WEP Warwick Evans Publishing

International Journal of Computer Science and Information Technology

ISSN: 3005-9682 (Print), ISSN: 3005-7140 (Online) | Volume 5, Number 1, Year 2025
DOI: https://doi.org/10.62051/ijcsit.v5n1.05
Journal homepage: https://wepub.org/index.php/IJCSIT/index

Practice and Optimization of Deep Learning Model Training

Yiwei Dong

School of Information Engineering, Chang'an University, Xi'an, Shaanxi, China

ABSTRACT

As IT rapidly advances, deep learning, a key AI area, has achieved significant results in image recognition, NLP, and speech recognition. This research focuses on "Practice and Optimization of Deep Learning Model Training," detailing efficient methods and techniques for practical applications. This paper reviews the basic theory and evolution of deep learning, summarizes some mainstream neural network architectures and their applicable scenarios, and emphasizes the importance of data preprocessing to improve model performance. Then, the common problems such as overfitting, underfitting, gradient disappearing or explosion in the process of model training are deeply analyzed, and the problems such as loss function selection, backpropagation mechanism, optimizer configuration and hyperparameter adjustment are discussed. In the face of the above challenges, a series of efficient optimization schemes are proposed, including data enhancement techniques, transfer learning, regularization methods, adaptive learning rate adjustment, distributed training and mixed precision training, etc., which can accelerate the convergence speed of the model and enhance the final performance. In addition, some concrete cases are selected for empirical study, and the actual effect of the proposed optimization scheme is tested by comparative experiments. This paper reviews deep learning research, discusses future directions, and notes room for optimization in theoretical architecture and computational performance. It offers practical guidance for deep learning experts and a new perspective for related scholars.

KEYWORDS

Deep learning; Model training; Optimization strategies; Hands-on experience; Performance improvement

1. INTRODUCTION

In recent years, deep learning, a key ML branch, has excelled in complex pattern recognition [1]. By mimicking brain neurons, it extracts features from large datasets, enhancing prediction and classification. This drives progress in image recognition, speech processing, and NLP, impacting lifestyles and boosting industries like autonomous driving, medical diagnosis, and financial risk management [2]. Yet, model training faces challenges: high-quality data, architecture selection, hyperparameter tuning, and avoiding over/underfitting. Increasing model complexity raises computing demands, adding difficulties. Thus, exploring efficient training methods and optimizations is crucial for academia.

This paper attempts to present a complete picture of deep learning model training through a combination of theoretical exposition and case analysis, hoping to help all researchers and technical practitioners who want to make progress in this rapidly developing field.

2. FUNDAMENTALS OF DEEP LEARNING

2.1. The Concept and Development of Deep Learning

DL, a machine learning branch, uses neural networks for deep data mining via multi-level transformations. Since the 1940s, neural networks have evolved, reaching prosperity with enhanced computing power, data resources, and algorithms. AlexNet's 2012 ImageNet win marked a new DL era, boosting research and innovation. CNNs are favored for image processing, reducing parameters via local connections and weight sharing, and degrading features via pooling [3]. This enhances spatial invariance and prevents overfitting.

To process sequence data, recurrent neural networks (RNNS) are designed to remember previous state information. However, standard RNNS suffer from vanishing gradients, limiting their ability to capture long-distance dependencies. As a result, variants such as short Term memory networks (LSTM) and gated cycle units (GRUs) have emerged, which effectively solve this problem by introducing special gating mechanisms.

The Transformer architecture abandons the traditional RNN/CNN architecture and relies entirely on the attention mechanism to capture dependencies between sequences. Due to its parallelization characteristics, Transformer greatly speeds up training and has achieved remarkable achievements in natural language processing and other fields, such as BERT and GPT series models [4].

2.2. Data Set Construction and Preprocessing

High-performance DL models rely on quality datasets. Data acquisition methods include open databases, web crawling, and user-generated content [5]. Data purification, such as deduplication, missing value filling, and label correction, is crucial. Numerical features often undergo standardization/normalization to accelerate convergence and improve stability, converting them to a standard normal distribution or a specific range like [0, 1]. For images, augmentation techniques like rotation, inversion, and clipping expand training samples, enhancing generalization. Text data can be enhanced through synonym substitution and sentence insertion.

Finally, in order to evaluate the effectiveness of the model, all data sets should be divided into training sets, verification sets and test sets, and the recommended ratio is 7:1.5:1.5 or 8:1:1, so as to ensure that each part of the data can effectively support the training, tuning and performance testing of the model.

3. MODEL TRAINING PROCESS

3.1. Selection and Design of Loss Function

In the training stage of deep learning model, loss function, as a key index to measure the gap between the predicted output and the real value, occupies an extremely important position. The proper selection of loss function has a decisive influence on the efficiency and accuracy of the model. The following are the definitions of several common loss functions and their mathematical expressions:

Mean Squared Error (MSE)

It is suitable for regression problems to calculate the predicted value

The mean squared variance between $\hat{y}$ and the actual value y. The formula is:

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \quad (1)$$

Cross-Entropy Loss (cross-entropy Loss)

It is widely used in classification tasks, especially in binary or multi-classification problems. It measures the difference between two probability distributions.

Binary Cross-Entropy

For binary classification problems, suppose that the output probability p is obtained after the sigmoid activation function, and the formula is:

$$\text{BCE}(p, y) = -[y\log(p) + (1-y)\log(1-p)] \tag{2}$$

Here $y \in \{0, 1\}$ represents the true label and p is the probability predicted by the model.

Categorical Cross-Entropy

It is suitable for multi-classification scenarios, assuming that the output probability vector is obtained after softmax activation function p, The formula is:

$$\text{CC}E_{(p,y)} = -\sum_{c=1}^{c} y_{c=1}\log(p_c) \tag{3}$$

Where C is the class number, y is the true label vector of the unique thermal coding, and p is the predicted probability distribution.

Hinge Loss

It is commonly used in support vector machines (SVM), but in some cases is also suitable for classification tasks in deep learning. The formula is:

$$\text{HingeLoss} = \max\big(0.1 - y \cdot f(x)\big) \tag{4}$$

Here $y \in \{-1,1\}$ is the real label, and $f(x)$ is the score of the model output (without activation function).

In order to adapt to specific application requirements or solve specific problems, researchers also design customized loss functions. For example:

Intersection ratio (IoU) loss

In the object detection task, the IoU loss function is used to measure the overlap between the predicted bounding box and the actual bounding box, and its mathematical expression is:

$$\text{IoU} = \frac{AreaofOverIap}{AreaofUnion} = \frac{A \cap B}{A \cup B} \tag{5}$$

Adversarial Loss

The anti-loss mechanism introduced in the framework of adversarial generation Network (GANs) can further greatly improve the quality of the generated image by putting the generator and discriminator in competition with each other.Its formula is usually expressed as:

$$\text{minGmaxDV}(D, G) = Ex{\sim}pdata(x)[IogD(x)] + Ez{\sim}pz(x)\left[Iog\left(1 - D\big(G(z)\big)\right)\right] \tag{6}$$

In this context, discriminator, generator, real sample and random noise each play a key role. Various loss functions have their own characteristics and can be adapted to different kinds of tasks and application environments. When selecting the appropriate loss function, we should not only consider the requirements of specific problems, but also comprehensively evaluate the data characteristics, model architecture and other factors to further achieve the optimal training results. Let's wait.

3.2. Introduction to Backpropagation Algorithm

As a widely used optimization technique in the field of deep learning, the core of backpropagation is to evaluate the contribution degree of each layer parameter to the final output error, and to correct the weight and bias accordingly. The basic principle of this method is to calculate the gradient layer by layer from the output end of the network until it reaches the input end. This process can be done efficiently using the Chain Rule. Here are the specific steps and formulas:

Forward Pass

In the forward propagation process, the input data x passes through the linear transformation and activation function of each layer successively, and finally the predicted output $\hat{y}$ is obtained.

For layer I :

Linear transformation:

$$z^{(i)} = W^{(i)}a^{i-1} + b^{(i)} \tag{6}$$

Activation function:

$$a^{(i)} = f\left(z^{i}\right) \tag{7}$$

Where $W^{(i)}$ is the weight matrix, $b^{(i)}$ is the bias vector, f is the activation function, and $a^{(i)}$ is the activation value of layer i .

Calculated loss

Based on the selected loss function L , the error between the model prediction result $\hat{y}$ and the true label y is calculated. For example, when using mean square error (MSE):

$$\mathrm{L} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{8}$$

Or when using cross entropy loss (CCE):

$$\mathrm{L} = -\sum_{c=1}^{c} y_c \log(\hat{p}_c) \tag{9}$$

Backward Pass

The gradient of each layer parameter relative to the loss function is calculated using the chain rule. For the last layer (the output layer), we first calculate the error term $\mathcal{S}^{(L)}$ and then pass it forward layer by layer.

For the output layer L , the error term:

$$\mathcal{S}^{(L)} = \nabla_{a(L)}L \odot f'\left(z^{(L)}\right) \tag{10}$$

For hidden layer I, the error term:

$$\mathcal{S}^{(L)} = \left(W^{i+1}\right)^{T\mathcal{S}^{(i+1)}} \odot f'\left(z^{(i)}\right) \tag{11}$$

Here $\odot$ stands for element level multiplication, and f′ is the derivative of the activation function.

Parameter updating

The optimizer is used to update the model parameters based on the calculated gradient. For the I th layer weight $W^{(I)}$ and bias $b^{(I)}$, the update rule is:

$$W^{(I)} := W^{(i)} - \eta\nabla_{W^{(i)}}L = W^{(I)} - \eta\delta^{(I)}\left(a^{(I-1)}\right)^{T} \tag{12}$$

$$b^{(I)} := b^{(I)} - \eta\nabla_{b_{(I)}}L = b^{(I)} - \eta\delta^{(I)} \tag{13}$$

Where η is the learning rate and controls the step size of parameter update.

The core of the backpropagation algorithm is to efficiently calculate the gradient of each layer using the chain rule, and update the parameters by gradient descent method to minimize the loss function. This process not only ensures that the model can learn useful feature representations from the data, but also provides theoretical assurance that deep neural networks can achieve good performance in practice.

3.3. Optimizer Selection

The optimizer determines how to update the model parameters based on the calculated gradient to minimize the loss function. Different optimizers have their own characteristics and scope of application:

Random gradient Descent (SGD): is a basic optimization strategy, the core of which is that only a single sample is used to update parameters in each iteration. Although this method is concise and clear, it also has the risk of falling into the local optimal solution.

Momentum optimization: Adding momentum terms on top of SGD to help accelerate convergence and overcome local minima.

AdaGrad: Assigns an independent learning rate to each parameter and is especially suitable for dealing with sparse data.

RMSprop: improves the problem that the late learning rate of AdaGrad is too low, and maintains the advantages of adaptive learning rate.

Adam (Adaptive Moment Estimation): Combines the strengths of Momentum and RMSprop to become one of the most popular optimizers available today.

3.4. The Role and Selection of Hyperparameters Such As Batch Size and Learning Rate

The selection of hyperparameters has a significant impact on the effectiveness of model training:

Batch Size: refers to the number of samples used each time a parameter is updated. Larger batches can provide more stable gradient estimation, but also require more memory resources. Smaller batches help explore more possible parameter Spaces, but can lead to unstable training.

Learning Rate: A key factor that controls the length of parameter update steps. A higher learning rate may result in missing the optimal solution, while a lower learning rate will prolong the training time. In practice, the learning rate is usually set in a way that is initially high and then gradually decreasing.

3.5. Identification and Solution of Overfitting and Underfitting

When the model architecture is too complicated, it is easy to cause overfitting phenomenon. Although the fitting degree of the training set is very high, its performance on the test set is greatly reduced. Strategies to solve the overfitting problem include introducing regularization methods (such as L2 regularization, Dropout mechanism), implementing early termination training, expanding the training sample size, or optimizing the model framework to simplify its complexity. On the contrary, if the model fails to effectively extract the feature pattern from the training data, underfitting will occur, which will lead to unsatisfactory prediction effect of the model on the training set. To address underfitting, one can increase model capacity by adding layers or nodes, adjust hyperparameters to better align with data characteristics, or try a different architecture or optimizer.

4. CHALLENGES OF MODEL TRAINING

4.1. Data Imbalance Problem

In practical applications, dataset category imbalance favors majority categories, weakening minority category identification. Several strategies can be adopted to deal with this problem:

Resampling: includes Oversampling and Undersampling. Oversampling is by copying or generating new minority samples to increase their proportion; Undersampling is to reduce the number of samples in the majority class to balance the proportion between the two classes.

Cost-Sensitive Learning: Adjusting the loss function to penalize misclassifying certain classes makes the model focus more on those samples.

Synthetic Minority Over-sampling Technique (SMOTE): Generate new minority samples by interpolating in feature space to avoid overfitting from replication.

4.2. High Dimensional Data Processing

With the increase of data size, the feature dimension also increases, which brings a huge computational burden to model training and may lead to the phenomenon of "dimension disaster". For the effective management of high-dimensional data, the following strategies can be adopted:

Dimensionality reduction: linear dimensionality reduction methods such as principal component analysis (PCA), linear discriminant analysis (LDA), or non-linear dimensionality reduction techniques such as t-SNE and UMAP are used to reduce the number of features while maintaining key information.

Feature selection: By means of statistical testing, mutual information evaluation and recursive feature elimination (RFE), the most representative feature set is selected to further optimize the model architecture. In addition, as a unique neural network design, the autoencoder can learn the compact representation form of input data under unsupervised conditions, and has shown good applicability to various data types such as images and text.

4.3. Tradeoffs Between Model Complexity and Computational Resources

The performance of deep learning models is often positively correlated with their complexity, however, excessive model complexity not only increases training time and memory requirements, but also can lead to overfitting challenges. Therefore, it is necessary to consider multiple dimensions comprehensively in the process of building the model. The first is hardware limitations. Determine the appropriate model size and structure based on the existing GPU/CPU configuration to ensure the efficiency of the training process. The second is the application of model compression technology. By means of pruning, quantization and knowledge distillation, the model volume is reduced while the performance is basically stable. It's a distributed training strategy. With the help of multiple compute nodes, the training task is processed in parallel, so as to accelerate the model convergence and improve the processing ability of large-scale data sets.

4.4. Gradient Disappearance/Explosion Problem

The phenomenon of gradient disappearance or gradient explosion refers to that in the process of backpropagation of a deep neural network, with the increase of the number of layers in the network, the gradient may shrink or enlarge sharply, which further causes the parameter updating to be slow or even abnormal. The following strategies can be adopted to alleviate this problem. The application of weight initialization technology, such as Xavier/Glorot initialization or He initialization, can

ensure that the distribution of activation values at each layer tends to be consistent, and further promote the stable transmission of gradients.

Batch Normalization: By standardizing the processing of each batch of input data, it can alleviate the problem of internal covariate migration and promote gradient flow.

Residual Connections: The introduction of skip connections (such as short-circuit connections in ResNet) allows information to be passed directly from one layer to deeper layers, effectively preventing gradients from disappearing.

5. OPTIMIZATION STRATEGY IN PRACTICE

5.1. The Application of Data Enhancement Technology

Data Augmentation enhances the dataset by generating extra training samples, improving model generalization and robustness. For different types of data, different enhancement techniques can be employed:

Geometric transformation: such as rotation, flipping, cropping, scaling and other operations; Color change: adjust brightness, contrast, saturation, hue and other parameters; Noise addition: Introducing random noise or blurring to simulate changes in the real world; Mixup: Linearly interpolate the input and labels of two different samples to create a new training instance. See Figure 1.

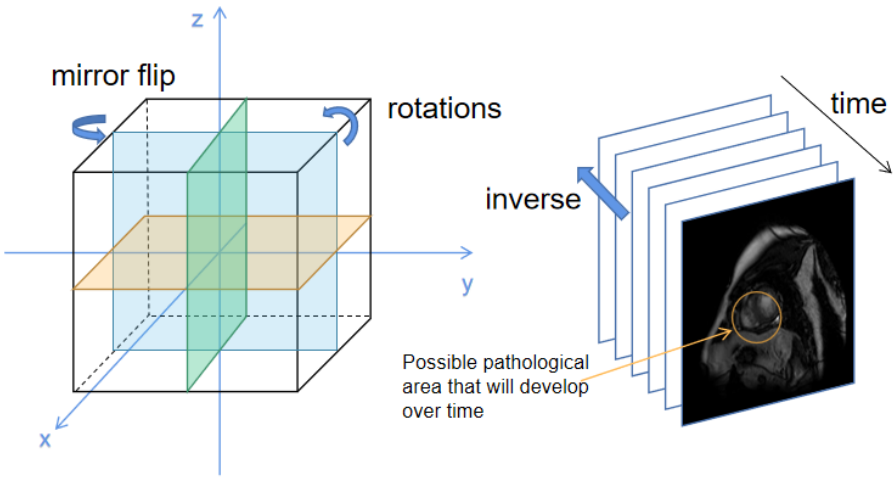

Figure 1. Geometric transformation diagram

Synonym substitution: replacing words in the original text with synonyms in the vocabulary; To insert or delete words at random places in a sentence; Back Translation: The translation of a text into another language and back again to generate variations.

Time Shifting: Changing the time starting point of an audio clip; Pitch Shifting: Adjusting the frequency characteristics of audio; Reverberation: simulates the propagation characteristics of sound in different environments.

5.2. Transfer Learning and Fine Tuning

Transfer learning technology can realize the transfer of knowledge from pre-trained models to new tasks, which greatly reduces the amount of data and computational cost required when training models from scratch. The specific steps are as follows:

First select the pre-trained model: select the appropriate pre-trained model according to the target task, such as ResNet, BERT, etc., on ImageNet. Second, freeze some layers: keep the earlier layers of the model unchanged, and only modify or retrain the top layer to adapt to the needs of the new task. The

third is Fine-tuning: if the target domain has enough labeled data, the model performance can be further optimized by adjusting all or some parameters. Finally, Multi-task Learning: When multiple related tasks share the same feature representation, these tasks can be trained simultaneously to improve each other's performance.

5.3. Regularization Method

Regularization is one of the important means to prevent overfitting. It improves the generalization ability by limiting the complexity of the model. Common regularization methods include: First, L1/L2 regularization, adding the absolute value or sum of squares of the weight to the loss function as a penalty term to encourage sparse or smooth solutions. The second is Dropout, which randomly drops a subset of neurons and their connections to form a different subnetwork structure in each iteration, thus reducing the risk of dependence on a specific neuron. The third is Early Stopping, monitoring the performance indicators on the verification set, and once it is found that there is no improvement, the training will be terminated in advance to avoid over-fitting.

5.4. Adaptive Learning Rate Adjustment

The selection of the appropriate learning rate plays an important role in the convergence rate and final performance of the model, and the adaptive learning rate adjustment mechanism can automatically optimize the learning rate according to the real-time feedback in the training process, and further enhance the training effect.The specific process is: first Learning Rate Decay: gradually reduce the learning rate with the increase of training rounds, such as exponential decay, piecewise constant decay, etc. Then came Cosine Annealing: Periodically adjusting the learning rate, following the cosine curve in each period, helped to get out of the local minimum. Finally, ReduceLROnPlateau: When the loss on the verification set is monitored and no longer decreases, the learning rate is automatically reduced until a predetermined minimum value is reached.

5.5. Distributed Training and Parallel Computing

In order to accelerate the training of large-scale deep learning models, parallelization and distributed training become indispensable techniques: Parallelism and distribution have the following forms.

Data Parallelism: The batch data is divided into several sub-batches, allocated to multiple Gpus/cpus for parallel computation, and finally the results are summarized to update the model parameters.

Model Parallelism: A large model can be split into parts and deployed on multiple devices due to too many parameters for a single machine.

Hybrid Parallelism:Combining data and model parallelism boosts computing efficiency and resolves memory bottlenecks.

Distributed training frameworks: Tools like Horovod, TensorFlow Distributed, and PyTorch Distributed offer easy APIs for efficient node communication and collaboration.

5.6. Accelerate Convergence Using Mixed Precision Training

Mixed Precision Training uses FP32 and FP16 to accelerate training and reduce memory usage without losing accuracy. Automatic Mixed Precision (AMP) simplifies this by deciding operations' precision. Gradient Scaling adjusts gradients to prevent overflow in FP16, ensuring stability and accuracy.

6. CASE STUDY

This chapter will deeply discuss the practice and optimization strategy of deep learning model training through specific application cases. Each case not only shows the technical implementation details for a specific application scenario, but also reflects how to apply the theoretical knowledge and optimization methods introduced in the previous chapters to solve practical problems. By studying these cases, we can better understand the effects of different optimization strategies and their scope of application, and provide valuable references for future research.

6.1. Image Classification Task: CIFAR-10 Dataset Experiment Based on ResNet50

CIFAR-10 is a small color image dataset widely used in image classification research, containing 10 categories with 6,000 32x32 pixel images each. Here, the original image is standardized, and enhancement techniques such as random flipping and cropping are used to expand the data set. The classic ResNet50 architecture was chosen as the base model and fine-tuned. The Adam optimizer was used, the initial learning rate was 0.001, the batch size was set to 128, and the learning rate was adjusted using the cosine annealing strategy.

After several iterations of training, the accuracy of the final model on the test set reaches more than 90%. Of particular note is the introduction of mixed precision training (AMP), which not only significantly reduces training time, but also maintains the same level of performance. In addition, Early Stopping method can effectively avoid overfitting and improve the generalization ability of the model. As shown in Figure 2.

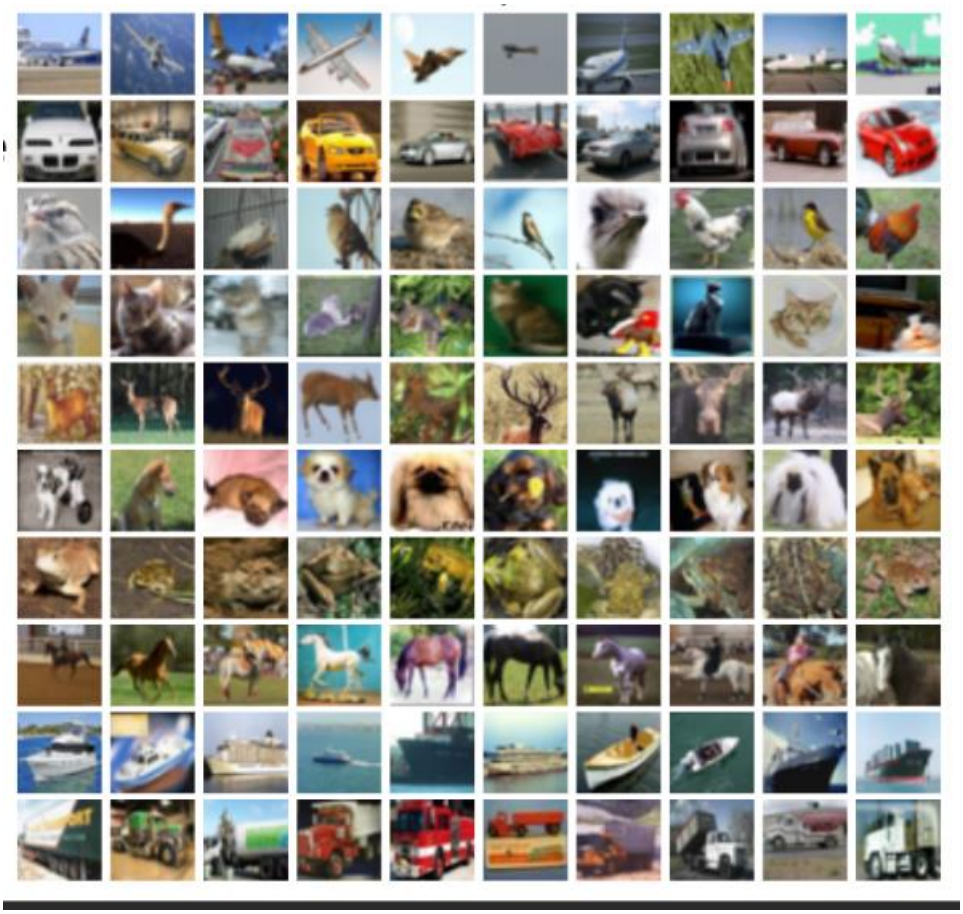

Figure 2. Classification task diagram

6.2. Experiment Result

This article looked at the performance differences between CNNS (method1) that are not regularized and regularized versus CNNS that are Dropout (method2), L2 regularized (method3), and CNNS that are both Dropout and regularized, as shown in Table 1.

Table 1. Accuracy of different methods

Methods	Accuracy
Method 1	78.3%
Method 2	80.2%
Method 3	81.4%
Method 4	86.6%

The table shows method1 (basic CNN) has 78.3% accuracy, suggesting overfitting. Method2 (with Dropout) raises accuracy to 80.2%, alleviating overfitting. Method3 (with L2 regularization) further boosts accuracy to 81.4%, highlighting the effectiveness of regularization.L2 regularization forces the model to learn smoother and simplified features by punishing large weight values, making it perform better on the test set. Using both Dropout and L2 regularization (method4) significantly boosts model accuracy to 86.6%, reducing overfitting and improving generalization for better classification on complex datasets.

7. LATEST DEVELOPMENTS AND FUTURE DIRECTIONS

Recent DL advancements have redefined AI tech and deepened our understanding. Self-supervised learning extracts info from unlabeled data, reducing dependence on labeled data and matching/surpassing supervised learning. Multimodal learning integrates text, images, and audio, enabling applications like image description and speech recognition. Deep reinforcement learning, exemplified by AlphaGo, shows promise in autonomous driving and robotics. As models become more sophisticated and explainable, visualization, LIME, and Shapley values enhance user trust and debugging. Challenges include reducing resource consumption, enhancing model adaptability, building a theoretical foundation, and addressing ethical issues like fairness, privacy, and misuse prevention. Academia-industry collaboration is crucial for a responsible AI ecosystem.

8. CONCLUSION

This paper focuses on the topic of "Practice and Optimization of Deep Learning model training", and systematically discusses how to effectively train and optimize deep learning models in practical applications. Through the discussion, we realize that successful deep learning projects not only rely on advanced algorithms and technologies, but also need careful design and optimization in combination with the actual situation. Although this paper has covered many aspects of deep learning model training practice and optimization, there are still many directions worth further research. In conclusion, this paper not only provides valuable practical experience for professionals engaged in deep learning research, but also points out a new exploration path for researchers in related fields. It is hoped that through the discussion of these issues, more innovative thinking can be stimulated and continuous progress and development in this field can be promoted.

REFERENCES

[1] Shlezinger N, Eldar Y C, Boyd S P. Model-based deep learning: On the intersection of deep learning and optimization [J]. IEEE Access, 2022, 10: 115384-115398.

[2] Sun R Y. Optimization for deep learning: An overview [J]. Journal of the Operations Research Society of China, 2020, 8(2): 249-294.

[3] Yang L, Shami A. On hyperparameter optimization of machine learning algorithms: Theory and practice [J]. Neurocomputing, 2020, 415: 295-316.

[4] Sun R. Optimization for deep learning: theory and algorithms [J]. arxiv preprint arxiv:1912.08957, 2019.

[5] Akay B, Karaboga D, Akay R. A comprehensive survey on optimizing deep learning models by metaheuristics [J]. Artificial Intelligence Review, 2022, 55(2): 829-894.