Individual Project II: Image Classification

Sourav Panda

sbp5911@psu.edu

October 28, 2023

1 Methodologies

The project is organized as follows, with the code provided in a separate file:

1.1 Data Preparation

1.1.1 Download CIFAR-10 Training Data and Add Augmentations

In the data preparation phase, the CIFAR-10 training dataset is acquired through torchvision, forming the core dataset for training the convolutional neural network (CNN). This dataset comprises a wide range of images spanning ten distinct classes. Concurrently, data augmentation techniques are employed to increase the dataset's diversity and robustness, ultimately enhancing the model's ability to generalize. These techniques involve random image rotation within a specified range, random resizing, and cropping from the center to introduce varying object scales and perspectives, horizontal flipping for diversity, and pixel value normalization, which contributes to faster and more stable training.

1.1.2 Partition the training data into Training and Validation subsets.

During this stage, the training dataset is partitioned into two distinct subsets, namely the Training Set, which accounts for 80% of the data, and the Validation Set, which represents the remaining 20%. The purpose of the Training Set is to facilitate the training of the neural network by enabling it to identify patterns and characteristics present in the data. On the other hand, the Validation Set serves an essential purpose in evaluating the model's ability to generalize to new data and prevent overfitting.

1.2 Test Set and Data Visualization

The "Test Set" section loads pre-processed test images, which are essential for assessing the model's ability to classify new, unseen data and validating its preparedness for real-world application. In addition, the "Data Visualization" section introduces a flexible function for image and label visualization, allowing for rapid tests of training data integrity and a concise overview of the dataset.

1.3 CNN Model

In this section, we define a Convolutional Neural Network (CNN) using PyTorch's nn.Module class. The model's architecture is composed of key layers and components:

Convolutional Layers: These layers extract features from input data through convolution operations, enabling pattern and feature identification in the images.

Batch Normalization: To enhance training stability and convergence, batch normalization is introduced, mitigating issues related to internal covariate shift during training.

ReLU Activation Functions: Rectified Linear Unit (ReLU) activation functions introduce non-linearity into the model, allowing it to learn intricate data representations.

Max-Pooling Layers: Max-pooling downsizes feature maps spatially, preserving crucial information while reducing dimensions.

Fully Connected Layer: The model concludes with a fully connected layer responsible for classifying input images and making the final class assignments.

This well-structured CNN architecture forms the backbone for feature extraction and classification, enabling accurate predictions on the CIFAR-10 dataset.

1.4 Optimizer and Loss Function

In this section, two critical elements for training the neural network are addressed:

Loss Function (Cross-Entropy): The chosen loss function, Cross-Entropy, is applied for classification tasks. It quantifies the dissimilarity between predicted class probabilities and actual class labels. Minimizing this loss during training leads to improved classification accuracy.

Optimizer (Stochastic Gradient Descent - SGD): Stochastic Gradient Descent is the optimizer of choice with a learning rate of 0.01. SGD is a widely used optimization algorithm that iteratively adjusts model parameters to minimize the loss.

1.5 Training Loop

The training loop spans 100 epochs, involving forward and backward training of the CNN model. Each epoch includes the computation of projected outputs and loss gradients. Performance is evaluated through training and validation losses. Training loss gauges errors within the training data, while validation loss assesses model performance on a separate dataset. Losses are recorded in each epoch to track the model's learning progress.

1.6 Plot Training and Validation Loss Curves

Training and validation loss curves are crucially visualized after training. The training progress shows how the model's loss changes. This is used to evaluate convergence and model learning. Overfitting (marked by a considerable drop in training loss and an increase in validation loss) or underfitting (with continuously high losses for both) can be found by analyzing the training and validation loss curves. This visualization guides adjustments to the model as needed. This image helps explain the model's training and generalization.

1.7 Make Predictions on Test Images

The trained model is used to make predictions on the test images. Predicted labels are obtained, and a sample of test images with their predicted labels is displayed.

2 Training and Results

The model is trained for 100 epochs, during which the training and validation losses are constantly monitored for evaluation. The declining training loss indicates the model's capacity to acquire knowledge from the given training data, progressively enhancing its ability to make accurate predictions. In order to ascertain the model's ability to apply its learned knowledge to new, unknown data, validation loss plays a crucial role as an essential metric for evaluating its performance.

In addition, the project utilizes data visualization techniques to graphically depict the training and validation loss curves, offering a visual depiction of the model's performance during the duration of the project. The provided visual depiction portrays the downward trajectory of the training loss, hence revealing the model's aptitude for acquiring knowledge. Nevertheless, a significant observation to highlight is the validation loss, which tends to reach a plateau or exhibit an upward trend. The observed behavior is indicative of overfitting, a phenomenon in which the model demonstrates exceptional performance in memorizing the training data but encounters difficulties in efficiently generalizing its knowledge. This stage is crucial for refining the model to achieve optimal performance and robustness.

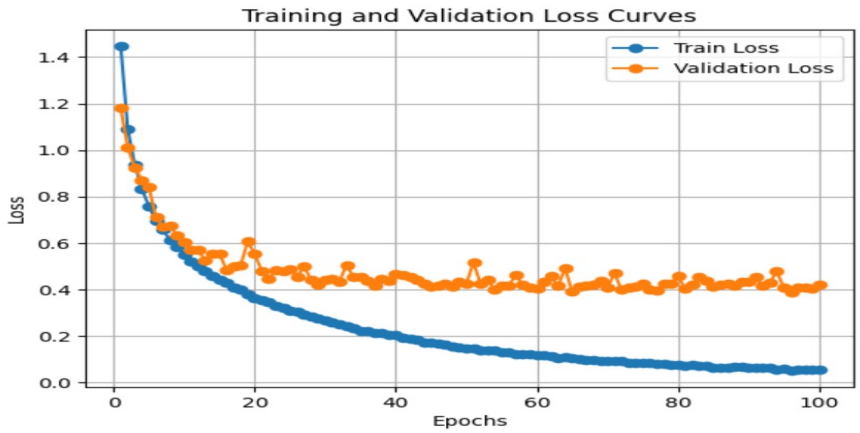

Figure 1: Loss Curves

3 Additional Experiments

In the pursuit of optimizing the model's performance and gaining deeper insights into its behavior, a series of additional experiments were conducted. These experiments encompassed various facets of the machine learning pipeline, including:

Model Variations: The project explored different model architectures, including ResNet-18, VGG(Visual Geometry Group), and models without batch normalization. This investigation aimed to assess how different model designs impact classification accuracy and training stability.

Optimizer Diversity: To understand the impact of optimization algorithms, both Adam and SGD were employed as optimizers for different models. This allowed for a comparative analysis of their performance in terms of convergence speed and final accuracy.

Worker Adjustments: The number of data loader workers was adjusted to observe the effects on data loading and training efficiency. This experiment shed light on the optimal number of workers for the given hardware configuration.

Batch Size Variation: Different batch sizes were tested to gauge their influence on training dynamics, model convergence, and memory consumption. This provided insights into the trade-offs between batch size and training speed.

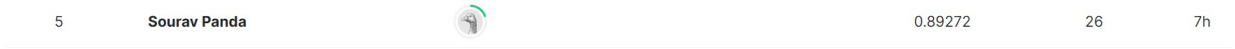

Figure 2: Final Leader board position - 5

4 Conclusion and Key Takeaways

The CIFAR-10 dataset is used to demonstrate CNN model construction and training for image classification in this research. Throughout this project, several key takeaways have emerged. Firstly, data augmentation has greatly improved the model's ability to generalize and detect objects under different settings. This stresses the importance of training dataset diversity. Additionally, the project highlights the significance of batch normalization for training stability and convergence, addressing challenges associated with internal covariate shift and contributing to a smoother training process. The variety of CNN architectures, including ResNet-18, VGG, and models without batch normalization, emphasizes the necessity of choosing the right model for a task. Lastly, the exploration of different optimization strategies, including Adam and SGD, has shed light on their unique impacts on training dynamics and final accuracy. Understanding optimizer behavior is essential for precise training modification. Overall, this project encapsulates the essence of effective image classification with CNNs and lays the groundwork for further advancements in the field.