Optimizing Image Classification with ResNet-18 on the CIFAR-10 Dataset

Written by Xinyu Wang, Jingwei Zhou, Xiaoxu Bai

[1]xw3080@nyu.edu
jz6022@nyu.edu
xb2057@nyu.edu
https://github.com/VioletGo319/ResNet18CIFAR10

Abstract

In this research, the CIFAR-10 dataset was subjected to various image preprocessing methods such as flipping, random cropping, and alterations in color, hue, and brightness to augment dataset diversity and enhance the robustness of the classification model. A modified ResNet-18 architecture was utilized, featuring 4,815,940 parameters distributed across an input layer, four hidden layers with dual residual blocks, and a fully connected output layer, with strategic dropout integration to mitigate overfitting. Optimization techniques included Stochastic Gradient Descent (SGD) and Adam, with dynamic adjustments of the learning rate facilitated by the ReduceLROnPlateau scheduler to fine-tune the training process. The refined model achieved a 80.4% accuracy rate on Kaggle's evaluation benchmarks.

Introduction

The CIFAR-10 dataset is made up of 60,000 color images, each 32x32 in size, distributed across 10 different categories with each category containing 6,000 images. It includes 50,000 images for training and 10,000 for testing. Labels are in the range 0-9. The number at index i indicates the label of the ith image in the array data.

This dataset is split into five training batches and one testing batch, which is a custom version of the original CIFAR-10), with each batch comprising 10,000 images. The testing batch features 1,000 images from each category chosen at random. The training batches are filled with the remaining images arranged randomly, although some batches might have a higher representation of certain categories. Overall, each category is equally represented across the training batches with 5,000 images.

ResNet18 is a variant of the ResNet (Residual Network) architecture, which is widely used for various deep learning tasks, particularly in image classification. It was introduced by researchers at Microsoft in a seminal paper titled "Deep Residual Learning for Image Recognition"(He et al. 2015).

ResNet18 is named for its 18 layers that contain trainable weights, making it one of the shallower versions of the ResNet family, which includes other models with varying depths like ResNet34, ResNet50, ResNet101, and ResNet152. The key feature of ResNet models, including ResNet18, is the use of residual blocks. These blocks incorporate shortcut connections that skip one or more layers and perform identity mapping, adding the input of the block to its output. This helps in combating the vanishing gradient problem, allowing the network to be efficiently trained with deeper architectures.

Copyright © 2024, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Methodology

Our model is predicated on the Residual Network (ResNet) architecture, specifically the ResNet-18 variant, chosen for its robustness in deep network scenarios due to its innovative use of skip connections.

Data Augmentation

Data augmentation is a critical process in training deep learning models, particularly for image recognition tasks. It artificially increases the size and diversity of the training dataset by applying random, realistic transformations to the training images. This not only helps to prevent the model from overfitting but also enhances its ability to generalize to new, unseen data. For the CIFAR-10 dataset, different sets of transformations were applied to the training/validation and test datasets as follows:

Training and Validation Sets For the training and validation images, the following transformations were applied to introduce variability and mimic more real-world conditions:

- **Random Horizontal Flip**: Images are flipped horizontally with a 50% chance, which assumes that horizontal orientation is not crucial for the classification task.
- **Random Rotation**: Images are randomly rotated between -7 and +7 degrees, allowing the model to handle orientation variations.
- **Random Affine Transformation**: Applies random translation, scale, and shear transformations. Scaling is performed between 80% and 120%, and shearing up to 10 degrees.
- **Random Crop**: Images are cropped to 32x32 pixels from padded images (padding of 4 pixels on each side), which

aids in model training to focus on different regions of the image.

- **Color Jitter**: Adjustments are made to the brightness, contrast, and saturation by up to 20% to simulate varying lighting conditions and camera quality.

These transformations were composed using the PyTorch `transforms.Compose` method, ensuring that each image undergoes a randomized sequence of transformations during training, thus enhancing the robustness of the model.

Test Set The test images were subjected to a simpler set of transformations, aimed at normalizing the data without introducing additional variability:

- **Normalization**: Each color channel (Red, Green, Blue) was normalized using mean = [0.485, 0.456, 0.406] and standard deviation = [0.229, 0.224, 0.225]. This transformation is crucial for matching the preprocessing conditions of the training set and ensures that model inputs during testing are processed similarly to training inputs.

The distinct handling of test data transformations ensures that the evaluation of the model's performance is based on realistic, consistent image data, providing a fair measure of its predictive power on new, unseen data.

Mathematical Formulas

Convolution Operation The convolution layers are foundational to the CNN architecture, mathematically described by:

$$y(k,i,j) = \sum_{n=1}^{N} \sum_{a=0}^{A-1} \sum_{b=0}^{B-1} x(n, i+a, j+b) \cdot w(k,n,a,b)$$

where x represents the input feature map, w denotes the kernel for convolution, y is the output feature map, and $A \times B$ are the dimensions of the kernel.

Activation Function Non-linear activation is introduced through the ReLU function, defined as:

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

which enhances the network's ability to capture nonlinearities in the data.

Batch Normalization To improve model stability and convergence, batch normalization standardizes the outputs of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation:

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

$$y^{(k)} = \gamma \hat{x}^{(k)} + \beta$$

where μ_B and σ_B^2 are the mean and variance of the batch, γ and β are learnable parameters, and ϵ is a small constant for numerical stability.

Loss Function The network utilizes the Cross-Entropy Loss for multi-class classification tasks, which is expressed as:

$$\text{L} = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$$

where $y_{o,c}$ is a binary indicator of whether class label c is the correct classification for observation o, and $p_{o,c}$ is the predicted probability of class c for observation o.

Optimization Algorithm Adam optimizer is employed, combining the advantages of two other extensions of stochastic gradient descent. Specifically, the algorithm computes adaptive learning rates for each parameter:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

where θ represents the parameters, η the learning rate, $\hat{m}_t$ and $\hat{v}_t$ are estimates of the first and second moments of the gradients, respectively, and ϵ improves numerical stability.

Training Process

Training involves multiple epochs over the dataset using the backpropagation algorithm to update the network's weights. The model is evaluated on a validation set at the end of each epoch to monitor progress and adjust parameters such as the learning rate.

Training Process

Optimization

- **Loss Function:** CrossEntropyLoss, appropriate for categorical classification tasks.
- **Optimizer:** Adam, with an initial learning rate of 0.0005 and weight decay set at 1×10^{-4}, to refine model weights efficiently.
- **Scheduler:** ReduceLROnPlateau, reducing the learning rate by 50% if no improvement in validation loss is observed over 20 epochs, facilitating fine-tuning of the model in later stages.
- **Hardware Utilization:** Employed DataParallel to leverage multiple GPUs, significantly accelerating the training process.

Metrics

- **Accuracy:** Computed as the primary performance metric to evaluate both training consistency and generalization capability on unseen data.

Pros and Cons of Different Architectures

This subsection evaluates various convolutional neural network architectures, with a focus on their application in image recognition tasks. Each architecture offers unique benefits and limitations, influencing their suitability for specific problems.

VGG **Pros:**

- **Simplicity:** VGG's uniform architecture makes it easy to implement and scale.
- **Feature Richness:** Multiple layers capture a rich set of features, which is beneficial for complex image recognition tasks.

Cons:

- **High Computational Cost:** VGG networks are very deep with many fully connected layers, leading to significant computational and memory requirements.
- **Overfitting Risk:** Due to its depth and complexity, VGG is quite prone to overfitting, especially on smaller datasets.

ResNet **Pros:**

- **Deep Learning Capability:** Allows training of extremely deep neural networks (up to hundreds of layers) by using skip connections to combat the vanishing gradient problem.
- **Adaptability:** Has been shown to perform well on a variety of image recognition tasks and datasets.

Cons:

- **Residual Block Design:** While beneficial, the design of residual blocks can be a limitation when attempting to model non-hierarchical data.
- **Complexity in Understanding Feature Relevance:** The depth and skip connections can make it difficult to interpret which features are most relevant to the model's decisions.

The choice of architecture largely depends on the specific requirements of the application, including the complexity of the task, the volume and variety of the data, and the computational resources available. ResNet was chosen for its efficiency and ability to be deeply layered, which is crucial for handling complex image classification tasks.

Lessons Learned

- **Architectural Insights:** The placement of dropout before activation layers (post-norm) marginally improved model stability and predictive performance.
- **Training Adjustments:** Modulating the batch size and learning rate dynamically in response to training plateaus proved critical in maintaining learning efficacy without sacrificing computational efficiency.
- **Scheduler Impact:** Initial aggressive learning rate reductions were found to precipitate premature model convergence, which was ameliorated by fine-tuning the scheduler's patience and cooldown parameters.
- **Early Stopping:** For this project, we also utilized the early stopping strategy by setting the patience as 40(twice of patience for scheduler). The parameters with lowest validation loss was updated for each iteration. This provides the flexibility for training, especially beneficial when epoch is large and the validation losses fluctuate.

Results

Number of parameters

The architecture comprises approximately 4.82 million total trainable parameters.

Architectural Design

Model Architecture

- **BasicBlock:** Comprises two convolutional layers each with 3x3 filters, followed by batch normalization, ReLU activation, and dropout (10%). A shortcut connection adds the input of the block to its output, mitigating the vanishing gradient problem.
- **Layers:** The network features four main convolutional layers with 42, 84, 168, and 336 filters, respectively. Each layer utilizes BasicBlocks with adaptive strides to manage spatial dimensions efficiently.
- **Classifier:** Post-convolutional processing, a global average pooling layer flattens the output, which is then fed into a fully connected layer to produce final class predictions.

Key Choices

- **Dropout:** Integrated within each BasicBlock and following shortcut connections to regularize the model and prevent overfitting.
- **Channel Configuration:** Initiated at 42 channels, expanding to 336, this setup was optimized to balance computational load against the network's capacity to learn complex features.
- **Pooling:** Average pooling reduces feature dimensions before classification, enhancing the network's focus on relevant spatial hierarchies.

For further details on the code, please visit the following URL: https://github.com/VioletGo319/ResNet18CIFAR10

Final test accuracy

The model implemented for classifying the CIFAR-10 dataset achieved a final validation accuracy of 90.90% with a validation loss of 0.3400. And the test accuracy for public test dataset on Kaggle achieved 80.4%.

Model Performance

The training and validation performance of the model is illustrated in Figure 1On the left, the blue line for training loss goes down, which means the model is learning from the training data. The orange line is the validation loss. It goes down too, but then it stops getting better. This mean the model is starting to memorize the training data instead of learning to predict new data. On the right, we see the model's accuracy. The blue line for training accuracy goes up, which is good. The orange line for validation accuracy also goes up but then doesn't get much higher. This tells us the model has learned as much as it can from the train data. We stopped training the model after 177 rounds because it wasn't getting better,which helps to prevent overfitting.

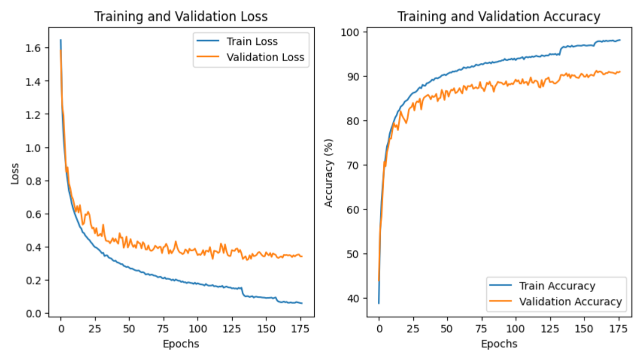

Figure 1: Training and validation loss, and training and validation accuracy over 177 epochs.

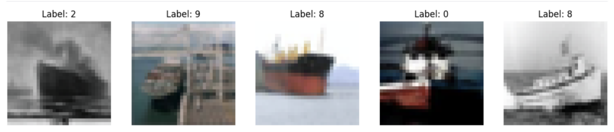

Figure 2: A selection of unlabeled images from the CIFAR-10 dataset and their corresponding labels as predicted by our ResNet-based model.

The image in Figure 2 displays a collection of images from the CIFAR-10 dataset that were not labeled. Our model, which is based on a residual network architecture, has attempted to identify each image. Below each image, we can see the model's guess for what it might represent, with labels like '2' for the first image, '9' for the second, and so on. This visual evidence shows that our model is capable of identifying images it has never seen before, giving us insight into its ability to classify new data.

References

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385.