Analysis of Learning Rates

wwx 12112245

November 3, 2023

Abstract

This assignment involved a series of experiments aimed at investigating the impact of different initial learning rates on the performance of a deep learning model. Learning rates of 1e-1, 1e-2, and 1e-3 were examined. The primary objective was to assess how these varying learning rates affected the training process and final accuracy of the model.

1 Introduction and Overview

Learning rate analysis is a fundamental aspect of training machine learning and deep learning models. The learning rate plays a pivotal role in determining the convergence and overall performance of these models. It represents the step size at which the model updates its parameters during the training process. Choosing an appropriate learning rate is a critical task for practitioners, as it can significantly influence the training dynamics, convergence speed, and final accuracy of a model.

1.1 Experimental Setup

- **Dataset**: CIFAR-10
- **Model**: Convolutional Neural Network (ConvNet)
- **Hyperparameters**:
  - **Batch Size**: 128
  - **Number of Epochs**: 30
  - **Evaluation Interval**: 1 epoch
  - **Optimizer**: Stochastic Gradient Descent (SGD) with Momentum
    - **Initial Learning Rates**: 1e-1, 1e-2, 1e-3
    - **Momentum**: 0.9
    - **Step Size for Learning Rate Decay**: 5 epochs
    - **Gamma for Learning Rate Decay**: 0.5

1.2 Experimental Procedure

1. We conducted three sets of experiments, each with a different initial learning rate (1e-1, 1e-2, and 1e-3).
2. For each experiment, we trained the ConvNet model on the CIFAR-10 dataset for 30 epochs.
3. During training, we recorded the training loss, training accuracy, testing loss, and testing accuracy at the end of each epoch.
4. We visualized the training and testing curves to analyze the impact of different learning rates.

2 Algorithm Implementation and Development

This study employs a Convolutional Neural Network (ConvNet) implemented in PyTorch. The CIFAR-10 dataset is selected as the evaluation dataset, known for its extensive array of ten distinct classes, encompassing airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks.

2.1 Dataset

We define the dataset and its transformations. In this instance, we utilize the CIFAR-10 dataset, applying specific transformations for both training and testing data.

```
# CIFAR-10 transforms
transform_cifar10_train = transforms.Compose([
    transforms.RandomCrop(32, padding=4),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    transforms.Normalize((0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010)),
])

transform_cifar10_test = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010)),
])

train_set = torchvision.datasets.CIFAR10(root='../data', train=True,
                                         download=True, transform=
                                            transform_cifar10_train)
train_dataloader = torch.utils.data.DataLoader(train_set, batch_size=BATCH_SIZE,
                                           shuffle=True, num_workers=2)

test_set = torchvision.datasets.CIFAR10(root='../data', train=False,
                                        download=True, transform=
                                           transform_cifar10_test)
test_dataloader = torch.utils.data.DataLoader(test_set, batch_size=BATCH_SIZE,
                                          shuffle=False, num_workers=2)

class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', '
    horse', 'ship', 'truck']
```

2.2 Model

We introduce the structure of the ConvNet model for multi-class classification.The model is trained using the Stochastic Gradient Descent (SGD) optimizer with a specified learning rate and momentum.

```
class ConvNet(nn.Module):
    def __init__(self):
        super(ConvNet, self).__init()
        self.conv1 = nn.Conv2d(3, 4, 3)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(4, 8, 3)
        self.fc1 = nn.Linear(8 * 6 * 6, 32)
        self.fc2 = nn.Linear(32, 10)

    def forward(self, x):
        x = self.pool(torch.relu(self.conv1(x)))
        x = self.pool(torch.relu(self.conv2(x)))
        x = x.view(-1, 8 * 6 * 6)
        x = torch.relu(self.fc1(x))
```

```
        x = self.fc2(x)
        return x

model = ConvNet()
model.to(device)
```

2.3 Optimizer

We define the optimizer and learning rate scheduler for our model training.

```
optimizer = optim.SGD(model.parameters(), lr=LEARNING_RATE, momentum=MOMENTUM)
scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=STEP, gamma=GAMMA
    )
```

3 Learning Rate Experiments

In this section, we conducted a series of experiments to examine the impact of different learning rates (1e-1, 1e-2, and 1e-3) on the training process and performance. We monitored the loss, accuracy, and F1 score for each learning rate at every epoch and recorded the total time taken for model training. Additionally, we created visualizations to illustrate the influence of these learning rates on the model's performance.

3.1 Per Batch Training/Testing

We denfine two function which are essential for training and evaluating machine learning models using batched data from dataloaders.

```
def train_batch(model, image, target, criterion):
output = model(image)
loss = criterion(output, target)
return output, loss
def test_batch(model, image, target, criterion):
output = model(image)
loss = criterion(output, target)
return output, loss
# Training loop
for epoch in range(NUM_EPOCHS):
model.train()
# ... (Training procedure)
# Testing loop
if (epoch + 1) % EVAL_INTERVAL == 0 or (epoch + 1) == NUM_EPOCHS:
model.eval()
# ... (Testing procedure)
```

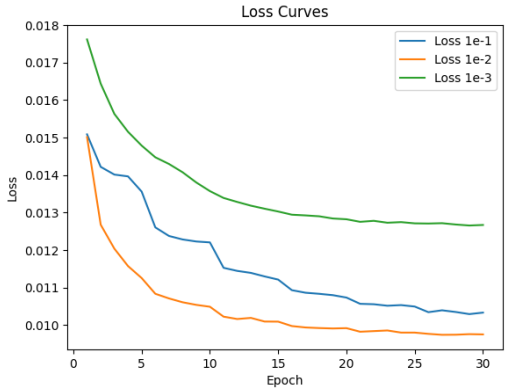
3.2 Evaluation Metrics and Results Logging

During both the training and testing phases, various metrics are meticulously recorded at the end of each epoch. These metrics encompass training loss, training accuracy, test loss, test accuracy, and the F1 score.The F1 score is a metric that effectively gauges the model's ability to strike a balance between precision and recall, making it a crucial indicator of classification performance The results of each epoch epoch, including the aforementioned metrics, are stored in several files,in order to facilitate easy access.

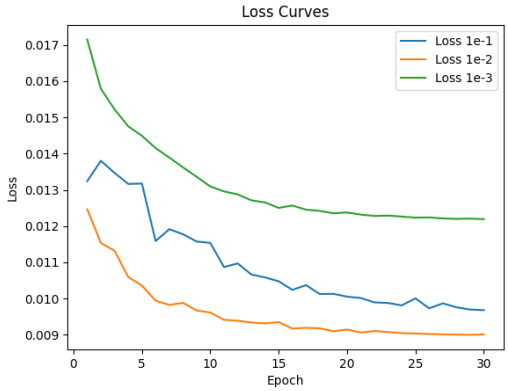
4 Visualization of the results

4.1 Loss curves

Loss curve reflects the model's training progress and its ability to minimize error during training. It helps in assessing model convergence, identifying overfitting or underfitting, tuning the learning rate, and comparing model performance.



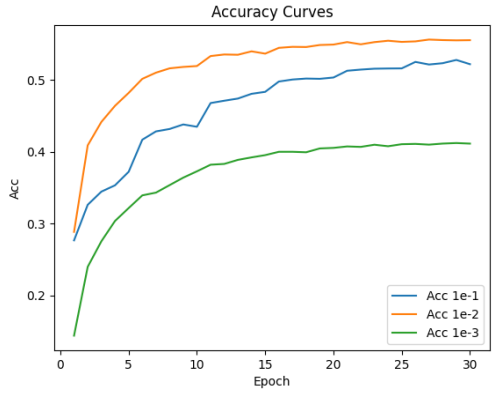
(a) train loss



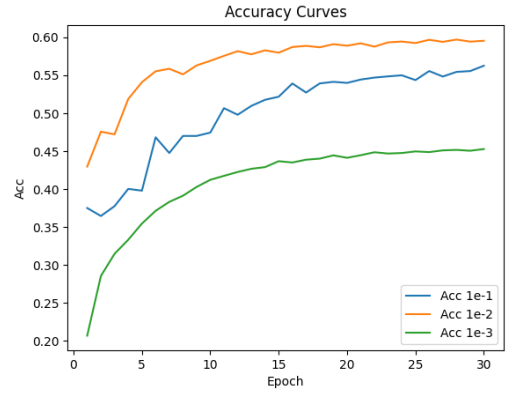
(b) test loss

4.2 Accuracy Curves

Accuracy curves reflect how the model's classification performance changes over epochs, showing its ability to make correct predictions during training and testing.



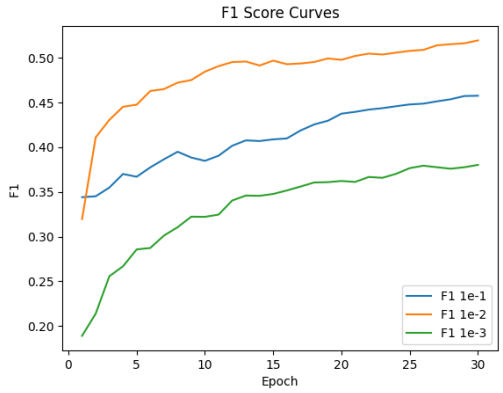
(a) train acc



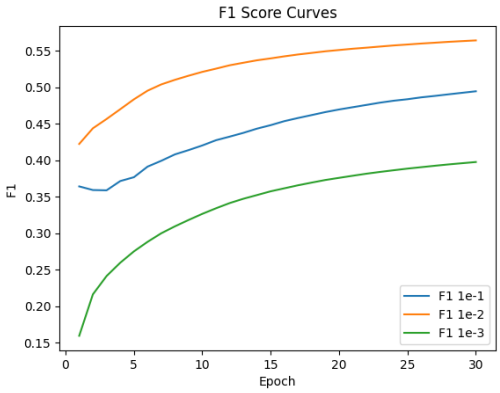
(b) test acc

4.3 F1 Score

The F1 score is a composite performance metric that reflects the trade-off between the precision and recall of a classification model.



(a) train f1



(b) test f1

4.4 Time cost

Time cost reflects the amount of time required to complete a specific task or operation and is often used to assess efficiency and performance.

	Learning Rate 1e-1	Learning Rate 1e-2	Learning Rate 1e-3
Time	233.839	233.839	233.839

5 Conlusion and discussion

5.1 Analysis of Experimental Results

5.1.1 Impact of Learning Rates on Training Process

We examine the influence of different learning rates (1e-1, 1e-2, and 1e-3) on the training process. We observed that both 1e-1 and 1e-2 converged quickly compared to 1e-3. However, the training process for 1e-1 was less stable. In contrast, a smaller learning rate (1e-3) led to a more stable training process but required more time to achieve the same level of accuracy.

5.1.2 Impact of Learning Rates on Final Performance

We assess the impact of learning rates on the final performance of the model. We found that the model with a learning rate of 1e-2 performed best during training, achieving the highest test accuracy and F1 score. The model with a learning rate of 1e-3, while stable, exhibited slightly lower final performance.

5.1.3 Comparison of Time Costs

The results show that the choice of learning rate did not significantly affect the training time.

5.1.4 Summary

To summarize, selecting an appropriate learning rate is crucial for the performance of deep learning models. The optimal choice of learning rate may vary for different tasks and datasets, so it is advisable to adjust and experiment based on specific circumstances. Furthermore, our results suggest that smaller learning rates may contribute to more stable training processes, especially for complex tasks.

5.2 Interpretation of Results

1. When the learning rate is smaller, the model's training process is more stable but may require more time to achieve optimal performance. This is because a smaller learning rate makes the model's parameter updates more cautious, reducing fluctuations during training but also slowing down the convergence of the model. Therefore, for more complex tasks or situations prone to local minima, a smaller learning rate often better avoids instability during training.

2. Conversely, when the learning rate is larger, the model may converge faster, but it can also increase training instability. A larger learning rate leads to larger parameter updates, which can cause the model to skip the optimal solution and get stuck in local minima. This can sometimes result in training failures or unstable performance.

3. In deep learning, there is sometimes a phenomenon known as the "learning rate sweet spot," where a specific learning rate can balance rapid convergence and stability during training. In this experiment, a learning rate of 1e-2 may be precisely positioned at this sweet spot, enabling a fast and stable training process, thereby achieving optimal performance.

6 Limitations and Future Improvements

6.1 Limitations

1. **Limited Hyperparameter Exploration:** We examined only three different learning rates (1e-1, 1e-2, and 1e-3), but in reality, the performance of deep learning models is influenced by multiple hyperparameters. Future research can expand the range of hyperparameters to gain a deeper understanding of additional influencing factors.

2. **Single Dataset Evaluation:** Our experiments were based on the CIFAR-10 dataset, but different datasets may exhibit varying sensitivities to learning rates. Future work can conduct experiments on multiple datasets to investigate the impact of learning rates on different domains and tasks.

3. **Fixed Network Architecture:** In this experiment, we used a fixed Convolutional Neural Network structure. Future research can explore the influence of learning rates with different network architectures to discover more optimal configurations.

6.2 Future Improvements

1. **Automated Hyperparameter Tuning:** Utilizing automated hyperparameter tuning techniques such as grid search or Bayesian optimization can comprehensively explore various combinations of learning rates to enhance model performance.

2. **Dynamic Learning Rate Scheduling:** In this experiment, we employed a fixed learning rate. Future improvements may consider employing dynamic learning rate scheduling strategies to further enhance model performance.

3. **Ensemble Learning:** Combining multiple models trained under different learning rates can improve model robustness and performance.

4. **Generalization to Other Tasks:** Extending the investigation of learning rate effects to other deep learning tasks such as object detection, segmentation, and natural language processing allows for exploration across a broader range of application domains.

A Pull Request

https://github.com/SustechSTA303/STA303-Assignment01/pull/40