Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 7, No. 1, 2024

Entropy Error-based Convolutional Neural Network Sparse Optimization and Application

Kaifang Yang *, Dewei Yang, Fen Wang, Jing Zhao, Qinwei Fan

Xi'an Polytechnic University, Xi'an Shaanxi, 710600, China

*** Corresponding author:** Kaifang Yang

Abstract: Convolutional Neural Networks (CNNs), as quintessential representatives of deep neural networks, have found widespread applications in numerous domains such as image recognition, object detection, and image generation, owing to their robust nonlinear mapping capabilities. However, the practical deployment of CNNs faces challenges such as low learning efficiency and subpar accuracy due to the intricacies in the network structure. This paper introduces the smoothing L1 regularization term atop the conventional entropy loss function, effectively enhancing both the learning efficiency and algorithmic precision of Convolutional Neural Networks. The efficacy of the improved algorithm is substantiated through numerical simulations.

Keywords: Convolution Neural Network (CNN); Smooth L1 Regularization; Cross-Entropy Loss Function; Classification Problems.

1. Introduction:

The introduction of Convolutional Neural Networks (CNNs) signifies a significant advancement in the field of deep learning. In 1998, LeCun and collaborators pioneered CNNs, applying them in the form of LeNet-5 to handwritten digit recognition tasks, thus propelling the development of automatic classification in digital image research. Subsequently, scholars widely applied various forms of CNNs across diverse fields such as computer vision, medical imaging, natural language processing, and video analysis, showcasing their extensive application prospects [1-6].

As models become more complex, they may merely memorize noise and minor variations in training data, resulting in redundant information and diminished generalization capabilities within the network. Regularization techniques become crucial to address this issue. Despite attempts by researchers to optimize neural networks for sparse solutions through L1 regularization, introducing L1 regularization terms may lead to discontinuities in the loss function. Therefore, we introduce the technique of smoothing L1 regularization, considering the sum of absolute values in the loss function. By enhancing smoothness, this effectively reduces model complexity, mitigates overfitting, and improves generalization performance through the pruning of nodes approaching zero.

To further explore the potential of Convolutional Neural Networks, this paper employs an innovative CNN architecture that integrates smoothing L1 regularization and cross-entropy loss function [7-8], fully leveraging the outstanding performance of CNNs in classification tasks.

In the subsequent section of this paper, the second part will introduce the structure and working principles of Convolutional Neural Networks (CNNs) [9]. It will provide a detailed exposition of the mathematical computation process of the SL1 regularization term in the convolutional network, showcasing the model construction process. Finally, the third part will describe and analyze the experimental results to validate the effectiveness of the algorithm.

2. Related Work:

(1) Convolutional Neural Network (CNN) Structure and Operating Principles:

The core architecture of Convolutional Neural Networks (CNNs) encompasses convolutional layers, pooling layers, and fully connected layers. Convolutional layers extract pivotal features through convolutional operations and activation functions, while pooling layers reduce computational complexity by downsizing feature map dimensions. By alternately stacking these layers, features are extracted from input data, mapped to outputs through fully connected layers, and a probability distribution is obtained using the SoftMax activation function. The loss calculation phase employs a cross-entropy loss function to compare network outputs with actual labels, yielding the loss value.

In the backpropagation phase, gradients of layer weights are computed using the chain rule [10] and subsequently updated. To expedite backpropagation, a momentum term is introduced as an optimization method, aiming to enhance the convergence speed of gradient descent and improve the stability of model training. Through iterative processes, network parameters are gradually adjusted to enhance model performance, The convolutional network structure we designed is illustrated in the accompanying diagram.

In CNN, we process input images of size 28×28. The images undergo convolutional operations with a 5×5 convolutional kernel, followed by non-linear processing through the sigmoid activation function. The resulting feature maps are then subjected to average pooling, reducing the image dimensions to 12×12. The images then pass through a second convolutional layer with the same pooling operation as the first layer. This step involves convolution and pooling processes again to extract features. The processed results are flattened into a 16×1 vector through a fully connected layer. The vectors are concatenated to form a 192×1 vector, aiding in the fusion of features extracted from different convolutional and pooling layers. Finally, by applying the SoftMax activation function, this vector is mapped to a

category probability distribution for label-based classification.

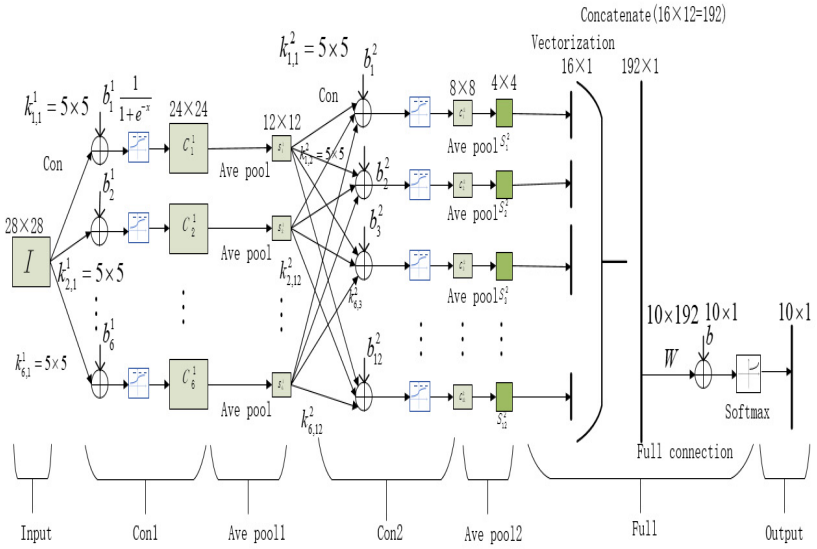

Figure 1. Convolutional Neural Network

(2) Mathematical computations

Convolution operation is a crucial step in feature extraction in deep learning, especially in the context of image processing. For the convolution operation, the calculation formula between the input matrix X and the convolutional kernel K is:

$$A(i,j) = \sum_{m}\sum_{n} X(i+m, j+n)\cdot K(m,n), \quad (1)$$

Given the convolved output matrix A and the size of the pooling window k×k, the computation process of the output matrix is as follows:

$$O(i,j) = \frac{1}{k^2}\sum_{m=0}^{k-1}\sum_{n=0}^{k-1} A(i\cdot k+m, j\cdot k+n), \quad (2)$$

In (1) and (2), denotes the element in the i-th row and j-th column of the input matrix, representing the i-th row and j-th column element in the convolutional kernel K. The calculation formula for the fully connected layer is the same as that of a conventional fully connected layer. For a fully connected layer, given the output of the previous layer, weight matrix, and bias vector, the calculation process for the output is as follows:

$$S = \mathrm{Re}LU(O\times U + b), \quad (3)$$

Where represents matrix multiplication, O is the output vector from the previous layer, U is the weight matrix of the fully connected layer, and b is the bias vector. ReLU is a commonly used activation function defined as $\mathrm{Re}LU(x) = \max(0,x)$ .The introduction of this non-linear activation function enables the network to better learn complex feature mappings. The output S of the fully connected layer can be utilized for classification tasks, and based on the entropy error function, the error function of the convolutional neural network is defined as:

$$E = -\sum_{j=1}^{J}[\xi \ln\frac{y^j}{\xi^j} + (1-\xi^j)\ln\frac{1-y^j}{1-\xi^j}]. \quad (4)$$

(3) Smoothed L1 Regularization Term

Smooth approximation is a strategy that substitutes the absolute value function with a continuously differentiable function, aiding in better support for the optimization process and enhancing the convergence and generalization of the model during training. Given a training sample $\{X^j, O^j\}_{j=1}^{J}$, where O^j represents the ideal output, we define the following smooth function:

$$h(u_k,\alpha) = \begin{cases} \|u_k\|, & \|u_k\| \ge \alpha \\ \frac{\|u_k\|^2}{2\alpha} + \frac{\alpha}{2}, & \|u_k\| < \alpha, \end{cases} \quad (5)$$

Where $\alpha > 0$, the weight vector u_k represents the vector connecting to the k-th node of the fully connected layer, and the k-th node connecting to all output nodes in the fully connected layer. Subsequently, the h (u_k, α) gradient of and the gradient of the vector u_k are as follows:

$$\nabla_{u_k} h(u_k,\alpha) = \begin{cases} \frac{u_k}{\|u_k\|}, & \|u_k\| \ge \alpha \\ \frac{u_k}{\alpha}, & \|u_k\| < \alpha, \end{cases} \quad (6)$$

The error function of the convolutional neural network based on L1 smooth regularization of the entropy error function, as indicated by the above formula, is defined as follows:

$$E(w) = -\sum_{j=1}^{J}\sum_{i=1}^{10}[\xi^j \ln\frac{g(U_i\cdot S^j)}{\xi^j} + (1-\xi^j)\ln\frac{1-g(U_i\cdot S^j)}{1-\xi^j}] + \lambda\sum_{j=1}^{J}\sum_{k=1}^{q} h(u_k\cdot x^j). \quad (7)$$

3. Numerical Experiments

(1) Dataset Introduction

The dataset is a classic collection of handwritten digit images, used for machine learning training and testing. It comprises 60,000 training images and 10,000 28×28-pixel test images. Through training our model, we can achieve accurate recognition of handwritten digits.

(2) InnParameter Configuration

In our convolutional neural network, the convolutional layer extracts feature information from the input image by performing element-wise multiplication and summation operations using a 5×5 convolutional kernel. To maintain the spatial dimensions of the output, we have chosen a stride of 1

and zero-padding. The pooling layer employs 2×2 average pooling to reduce data dimensions and enhance the model's robustness. The Softmax activation function is used to transform the neural network output into a probability distribution. During the parameter update phase, we introduce a momentum parameter set to 0.2, which aids in accelerating convergence, especially in the early stages of training. Interestingly, in the initial 20 iterations, we gradually increase the momentum value to better explore local minima of the loss function and then rapidly converge towards the global minimum. The experimental parameter settings are presented in the table below.

Table 1. Model Parameter Settings

size	Operations
28×28×6	Convolutional Stride = 1, Padding = 0
24×24×6	2×2 Average Pooling with ReLU Activation Function
12×12×12	Convolution Stride = 1, Padding = 0
8×8×12	2×2 Average Pooling with ReLU Activation Function
16×1	Fully Connected
192×1	Fully Connected
10×192	Fully Connected

(3) Results Analysis

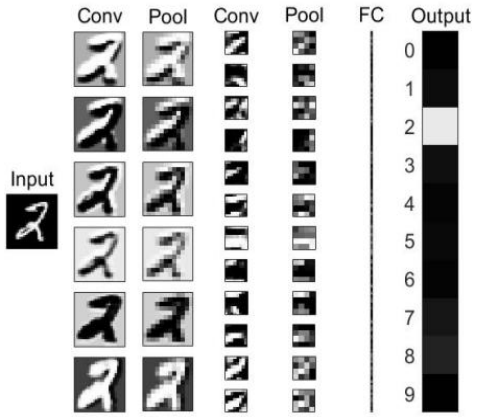

Figure 2. Network Visualization

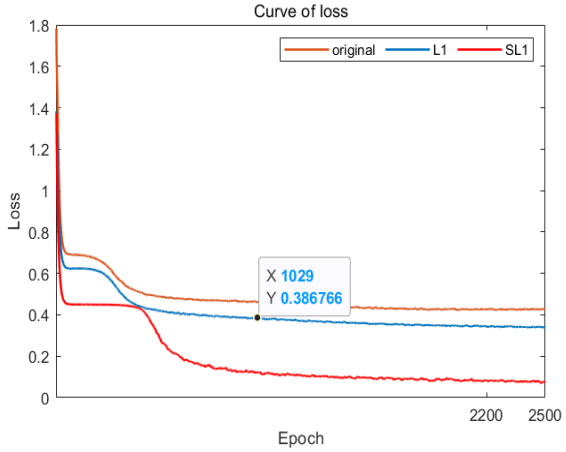

Figure 3. Loss Function Plot

In Figure 2, we present a detailed visualization of handwritten digit recognition. Figure 3 further compares the error functions of different algorithms, with a particular focus on the outstanding performance of the SL1 algorithm in enhancing network performance. The figure illustrates the rapid reduction of network error during the early stages of training using the SL1 algorithm, ultimately stabilizing and exhibiting a gradual convergence trend, while other algorithms show relatively weaker performance. This clearly indicates a significant advantage of the SL1 algorithm over others in terms of error reduction speed and convergence. Through the incorporation of smooth L1 regularization and entropy error functions, the SL1 algorithm achieves higher precision during the model inference phase, successfully enhancing the accuracy of handwritten digit recognition.

4. Conclusion

This study has achieved success in optimizing Convolutional Neural Networks (CNNs) by introducing a smoothed L1 regularization term and an entropy error function. The smoothed L1 regularization effectively prunes relevant nodes, significantly enhancing the network's computational efficiency and generalization performance. In comparison to traditional regularization methods, it provides a more robust mathematical foundation for neural network training. Simultaneously, the application of the entropy error function enhances the assessment of differences between model outputs and label distributions. A series of experiments demonstrate that our proposed algorithm effectively balances algorithm development and exploration. Results from classification tests on the MNIST dataset indicate that the algorithm exhibits good generalization performance and stability. This suggests that the optimization approach, considering both the smoothed L1 regularization term and entropy error function, is successful and viable for improving CNN performance.

References

[1] Leon. CNN: A Vision of Complexity, International Journal of Bifurcation and Chaos, 7 (1997) :2219-2425.

[2] B. T. Li. Learning deep neural networks for node classification, Expert systems with applications, 137(2019): 324-334.

[3] F. Ghasemi. Deep neural network in qsar studies using deep belief network, Applied Softcomputing, 62 (2018): 251-258.

[4] Y. S. Xu. Convergence of deep convolutional neural networks, Neural Networks, 153 (2022): 553-563.

[5] S. F. Wen, Translation analysis of English address image recognition based on image recognition, image and video processing, 11 (2019): 1-9.

[6] M.Y.Park, T.Hastie, L1-Regularization Path Algorithm for Generalized Linear Model, Journal of the Royal Statistical Society Series B: Statistical Methodology, 69 (2007): 659–677.

[7] C.Yang. Structured pruning of convolutional neural networks via L1 Regularization, IEEE Access, 7 (2019): 106385-106394.

[8] L. Li. Approximating the Gradient of Cross-Entropy Loss Function, IEEE Access, 2020 (8): 111626-111635.

[9] T.Wiatowski.A mathematical theory of deep convolutional neural networks for feature extraction, IEEE Transactions On Information Theory, 64(2017): 1845-1866.

[10] P.Y.Wang. Differentially private sgd with non-smooth losses, Applied and Computational Harmonic Analysis, 56 (2022): 306-336.