Bengio. "Deep sparse rectifier neural networks." Proceedings of the fourteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings, 2011.

[16] Maas, Andrew L., Awni Y. Hannun, and Andrew Y. Ng. "Rectifier nonlinearities improve neural network acoustic models." Proc. icml. Vol. 30. No. 1. 2013.

[17] Liu, Ziwei, et al. "Deep learning face attributes in the wild." Proceedings of the IEEE international conference on computer vision. 2015.

[18] Zaheer, Raniah, and Humera Shaziya. "GPU-based empirical evaluation of activation functions in convolutional neural networks." 2018 2nd International Conference on Inventive Systems and Control (ICISC). IEEE, 2018.

[19] Gal, Yarin, Mark Van Der Wilk, and Carl Edward Rasmussen. "Distributed variational inference in sparse Gaussian process regression and latent variable models." Advances in neural information processing systems 27 (2014).

[20] Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton. "Layer normalization." arXiv preprint arXiv:1607.06450 (2016)

[21] Wu, Yonghui, et al. "Google's neural machine translation system: Bridging the gap between human and machine translation." arXiv preprint arXiv:1609.08144 (2016)

[22] Huang, Gao, et al. "Densely connected convolutional networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[23] Iandola, Forrest N., et al. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size." arXiv preprint arXiv:1602.07360 (2016).

[24] Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio. "On the difficulty of training recurrent neural networks." International conference on machine learning. Pmlr, 2013.

[25] Keskar, Nitish Shirish, et al. "On large-batch training for deep learning: Generalization gap and sharp minima." arXiv preprint arXiv:1609.04836 (2016).

[26] Hinton, Geoffrey, Nitish Srivastava, and Kevin Swersky. "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent." Cited on 14.8 (2012): 2.

[27] Wang, Xiaolong, et al. "Non-local neural networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

[28] Sutskever, Ilya, et al. "On the importance of initialization and momentum in deep learning." International conference on machine learning. PMLR, 2013.

[29] Zhang, Chiyuan, et al. "Understanding deep learning (still) requires rethinking generalization." Communications of the ACM 64.3 (2021): 107-115.

[30] Zeiler, Matthew D. "Adadelta: an adaptive learning rate method." arXiv preprint arXiv:1212.5701 (2012).

[31] Tieleman, Tijmen, and Geoffrey Hinton. "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude." COURSERA: Neural networks for machine learning 4.2 (2012)

[32] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014).

[33] Duchi, John, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization." Journal of machine learning research 12.7 (2011)

[34] Reddi, Sashank J., Satyen Kale, and Sanjiv Kumar. "On the convergence of adam and beyond." arXiv preprint arXiv:1904.09237 (2019).

[35] Nesterov, Yurii. "A method of solving a convex programming problem with convergence rate O (1/k2)." Soviet mathematics doklady 27.2 (1983): 372-376.

[36] Qian, Ning. "On the momentum term in gradient descent learning algorithms." Neural networks 12.1 (1999): 145-151.

[37] Polyak, Boris T. "Some methods of speeding up the convergence of iteration methods." *Ussr computational mathematics and mathematical physics* 4.5 (1964): 1-17.

[38] Nesterov, Yurii. Introductory lectures on convex optimization: A basic course. Vol. 87. Springer Science & Business Media, 2003.

[39] Dozat, Timothy. "Incorporating nesterov momentum into adam." (2016).

[40] Hinton, Geoffrey E., et al. "Improving neural networks by preventing co-adaptation of feature detectors." arXiv preprint arXiv:1207.0580 (2012).

[41] Morgan, Nelson, and Hervé Bourlard. "Generalization and parameter estimation in feedforward nets: Some experiments." Advances in neural information processing systems 2 (1989).

[42] David Kriesel, A Brief Introduction to Neural Networks, Fundamentals on learning and training samples, published online, Germany, 2005.

[43] Gonzalez, Rafael C., and Richard E. Woods. Processamento de imagens digitais. Editora Blucher, 2000.

[44] Chen, Yahui. Convolutional neural network for sentence classification. MS thesis. University of Waterloo, 2015.

[45] Huang, James Y., Kuan-Hao Huang, and Kai-Wei Chang. "Disentangling semantics and syntax in sentence embeddings with pre-trained language models." arXiv preprint arXiv:2104.05115 (2021).

[46] Zhang, Ye, and Byron Wallace. "A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification." arXiv preprint arXiv:1510.03820 (2015).

[47] Veselý, Karel, et al. "Sequence-discriminative training of deep neural networks." *Interspeech*. Vol. 2013. 2013.

[48] Graves, Alex, et al. "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks." Proceedings of the 23rd international conference on Machine learning. 2006.

[49] Chan, William, et al. "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition." 2016 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, 2016.

[50] Goswami, Somdatta, et al. "Deep transfer learning for partial differential equations under conditional shift with DeepONet." arXiv preprint arXiv:2204.09810 (2022).

[51] Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." nature 518.7540 (2015): 529-533.

[52] Peters, Jan, Sethu Vijayakumar, and Stefan Schaal. "Reinforcement learning for humanoid robotics." Proceedings of the third IEEE-RAS international conference on humanoid robots. 2003.

[53] Gu, Shixiang, et al. "Continuous deep q-learning with model-based acceleration." International conference on machine learning. PMLR, 2016.

[54] Bagnell, J. Andrew, and Jeff G. Schneider. "Autonomous helicopter control using reinforcement learning policy search methods." Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164). Vol. 2. IEEE, 2001.

[55] Kirsch, Louis, Sjoerd van Steenkiste, and Jürgen Schmidhuber. "Improving generalization in meta reinforcement learning using learned objectives." arXiv preprint arXiv:1910.04098 (2019).

[56] Cai, Chen, and Yusu Wang. "A simple yet effective baseline for non-attributed graph classification." arXiv preprint arXiv:1811.03508 (2018).

[57] Raissi, Maziar, Paris Perdikaris, and George Em Karniadakis. "Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations." arXiv preprint arXiv:1711.10561 (2017).

[58] Schulman, John, et al. "Trust region policy optimization." *International conference on machine learning*. PMLR, 2015.

[59] Schulman, John, et al. "High-dimensional continuous control using generalized advantage estimation." *arXiv preprint arXiv:1506.02438* (2015).

[60] Silver, David, et al. "Mastering the game of Go with deep neural networks and tree search." nature 529.7587 (2016): 484-489.

[61] Bengio, Yoshua, Nicholas Léonard, and Aaron Courville. "Estimating or propagating gradients through stochastic neurons for conditional computation." *arXiv preprint arXiv:1308.3432* (2013).

[62] Loshchilov, Ilya, and Frank Hutter. "Sgdr: Stochastic gradient descent with warm restarts." *arXiv preprint arXiv:1608.03983* (2016).

[63] McClarren, Ryan G., and Ryan G. McClarren. "Reinforcement Learning with Policy Gradients." *Machine Learning for Engineers: Using data to solve problems for physical systems* (2021): 219-237.

[64] Goodfellow, Ian, et al. "Generative adversarial networks." *Communications of the ACM* 63.11 (2020): 139-144.