Convolutional Neural Network for CIFAR-10 Image Classification

Abdulrahman Aroworamimo

May 2024

1 Introduction

This project explores the implementation and evaluation of a Convolutional Neural Network (CNN) for image classification on the CIFAR-10 dataset. The CIFAR-10 dataset consists of 60,000 32x32 color images in 10 classes, with 50,000 training images and 10,000 test images. The aim is to design a CNN architecture that achieves comparable performance with the known benchmark.

2 CNN Architecture and Training

The implemented CNN architecture includes three convolutional layers with batch normalization and max pooling, followed by two fully connected layers with dropout for regularization. The model was trained using the PyTorch Lightning framework, incorporating data augmentation techniques such as random horizontal flipping and random cropping to enhance generalization. Early stopping and model checkpointing were utilized to monitor and optimize the training process, ensuring the best-performing model is saved and training is halted when the validation accuracy stops improving.

3 Results

The CNN achieved the following metrics on the CIFAR-10 test set:

- **Test Accuracy**: 82.20%
- **Test Precision**: 82.29%
- **Test Recall**: 82.23%
- **Test F1 Score**: 82.00%

These metrics demonstrate the robustness of the CNN model. They indicate that the model performs well in predicting each class in the dataset. However,

the performance, while solid, could be further improved. The state-of-the-art Vision Transformer (ViT-H/14) model achieves a top-1 accuracy of 99.5% on CIFAR-10. Achieving similar levels of performance would require a more complex neural network, which would be computationally expensive. In this project, striking a balance between model complexity, resource requirements, and performance was crucial.

4 Data Preprocessing Steps

The images in the CIFAR-10 dataset were preprocessed using the following steps to ensure they were appropriately prepared for input into the Convolutional Neural Network (CNN):

4.1 Normalization

The images were normalized to ensure that the pixel values were scaled to a standard range. This step helps in stabilizing and speeding up the training process. Specifically, the pixel values of each image were normalized to have a mean of (0.4914, 0.4822, 0.4465) and a standard deviation of (0.2023, 0.1994, 0.2010). These values correspond to the channel-wise mean and standard deviation of the CIFAR-10 training set.

4.2 Data Augmentation

To enhance the model's generalization capability and reduce overfitting, data augmentation techniques were applied to the training images:

- **Random Horizontal Flip**: Each image was randomly flipped horizontally with a probability of 0.5. This augmentation helps the model become invariant to the horizontal orientation of objects within the images.
- **Random Crop**: Each image was randomly cropped to a size of 32x32 pixels with a padding of 4 pixels on each side. This augmentation introduces slight variations in the positioning of objects, helping the model become more robust to spatial translations.

4.3 Conversion to Tensors

The images were converted to PyTorch tensors to facilitate their use in the training and evaluation pipelines. This conversion changes the data format from a NumPy array to a tensor, which is the primary data structure used in PyTorch for building and training neural networks.

These preprocessing steps were essential in preparing the CIFAR-10 images for effective training and evaluation of the CNN model. The application of normalization and data augmentation techniques aimed to enhance the model's performance by ensuring standardized input data and increasing the diversity of the training set.

5 The Convolutional Neural Network

The Convolutional Neural Network (CNN) implemented in this project is designed to classify images from the CIFAR-10 dataset. The architecture consists of three convolutional layers, followed by batch normalization and max pooling layers, and two fully connected (dense) layers. The total number of parameters in the model is approximately 1.1 million, indicating a moderately complex model suitable for the CIFAR-10 dataset. The estimated size of the model parameters is 4.592 MB, which is manageable for training on standard hardware. Below is a detailed description of each component of the architecture:

- **Convolutional Layer 1**:
  - Filters: 32
  - Kernel Size: 3x3
  - Padding: 1 (to maintain the spatial dimensions)
  - Activation: ReLU
  - Batch Normalization: Applied after the convolution to stabilize and speed up training.
- **Max Pooling Layer 1**:
  - Pool Size: 2x2
  - Strides: 2
  - Purpose: To reduce the spatial dimensions by a factor of 2, thus downsampling the feature maps.
- **Convolutional Layer 2**:
  - Filters: 64
  - Kernel Size: 3x3
  - Padding: 1
  - Activation: ReLU
  - Batch Normalization: Applied after the convolution.
- **Max Pooling Layer 2**:
  - Pool Size: 2x2
  - Strides: 2
- **Convolutional Layer 3**:
  - Filters: 128
  - Kernel Size: 3x3
  - Padding: 1

  - Activation: ReLU
  - Batch Normalization: Applied after the convolution.

- **Max Pooling Layer 3**:
  - Pool Size: 2x2
  - Strides: 2

- **Fully Connected Layer 1**:
  - Units: 512
  - Activation: ReLU
  - Dropout: 0.5 (to prevent overfitting by randomly setting half of the input units to 0 at each update during training).

- **Fully Connected Layer 2**:
  - Units: 10 (corresponding to the 10 classes in CIFAR-10)
  - Activation: None (Logits will be passed to CrossEntropyLoss which applies Softmax internally)

5.1 Rationale for Choosing This Architecture

The chosen architecture strikes a balance between complexity and computational efficiency. The key considerations for this architecture are:

- **Simplicity and Effectiveness**: The architecture includes three convolutional layers, which are sufficient to capture the hierarchical patterns in the CIFAR-10 images without being overly complex. This is particularly important given the relatively small size of the CIFAR-10 dataset (32x32 pixels).

- **Regularization**: Batch normalization is used after each convolutional layer to normalize the inputs to the layers, which helps in faster convergence and more stable training. Dropout is applied to the fully connected layer to prevent overfitting, which is crucial given the relatively small training set size.

- **Max Pooling**: Max pooling layers are used after each convolutional block to progressively reduce the spatial dimensions and the number of parameters, which helps in reducing the computational cost and controlling overfitting.

- **Metric Logging**: During the test phase, the model computes and logs additional metrics such as precision and F1 score using the `torchmetrics` library. This provides a more comprehensive evaluation of the model's performance beyond just accuracy.

6 Defining the Checkpoint and Early Stopping Callbacks

The checkpoint callback was implemented to save the best-performing model based on validation accuracy, ensuring that the optimal model is retained. The early stopping callback was implemented to prevent overfitting by halting training when the validation accuracy stops improving for a specified number of epochs.

7 Training the Model

7.1 Training Process and Early Stopping

The model was trained for a total of 38 epochs. During the training process, the validation accuracy was monitored to evaluate the model's performance on the validation set. The early stopping callback was triggered at the 38th epoch after three consecutive epochs without improvement in validation accuracy, halting the training to prevent overfitting. The training process concluded at that point with the best validation accuracy recorded at 83.5%.

8 Evaluating the Model on the Test Set

8.1 Test Set Results and Benchmark Comparison

The model was evaluated on the CIFAR-10 test set, achieving the following metrics:

- **Test Accuracy**: 82.20%
- **Test F1 Score**: 82.00%
- **Test Precision**: 82.29%
- **Test Recall**: 82.23%
- **Test Loss**: 0.5383

These results indicate solid performance on the CIFAR-10 dataset, reflecting the model's ability to accurately classify images.

In comparison, the Vision Transformer (ViT-H/14) model, as reported in the paper *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*, achieves significantly higher metrics on the same dataset:

- **Top-1 Accuracy**: 99.5%
- **Number of Parameters**: 632 million

The ViT-H/14 model's superior performance can be attributed to several factors:

- **Model Complexity**: The ViT-H/14 model has 632 million parameters compared to the 1.1 million parameters in this model, enabling it to capture more intricate patterns and features.
- **Training Data**: The model was trained using extra training data, as mentioned in Paperswithcode.
- **Resource Requirements**: The ViT-H/14 model's increased number of parameters requires significantly more computational resources for training and inference. The architecture adopted in this project balances efficiency with performance due to limitations in computational resources.

Overall, while this model performs well, the Vision Transformer model achieves state-of-the-art results by leveraging a much larger and more complex architecture.