Harry Vu

CSE 432

Professor Feng

May 5, 2023

Final Report

I. Initial Model:

1. To augment the training dataset, the following data preprocessing techniques were applied:

- RandomCrop with padding 4
- RandomHorizontalFlip
- RandomRotation with a 15-degree range
- Normalization with mean (0.4914, 0.4822, 0.4465) and standard deviation (0.2023, 0.1994, 0.2010)
- The test dataset was normalized with the same mean and standard deviation values as the training dataset.

2. Model:

An initial CNN model was developed with the following architecture:

- Feature Extractor:
  - Conv2D layer with 32 filters, kernel size 3x3, stride 1, padding 1, followed by ReLU activation

- Conv2D layer with 64 filters, kernel size 3x3, stride 1, padding 1, followed by ReLU activation
- Conv2D layer with 128 filters, kernel size 3x3, stride 1, padding 1, followed by ReLU activation
- Conv2D layer with 256 filters, kernel size 3x3, stride 1, padding 1, followed by ReLU activation
- Conv2D layer with 256 filters, kernel size 3x3, stride 1, padding 1, followed by ReLU activation
- MaxPool2D layer with kernel size 2x2, stride 2

- Classifier:
  - Fully connected layer with 512 neurons, followed by ReLU activation
  - Fully connected layer with 256 neurons, followed by ReLU activation
  - Fully connected layer with 10 neurons
  - The model was trained using the following hyperparameters:

Batch size: 128

Epochs: 50

Learning rate: 0.01

Optimizer: Stochastic Gradient Descent (SGD) with momentum 0.9 and weight decay 5e-4

3. **Results:**

The initial CNN model achieved an accuracy of 86% on the CIFAR-10 test dataset. This result provides a solid foundation for further improvement through modifications in the model architecture, data preprocessing techniques, and hyperparameter tuning.

II. Exploring Various Techniques for Optimal Model Performance

1. **Experimenting with different numbers of layers:**

- 7 convolutional layers + 3 fully connected layers
- 9 convolutional layers + 1 fully connected layer
- 6 convolutional layers + 4 fully connected layers
- 8 convolutional layers + 2 fully connected layers

Result: The optimal architecture consists of 8 convolutional layers and 2 fully connected layers.

2. **Evaluating various activation functions:**

- ReLU()
- LeakyReLU(0.1)
- LeakyReLU(0.5)
- LeakyReLU(1)
- Tanh()
- Sigmoid()

Result: LeakyReLU(0.1) provided the best performance among the activation functions tested.

3. **Implementing additional data augmentation techniques:**

- transforms.RandomCrop(32, padding=4)

- transforms.RandomHorizontalFlip()
- transforms.RandomRotation(15)
- transforms.RandomGrayscale(p=0.1)
- transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.1)
- transforms.RandomAffine(degrees=15, translate=(0.1, 0.1), scale=(0.9, 1.1), shear=10)
- transforms.RandomVerticalFlip(p=0.5)
- transforms.ToTensor()
- transforms.Normalize(train_mean, train_std)

Result: Excessive data augmentation led to underfitting. Therefore, only five random data augmentation techniques were used: random crop, random horizontal flip, random rotation, random grayscale, and color jitter.

4. **Modifying the test and main functions to display average loss and accuracy for both training and testing sets, along with line graphs illustrating the model's performance history.**

Result: Visualizing the model's performance in this manner allowed for a better understanding of its behavior, including its fit to the training and testing sets, loss reduction, accuracy improvement, and potential overfitting.

5. **Implementing batch normalization:**

Result: This technique facilitated faster model convergence, improved gradient flow, and provided regularization to reduce overfitting.

6. **Experimenting with the number of max pooling layers:**

Result: only 1 max pooling layers were included in the final model.

7. **Incorporating dropout:**

- Tested various dropout layer quantities and dropout rates.

Result: The optimal configuration included approximately four dropout layers with dropout rates of around 0.25 and 0.5.

8. **Adding residual blocks (Basic blocks):**

- Facilitated gradient flow through shortcut connections.

Result: The model's performance improved, but only 1 to 2 residual blocks were necessary due to the project's limited size (maximum of 10 layers).

9. **Integrating drop block:**

- Included one drop block to mitigate overfitting.

10. **Testing different optimizers:**

- Stochastic Gradient Descent (SGD)
- AdaGrad (Adaptive Gradient)
- RMSprop (Root Mean Square Propagation)

Result: SGD, which was used for the initial model, proved to be the most effective optimizer.

11. **Investigating second-order optimization:**

Result: Due to the high computational complexity and slow convergence speed, this approach was deemed unsuitable for classifying the CIFAR-10 dataset.

12. Assessing various momentum and weight_decay values for SGD:

Result: The best combination was found to be momentum=0.9 and weight_decay=5e-4.

13. Comparing different loss functions:

- Mean Squared Error (MSE) Loss
- Cross-Entropy Loss
- L1 Loss
- Binary Cross-Entropy Loss

Result: Cross-Entropy Loss was the most effective loss function for the model.

14. Utilizing a scheduler (Multi-step learning rate):

Result: Implementing a scheduler enabled the model to begin with a larger learning rate for faster training and then gradually decrease the learning rate as the model reached certain epochs. This approach contributed to improved performance and convergence.

15. Experimenting with different batch sizes and epochs:

Result: The optimal batch size for my model was found to be 128, with approximately 100 epochs (There isn't a specific number of epochs, as the model is configured to stop when convergence is reached).

III. Final Model:

1. **Data Preprocessing and Augmentation:**

Before training, the dataset's mean and standard deviation were calculated for each channel to normalize the data. For data augmentation, the following techniques were applied on the training set:

- RandomCrop with padding=4
- RandomHorizontalFlip
- RandomRotation with 15 degrees
- RandomGrayscale with probability=0.1
- ColorJitter with brightness=0.2, contrast=0.2, saturation=0.2, and hue=0.1

2. **Model:**

- **Feature Extractor:**

1. Convolutional layer (conv1): Conv2d with 3 input channels, 32 output channels, kernel size of 3, stride of 1, padding of 1, and no bias.
2. Batch Normalization (bn1): BatchNorm2d with 32 features.
3. Activation function: LeakyReLU with a negative slope of 0.1.
4. Dropout: Dropout layer with a dropout rate of 0.25.
5. Convolutional layer (conv2): Conv2d with 32 input channels, 64 output channels, kernel size of 3, stride of 1, padding of 1, and no bias.
6. Batch Normalization (bn2): BatchNorm2d with 64 features.
7. Activation function: LeakyReLU with a negative slope of 0.1.

8. Dropout: Dropout layer with a dropout rate of 0.25.

- **BasicBlock Layers:**

9. BasicBlock1: Consisting of two Conv2d layers, two BatchNorm2d layers, and LeakyReLU activation function with 64 input channels, 128 output channels, and a stride of 2.
10. BasicBlock2: Consisting of two Conv2d layers, two BatchNorm2d layers, and LeakyReLU activation function with 128 input channels, 256 output channels, and a stride of 2.

- **DropBlock Layer:**

11. DropBlock2D: DropBlock2D layer with a drop probability of 0.1 and a block size of 3.

- **Convolutional Layer:**

12. Convolutional layer (conv3): Conv2d with 256 input channels, 512 output channels, kernel size of 3, stride of 2, padding of 1, and no bias.
13. Batch Normalization (bn3): BatchNorm2d with 512 features.
14. Activation function: LeakyReLU with a negative slope of 0.1.
15. Dropout: Dropout layer with a dropout rate of 0.25.
16. Convolutional layer (conv4): Conv2d with 512 input channels, 1024 output channels, kernel size of 3, stride of 2, padding of 1, and no bias.
17. Batch Normalization (bn4): BatchNorm2d with 1024 features.
18. Activation function: LeakyReLU with a negative slope of 0.1.

19. MaxPool2D layer with kernel size 2x2, stride 2
20. Dropout: Dropout layer with a dropout rate of 0.25.

- **Classifier:**

21. Fully connected layer (fc1): Linear layer with 1024 * 1 * 1 input features and 2048 output features.
22. Activation function: LeakyReLU with a negative slope of 0.1.
23. Dropout: Dropout layer with a dropout rate of 0.5.
24. Fully connected layer (fc2): Linear layer with 2048 input features and 10 output features (corresponding to the number of classes).

Overall: 8 Convolutional Layers, 2 Fully Connected Layers

2. Results:

The custom CNN model achieved a final test accuracy of 92.21%. The training and testing losses and accuracies were plotted against the number of epochs to visualize the model's performance during training. The total execution time for training and testing was also reported.

Batch size: 128

Epoches: 126

Traning and Testing total execution time is: 19128.993139982224 seconds

3. Conclusion:

The custom CNN model effectively classified the CIFAR-10 dataset with a test accuracy of 92.21%. The data preprocessing and augmentation techniques, along with the model architecture and training strategy, contributed to the high performance of the model.