Self-supervised and Supervised Learning on CIFAR10

2019-12172 Jaewon Lee

June 28, 2024

Abstract

This report explores the application of self-supervised and supervised learning techniques on the CIFAR10 dataset. A self-supervised rotation prediction task is employed to pre-train a ResNet18 model, which is then fine-tuned for the CIFAR10 classification task. The performance of the pre-trained model is compared with a model trained from scratch in two scenarios: fine-tuning only the final layer and training the entire network. Results indicate that pre-training with the rotation task improves classification accuracy. Additionally, the benefits of self-supervised pre-training are highlighted in scenarios with limited labeled data. The findings underscore the potential of self-supervised learning in enhancing model performance on downstream tasks.

1 Introduction

The goal of this project is to gain experience with PyTorch and pre-trained models, adapting them for new tasks and losses. Specifically, we use a self-supervised rotation prediction task for pre-training on the CIFAR10 dataset and then fine-tune the model for CIFAR10 classification.

2 Self-Supervised Learning: Rotation Prediction

The self-supervised task involves training a ResNet18 model to predict the rotation of input images. The images from CIFAR10 are rotated by 0, 90, 180, or 270 degrees, and the model is trained to classify these rotations.

2.1 Data Loader

A custom data loader is implemented to generate rotated images and their corresponding labels. The CIFAR10 dataset is used without class labels for this task.

2.2 Model Training

The ResNet18 model, pre-trained on ImageNet, is adapted for the rotation prediction task. The final fully connected layer is modified to output four classes, corresponding to the four possible rotations. The model is trained using cross-entropy loss and the Adam optimizer. This model is saved as 'final_project_1.pth'.

```
# Example code snippet for training the rotation prediction task
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import datasets, transforms, models

# Define data transformations
transform = transforms.Compose([
    transforms.RandomRotation((0, 270)),
    transforms.ToTensor(),
])
```

```
# Load CIFAR10 dataset
trainset = datasets.CIFAR10(root='./data', train=True, download=True, transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=64, shuffle=True, num_workers=2)

# Load pre-trained ResNet18 model
model = models.resnet18(pretrained=True)
model.fc = nn.Linear(512, 4)  # Modify final layer for 4 rotation classes

# Define loss function and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# Training loop
for epoch in range(10):
    for inputs, labels in trainloader:
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
```

3 Fine-Tuning for CIFAR10 Classification

After training the model on the rotation prediction task, we fine-tune it on the CIFAR10 classification task.

3.1 Final Layer Fine-Tuning

The final layers of the model, including the last block of convolutional layers and the fully connected layer, are fine-tuned using CIFAR10 class labels.

2 models are implemented this way. One model is trained with pretrained model from section 2. This model is saved as *'final_project_2_rotation.pth'*.

The other model is implemented without pretrained model. So it initialized with random parameters. This model is saved as *'final_project_2_random.pth'*.

3.2 Full Network Training

The entire network is trained on the CIFAR10 classification task, starting from the weights obtained from the rotation prediction task.

2 models are implemented this way. One model is trained with pretrained model from section 2. This model is saved as *'final_project_3_rotation.pth'*.

The other model is implemented without pretrained model. So it starts training with random parameters. This model is saved as *'final_project_3_random.pth'*.

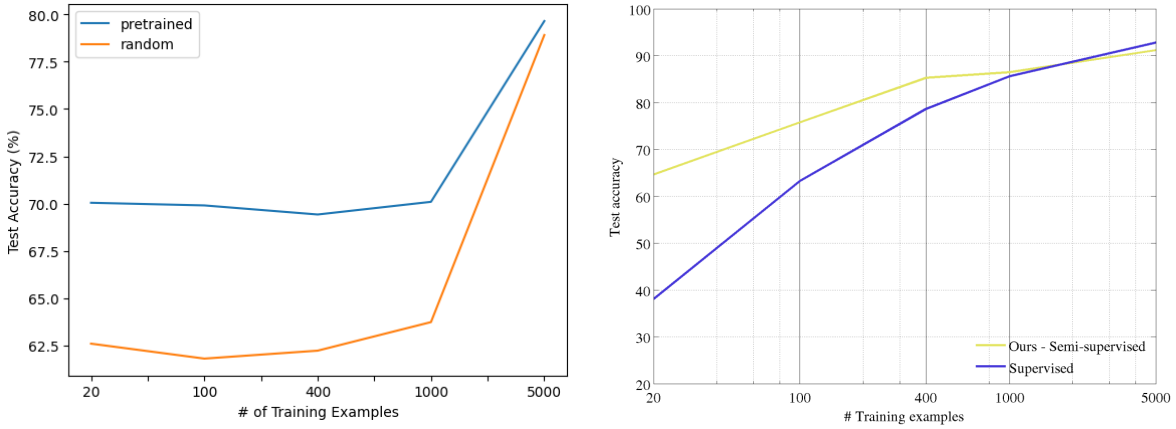
4 Results and Discussion

4.1 Performance Comparison

The performance of the model fine-tuned from the rotation task is compared with the model trained from random initial weights. The comparison shows that pre-training with the rotation task improves the classification accuracy on CIFAR10, in both cases where only the final layer was fine-tuned and the full network was trained. Also, accuracy improved when training full layers compared with training only final layer. All the models were trained with SGD optimizer, momentum=0.9 with adjusted learning rate. This model is trained with data size of 50,000, which is 5,000 training examples.

	Rotation Task	Classification Task			
		Final Layer Fine-Tuning		Full Layer Training	
		pre-trained	non-pretrained	pre-trained	non-pretrained
Final Accuracy(%)	76.78	61.17	40.34	**79.65**	78.91

Table 1: Final Accuracy of trained models. Epochs were set as 45 for rotating tasks, 20 for classification tasks.



(a) My result

(b) Gidaris et al. (2018) Figure 5(b)

Figure 1: Comparison of pretrained(semi-supervised) model and random(supervised) model.

4.2 Extra Credit Task

A plot similar to Figure 5(b) from Gidaris et al. (2018) is generated, showing the CIFAR10 classification performance versus the number of training examples per category. The results demonstrate the advantage of self-supervised pre-training when only a small amount of labeled data is available.

In the original paper, Figure 5(b) is plotted by number of examples with 20, 100, 400, 1,000, 5,000. As there are 10 classes for each data, the size of data is multiplied by 10. In this experiment full layers are trained. The result of training examples with 6,000 was already shown at section 4.1.

In Figure 1a, semi-supervised model is always better like in Figure 1b. However, when there were not enough data, the differences of accuracy are not big as in the original paper. One reason for this is because appropriate hyperparameters were not properly set when I plotted the results. So the effect of increased data was not big enough. Especially, too much epochs(20) were set to my model, which makes supervised model to easily converge.

5 Conclusion

This project demonstrates the effectiveness of self-supervised learning tasks in pre-training feature representations for image classification. The rotation prediction task significantly enhances the performance of the CIFAR10 classification task when fine-tuned on the pre-trained model. Furthermore, fine-tuning on the pre-trained model is effective if the number of data is low.

6 References

- Gidaris, S., Singh, P., & Komodakis, N. (2018). Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*.
- PyTorch Documentation. https://pytorch.org/docs/stable/index.html