Knowledge Distillation on CIFAR-10 with a Sparse CNN and ResNet101

Overview

This project explores the application of knowledge distillation to train a small, sparse CNN model on the CIFAR-10 dataset using ResNet101 as a teacher model. The goal is to evaluate the effectiveness of distillation and analyze how teacher initialization and training strategies impact student model performance.

Step 1: Designing the Sparse Student Model

The student model is a lightweight convolutional neural network:

```
class SmallCNN(nn.Module):
    def __init__(self, num_classes=10):
        super(SmallCNN, self).__init__()
        self.features = nn.Sequential(
            nn.Conv2d(3, 32, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2),

            nn.Conv2d(32, 64, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2),

            nn.Conv2d(64, 128, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.AdaptiveAvgPool2d((1, 1)),
        )
        self.classifier = nn.Linear(128, num_classes)

    def forward(self, x):
        x = self.features(x)
        x = torch.flatten(x, 1)
        return self.classifier(x)
```

Step 2: Dataset and Preprocessing

We use CIFAR-10 as the target dataset with standard data augmentations for training.

```
transform_train = transforms.Compose([
    transforms.RandomHorizontalFlip(),
    transforms.RandomCrop(32, padding=4),
    transforms.ToTensor(),
```

```
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)),
])

transform_test = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)),
])
```

Step 3: Model Initialization

We initialize three teacher/student configurations:

- `teacher` : ResNet101 trained from scratch
- `teacher2` : ResNet101 initialized with ImageNet pre-trained weights
- `student` : Sparse CNN model

Step 4: Teacher Training

Each teacher model was trained on CIFAR-10 for 10 epochs.

- `teacher` : Achieved 79.48% test accuracy
- `teacher2` : Achieved 87.54% test accuracy

Step 5: Knowledge Distillation

We use a combination of soft (KL divergence) and hard (cross entropy) losses to train the student model from the teacher:

```
def distillation_loss(student_outputs, teacher_outputs, targets, T=2.0, alpha=0.5):
    soft_loss = F.kl_div(F.log_softmax(student_outputs / T, dim=1),
                         F.softmax(teacher_outputs / T, dim=1),
                         reduction='batchmean') * (T * T)

    hard_loss = F.cross_entropy(student_outputs, targets)

    return alpha * soft_loss + (1. - alpha) * hard_loss
```

Results:

- Student with `teacher` : **75.98% accuracy**
- Student with `teacher2` : **81.00% accuracy**
- Student trained without distillation: **67.51% accuracy**

Conclusion

1. **Teacher Initialization Matters**: Pre-trained weights significantly improved the performance of the fine-tuned teacher.
2. **Knowledge Distillation Works**: The student trained via distillation outperformed the non-distilled version by over **15%**, demonstrating the value of transferring knowledge from a more capable model.

This project highlights the practical benefits of distillation for compressing models and improving performance with limited model capacity.