Applications of Deep Learning in Image Classification

Authors: Li Aosheng, Li Aosheng

Date: 2024-01-06T00:00:00+00:00

Abstract

The vigorous development of deep learning technology has brought revolutionary changes to the computer vision domain through its widespread application in image classification. This paper focuses on the application of deep learning in image classification, based on the CIFAR-10 dataset, constructing different models via convolutional neural networks (CNN) to evaluate classification accuracy. We also employed renowned models such as ResNet to assess their classification performance on the CIFAR-10 dataset. To thoroughly investigate model characteristics, we introduced different activation functions in our experiments, and by comparing their performance in classification tasks, revealed the impact of activation functions on model performance. Furthermore, by adjusting hyperparameters such as the number of training epochs, we systematically investigated the classification accuracy of models under different training epochs. In the experiments, we found that increasing the number of training epochs is not always beneficial, as the phenomenon of overfitting may occur. However, by adjusting the model and adding Dropout layers, we mitigated the overfitting problem to a certain extent. Simultaneously, through optimization and adjustment of the model, we achieved a significant improvement in classification accuracy, reaching a maximum of 0.7509. Through the continuous optimization process, we have become increasingly confident in the prospects of deep learning for image classification, and we will continue to conduct in-depth research in this direction.

Full Text

Preamble

Beijing University of Chemical Technology
Technical Report (Thesis)

Title: Application of Deep Learning in Image Classification
Major: Information and Communication Engineering

Student: Li Aosheng
Class: Xinyan 2308
Student ID: 2023200842

December 27, 2023

Abstract

The rapid development of deep learning technology has brought revolutionary advances to computer vision through its widespread application in image classification. This paper focuses on deep learning applications for image classification, using the CIFAR-10 dataset as a foundation to evaluate classification accuracy through various convolutional neural network (CNN) architectures. We also assess the performance of renowned models such as ResNet on CIFAR-10. To thoroughly investigate model characteristics, we introduce different activation functions in our experiments and reveal their impact on model performance through comparative analysis. Additionally, by adjusting hyperparameters such as training epochs, we systematically examine classification accuracy across different training durations. Our experiments demonstrate that more training epochs do not necessarily yield better results, as overfitting may occur. We address this issue through model adjustments and the addition of Dropout layers, substantially improving classification accuracy to a peak of 0.7509. Through continuous optimization, we have gained increasing confidence in the prospects of deep learning for image classification and will pursue further research in this direction.

Keywords: Deep learning, Image classification, CNN, ResNet

Chapter 1: Introduction

1.1 Research Background and Significance

Image classification has long been a fundamental problem in computer vision, attracting widespread attention. In recent years, the rise of deep learning technology, particularly the successful application of convolutional neural networks (CNNs), has yielded remarkable progress in both classification accuracy and efficiency.

Before the deep learning era, image classification primarily relied on handcrafted feature extraction combined with traditional machine learning methods such as Support Vector Machines (SVM) and random forests. These approaches often suffered from limitations in feature design and generalization capabilities when handling complex image data, making them ill-suited for large-scale and multi-category classification tasks.

The emergence of CNNs has accelerated image classification development, establishing them as essential tools for the problem. Through multi-layered convolutional and pooling operations, CNNs automatically learn hierarchical feature representations that capture semantic information at different levels. This end-to-end learning paradigm enables models to better understand image structure and content. The multi-layer architecture facilitates hierarchical feature learning: lower convolutional layers capture low-level edges and textures, while deeper layers learn more abstract and semantic features, thereby enhancing classification accuracy.

The availability of large-scale image datasets such as ImageNet has further propelled image classification research. These datasets provide millions of richly annotated images, offering deep learning models abundant training samples to learn more generalizable feature representations. Pre-trained models including VGG, ResNet, and EfficientNet can be fine-tuned for specific image classification tasks through transfer learning, achieving significant performance improvements even on relatively small datasets and enhancing model generalization.

As deep learning technology evolves, image classification research has gradually expanded to multi-modal learning, simultaneously processing images, text, and other modalities. This expansion broadens the scope of image classification tasks, enabling models to more comprehensively understand and process information from multiple sources.

Deep learning has achieved remarkable success in image classification, improving not only classification accuracy but also expanding research horizons. As the technology continues to develop, the field will face new challenges and opportunities.

1.2 CNN Applications in Image Classification

With the exponential growth of digital images, traditional classification methods face challenges when handling complex scenes and large-scale datasets. Convolutional Neural Networks (CNNs) have achieved tremendous success in image classification, with applications spanning numerous domains. Key aspects include:

Feature Learning: CNNs learn image features through convolutional and pooling layers. Convolutional layers apply filters to capture local visual features, while pooling layers reduce spatial resolution while preserving essential information. This hierarchical feature learning helps networks understand image structure and content.

Hierarchical Representation: The multi-layer architecture enables gradual learning of increasingly abstract features. Lower layers capture edges and textures, while deeper layers learn semantic features, facilitating more accurate classification.

Local Receptive Fields: CNNs employ local receptive fields where each neu-

ron only attends to a small region of the input image rather than the entire image. This design provides invariance to transformations such as translation, rotation, and scaling, improving model generalization.

Parameter Sharing: The parameter sharing mechanism in CNNs reduces the total number of parameters, decreasing training data requirements and mitigating overfitting risks. Shared convolutional kernels can detect similar features across the entire image, improving statistical efficiency.

Global Average Pooling: The final layer typically uses global average pooling to reduce the entire feature map to a scalar, followed by a fully connected layer for classification. This pooling method better preserves global image information, helping the network classify based on overall features.

Transfer Learning: Pre-trained CNN models learn rich feature representations from large-scale image data, which can be transferred to specific classification tasks. This approach enables good performance even on relatively small datasets.

CNN applications in image classification have not only improved accuracy but also achieved outstanding results in numerous computer vision applications, including face recognition, object detection, and medical image analysis. Due to their powerful feature learning capabilities, CNNs have become a mainstream model for image classification tasks.

1.3 Main Research Work and Chapter Arrangement

This thesis is organized into four chapters, with the following structure:

Chapter 2 introduces the CIFAR-10 dataset, covering its origin, image quantity, and common applications.

Chapter 3 presents CNN concepts, describes the representative ResNet model, and provides code implementation details. We evaluate and analyze classification accuracy using different training epochs, activation functions, and model architectures.

Chapter 4 concludes the paper, summarizing our research contributions and identifying directions for future investigation.

Chapter 2: Dataset Introduction

2.1 CIFAR-10

CIFAR-10 (Canadian Institute for Advanced Research - 10) is a classic image classification dataset created by the Canadian Institute for Advanced Research in 2004. Designed to advance machine learning and computer vision research, it provides a standardized benchmark for algorithms and models. CIFAR-10

has become one of the most widely used datasets in the computer vision and machine learning communities. Researchers commonly employ CIFAR-10 to develop and evaluate CNNs and other deep learning models due to its moderate size and multiple categories, serving as a standard benchmark for assessing image classification algorithm capabilities.

2.2 Dataset Composition

The CIFAR-10 dataset comprises 60,000 color images of size 32$×$32 pixels, divided into 10 distinct categories with 6,000 images per class. The ten categories are: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Sample images are shown in Figure 2 [Figure 2: see original paper]-1.

The dataset is partitioned into training and test sets, with 50,000 images for training and 10,000 for testing. Each image is labeled with one of the ten categories, making it suitable for supervised learning.

2.3 Common Applications

CIFAR-10 is primarily used to evaluate and compare the performance of machine learning algorithms and models on image classification tasks. Researchers and developers frequently use this dataset to test new image classification algorithms, particularly deep learning models such as CNNs.

The dataset presents a challenging task for deep learning research due to its relatively low image resolution and the fact that objects may appear in various positions and poses, increasing classification difficulty. Researchers leverage CIFAR-10 to explore different neural network architectures, loss functions, and optimization methods to improve model performance on image classification tasks.

Additionally, CIFAR-10 is widely used in transfer learning research. Due to its relatively small size and diversity, researchers can pre-train models on CIFAR-10 and then transfer them to larger, more complex image datasets to improve model generalization.

Overall, the CIFAR-10 dataset plays a crucial role in machine learning, providing a common benchmark that drives research and development in image classification.

Chapter 3: PyTorch-Based Image Classification Research

3.1.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are network structures designed for feature extraction from data that can be represented as multi-dimensional grids,

such as images and videos. Characterized by sparse connections and parameter sharing, CNNs have achieved superior results in various computer vision tasks including detection and classification. A complete CNN network can be constructed using combinations of convolutional layers, pooling layers, and activation functions to extract features from input data, followed by fully connected layers and objective functions to accomplish specific vision tasks. CNN parameters are updated using the backpropagation algorithm, which propagates computed residuals layer by layer to update network parameters until convergence.

Convolutional Layers: As the core component of CNNs, convolutional layers extract local features from images. Unlike matrix multiplication in traditional neural networks, convolution employs sparse connections and local interactions through convolutional kernels. Since adjacent pixels in image data exhibit stronger correlations while distant pixels show weaker correlations, the sparse connectivity of convolution allows effective feature extraction by focusing only on pixel neighborhoods, requiring fewer stored parameters and offering higher computational efficiency.

Pooling Layers: After convolutional operations and activation, feature maps often become very large. Pooling layers adjust the scale of output feature maps by using summary statistics of neighboring features at each location to represent the output. This reduces the spatial dimensions while maintaining the number of channels, decreasing model parameters and improving overall training efficiency. Common pooling methods include max pooling and average pooling.

Fully Connected Layers: Fully connected layers map features extracted by convolutional layers to final output categories. In these layers, each neuron connects to all neurons in the previous layer, learning weights and biases to complete feature combination and classification.

3.1.2 Typical Image Classification Algorithms

ResNet, proposed by Kaiming He et al., is a renowned convolutional neural network that successfully mitigates the vanishing gradient problem through residual blocks and skip connections. Residual blocks allow information to pass directly to subsequent layers, enabling the network to learn identity mappings more easily. This design facilitates smoother gradient propagation during deep network training and simplifies optimization. Skip connections allow information to bypass intermediate layers directly, effectively alleviating gradient degradation and enabling training of networks with hundreds or even thousands of layers. This architecture has enabled ResNet to successfully train very deep neural networks, achieving significant performance improvements on image classification and other tasks.

3.1.3 Activation Functions

Activation functions introduce non-linearity into deep networks. Without them, input-output relationships could be represented by linear combinations regard-

less of network depth, eliminating the distinction between deep networks and original perceptrons. We now discuss common activation functions including Sigmoid, ReLU, and the recently prominent Swish function for image tasks.

ReLU Activation Function: Proposed by G. Hinton et al., ReLU remains a widely used piecewise activation function with the formula:

$$ReLU(x) = \max(0, x)$$

For input x, ReLU outputs x if x is positive and zero otherwise. ReLU mitigates the vanishing gradient problem and offers fast computation, requiring only a simple comparison with zero and faster convergence than Sigmoid. However, it suffers from "dead neurons" that may never activate during training, potentially halting gradient updates. Common solutions include Leaky ReLU and other variants. Overall, ReLU is a simple yet effective activation function that has achieved excellent results in deep learning, though other variants may be more suitable depending on task requirements and network architecture.

Leaky ReLU: Leaky ReLU is an improved version that addresses the dead neuron problem by allowing a small slope for negative values, providing non-zero outputs for negative inputs:

$$Leaky\ ReLU(x) = \max(\alpha x, x)$$

where α is a small positive number, typically set to a small value like 0.01. This slight slope introduces a non-zero gradient for negative values, facilitating better gradient propagation during training and preventing neurons from remaining permanently inactive. While Leaky ReLU alleviates the dead neuron problem, some neurons may still become inactive, in which case alternatives like Parametric ReLU (PReLU) may be preferable. In practice, Leaky ReLU is commonly used in hidden layers of deep learning models, though the specific choice depends on dataset and task characteristics.

Swish Activation Function: Proposed by Google Brain in 2017, Swish is defined as:

$$f(x) = x \cdot Sigmoid(x)$$

This simple function multiplies the input by its Sigmoid value, offering smoothness, non-monotonicity, and a lower bound without an upper bound. Swish has demonstrated significant performance improvements over ReLU in image classification and detection tasks.

3.2.1 Data Processing

We used the CIFAR-10 dataset downloaded from https://www.cs.toronto.edu/~kriz/cifar.html. The extracted file format is shown in Figure 3 [Figure 6: see original paper]. We processed the data using official methods and additional code to output images with their corresponding labels. This processing was implemented in cifar_{10}.ipynb, resulting in images organized into training and test folders named train_{cifar10} and test_{cifar10}, with labels stored as trainLabels.csv and testLables.csv. Sample test images and labels are shown in Figures 3 [Figure 7: see original paper] and 3 [Figure 8: see original paper]. The processed images and labels were placed in the temp folder within imageclassification for training and testing.

3.2.2 Image Classification Model

Our implementation is contained in ImageNet Classification.ipynb. The model is a CNN where input data passes through the first convolutional layer (self.conv1), followed by ReLU activation and max pooling (self.pool) for downsampling. Feature maps then proceed through the second convolutional layer (self.conv2), another ReLU activation, and a final max pooling layer. The pooled feature maps are flattened into a one-dimensional vector via x.view(-1, 1655). This vector passes through two fully connected layers (self.fc1 and self.fc2) with ReLU activation, culminating in the output layer (self.fc3) producing class scores. The model architecture is illustrated in Figure 3 [Figure 9: see original paper].

3.2.3 Training and Evaluation

We split the 50,000 training images into 40,000 for training and 10,000 for validation to assess training effectiveness. During training, we iterate through mini-batches from the training set, compute predictions via forward propagation, and measure differences from true labels using cross-entropy loss. Backpropagation adjusts model parameters through gradient descent optimization to minimize loss. We accumulate loss values per mini-batch and periodically print training progress. Finally, we record and output the loss for the last mini-batch.

All test set images were predicted in batches, with results saved to submission.csv. We compared these predictions with testLables.csv to calculate classification accuracy. Figure 3 [Figure 6: see original paper] shows classification results for 12 randomly selected images using a model trained for 60 epochs, with 8 out of 12 predictions correct, demonstrating acceptable performance.

Classification accuracy varies with training duration, as shown in Table 3 -1. Accuracy peaks at 30 epochs but does not consistently improve with additional training, suggesting overfitting at higher epoch counts—a concern addressed in subsequent model improvements.

3.2.4 Activation Function Improvements

We evaluated classification accuracy using different activation functions, specifically Leaky ReLU and Swish. For Leaky ReLU, α was set to 0.01. Experiments using these activation functions were implemented in ImageNet Classification_{Leak}.ipynb and ImageNet Classification_{Swish}.ipynb, yielding accuracy values across different training epochs as shown in Table 3 -2.

Leaky ReLU demonstrates strong performance even with limited training, achieving 0.61 accuracy after only 10 epochs. Overall classification accuracy improves compared to standard ReLU. However, the pattern of decreasing performance with increased epochs persists across all activation functions, indicating that the issue lies in the model architecture itself, prompting further modifications.

3.2.5 Model Improvements

To address performance degradation at higher training epochs, we modified the model by adding Dropout layers with a 0.5 probability for random neuron deactivation to prevent overfitting. The adjusted implementation is contained in Improve.ipynb, with the modified architecture shown in Figure 3 [Figure 7: see original paper].

The improved model structure consists of: (1) Convolutional Layer 1 extracts 32 feature maps using 3$\times$3$kernels$, $followedbyReLUactivationandpooling$; (2)$ConvolutionalLayer2extracts64f$ kernels, followed by ReLU activation and pooling; (4) Fully Connected Layer 1 flattens feature maps into a 512-dimensional feature space with ReLU activation; (5) Fully Connected Layer 2 maps from 512 to 256 dimensions with ReLU activation; (6) Fully Connected Layer 3 maps from 256 dimensions to 10 output classes. The key modification is the addition of Dropout layers between fully connected layers with 0.5 probability. The final output layer uses no activation function, as softmax is typically integrated into the loss function for multi-class classification.

Classification results for the improved model are presented in Table 3 -3. The enhanced model achieves significantly higher accuracy, reaching 0.7509 after 100 training epochs while resolving overfitting issues. Due to computational constraints, we did not increase training further, though accuracy shows continued improvement potential.

3.2.6 ResNet Model

We also evaluated classification performance using ResNet, implemented in ResNet.ipynb. Due to hardware limitations, we trained for only 10 epochs, achieving accuracy of 0.7088 as shown in Table 3 -4. This represents substantial improvement over the original model and a 0.0232 improvement over the improved model.

3.3 Chapter Summary

Through iterative optimization experiments using different activation functions and training epochs, we progressively improved image classification performance. Our best-performing model, Improve, trained for 100 epochs, achieved 0.7509 accuracy with continued improvement potential. Computational limitations prevented further training, but future work with better hardware and GPU acceleration could yield additional gains.

All code used in this paper is available at: https://github.com/LAS666/DeepLearingCode.git

We began with a simple CNN that exhibited overfitting at higher training epochs. Changing activation functions did not resolve this issue. By modifying the architecture to include Dropout layers and adjusting the network structure, we achieved progressively better performance with increased training epochs, mitigating overfitting while significantly improving accuracy to 0.7509 after 100 epochs. These experiments deepened our understanding of deep learning-based image classification and revealed its promising future. We anticipate continued emergence of novel architectures and models that will further enhance classification performance, and we remain committed to advancing research in this direction.

Note: Figure translations are in progress. See original paper for figures.