PyTorch On CIFAR-10 Report

Rahul Jha | rahuljha@umd.edu | University of Maryland College Park

Accomplishments Description

- **Achieved Validation Accuracy:** I was able to create a convolutional neural network (CNN) that reached a validation accuracy of 77.70% in 10 epochs by using a methodical approach to model building and training. This result exceeded the original target of 70%, indicating the efficacy of the selected architecture and hyperparameters.
- **Implemented Batch Normalization:** Batch normalization was incorporated following each convolutional layer to improve the model's training effectiveness. By normalizing the activations, this method stabilizes the learning process, resulting in less sensitivity to weight initialization and quicker convergence rates. This feature's inclusion was crucial in raising training's general accuracy and consistency.
- **Utilized Dropout Regularization:** I used dropout regularization with a probability of 0.5 to counteract the issue of overfitting, which is prevalent in deep learning with small samples. By randomly deactivating a fraction of neurons during training, this method encourages variety in feature learning and improves the model's ability to generalize to new data. Consequently, the model performed better on the validation set.

Key Tasks:

- **Network Architecture Design:** The CNN was designed with four convolutional layers, each of which was followed by ReLU activation functions and batch normalization. Dropout layers were used to regularize the model, and max pooling layers were positioned to minimize spatial dimensions. In order to generate class scores, the output from the convolutional layers was flattened and then run through a fully connected layer. In order to efficiently extract hierarchical information from the input photos, this architecture was carefully selected.
- **Hyperparameter Tuning:** Because of its adaptable learning capabilities, I chose to start the training process with a learning rate of 1e-3, which is a typical starting point for the Adam optimizer. In order to strike a compromise between regularization and effective training, a dropout probability of 0.5 was established using best practices. Achieving the desired validation accuracy was made possible in large part by this careful hyperparameter selection.
- **Training and Evaluation:** The train_part34 function was used to do the training, handling parameter updates, loss computation, and forward and backward passes. I kept a careful eye on the validation set's accuracy and loss metrics, recording significant iterations when losses were recorded as 0.6811 and 0.5938. This continuous assessment allowed for necessary modifications and offered insight into the model's learning trajectory.

Missed Points:

- **Additional Hyperparameter Optimization:** Although the model achieved the desired accuracy, more hyperparameter adjustment may result in even greater performance. Investigating changes in the batch size, dropout frequency, and learning rate may help find a better model configuration and further improve training.
- **Further Architecture Exploration:** Using more intricate architectures, including deeper networks with residual connections or more convolutional layers, may open up possibilities for increasing accuracy. Examining different architectures, like as Inception or ResNet, may show improvements in feature extraction, which would raise performance above the present levels.
- **Comprehensive Evaluation Metrics:** Particularly in multi-class classification settings, determining other evaluation metrics like precision, recall, and F1-score in addition to accuracy may offer a more comprehensive picture of the model's performance. Potential class imbalances would be highlighted by this thorough study, which would also help guide model modifications.

Explanation of Implementation Decisions

- **Batch Normalization:** Because batch normalization may reduce internal covariate shift, which is essential for deep network training, it was chosen to be used. It stabilizes learning and permits the utilization of larger learning rates by normalizing activations. Rapid convergence and increased validation accuracy were made possible by this feature.
- **Dropout Regularization:** To reduce the possibility of overfitting in a model trained on a small dataset, dropout was purposefully used. The model's capacity to generalize was improved by forcing it to learn duplicate representations by the deactivation of certain neurons during training. This regularization method was essential to enhancing the resilience of the model.
- **Sequential API Usage:** The CNN was constructed in a simplified and modular manner by using PyTorch's nn.Sequential for model creation. The code structure was made simpler by this design decision, which made it simpler to understand, edit, and debug. This is particularly useful in group settings and for further model development iterations.

Critical Thinking and Analysis

- **Evaluation of Model Performance:** The model's performance shows successful learning and feature extraction from the dataset, with a validation accuracy peaking at 78.60%. Although the model performed well on validation data, its generalization to unknown data might be improved, as indicated by the observed test set accuracy of 76.63%, which points to a little amount of overfitting. This disparity calls for more research into the intricacy of the model and any overfitting problems.
- **Training Dynamics:** The measured losses varied between 0.5 and 0.8 during training, suggesting that the model was learning successfully. Variations in these measures, however, might indicate problems like overfitting or underfitting, thus regular monitoring is crucial. Further insights into model behavior and guidance for future training procedures may be obtained by a more thorough examination of learning rates and loss curves.
- **Trade-offs in Model Complexity:** The performance of the architecture is weighed against its complexity; deeper networks may boost accuracy, but they also raise the danger of overfitting and demand more computing power. In order to identify the best solution for the job, future iterations might examine both simpler and more complicated designs to investigate this balance.
- **Lost Opportunities:** There was a chance to improve model performance even more by experimenting with different optimizers or cutting-edge methods like data augmentation or transfer learning. Investigating these possibilities may result in increases in training accuracy and speed, offering a stronger foundation for further research.