ECE371 Neural Networks and Deep Learning Assignment 1

huang yusheng 22308061
School of Electronics and Communication Engineering
Sun Yat-sen University, Shenzhen Campus
huangysh67@mail2.sysu.edu.cn

Abstract: Through learning the completion code, I deepened my understanding and mastery of data processing, model configuration, and model training. When processing the dataset, the images should be enhanced to increase the robustness of the training set and improve the model performance. When configuring the model, you should choose the appropriate loss function and optimizer. Finally, the model is trained, evaluated, and saved.

Keywords: Deep Learning,Data Process,Model Settings

1 Introduction

Supplement the given broken python code and use pytorch to train the classification model for the flower_dataset. First, we need to augment the dataset and divide the training set and validation set, then set up the model, select the appropriate loss function, optimizer and network parameters, and finally train the model to select the model with the best performance.

2 Related Work

The model used in this code is ResNet-18. This model effectively solves the problem of gradient vanishing in deep neural networks by introducing residual learning mechanism. The core innovation lies in the design of Residual Blocks with cross layer connections, allowing the network to directly transfer underlying features through Identity Mapping, making it possible to train networks with up to 18 layers of depth.

ResNet-18 is composed of four stacked residual blocks (each containing two 3x3 convolutional layers), with a total of 18 weight layers (including pooling and fully connected layers). Compared with traditional networks such as VGG, its parameter count is only about 11 million, but it achieved a top-5 error rate of 5.25% on the ImageNet dataset, significantly better than contemporary models [1]. This work provides a paradigm shift for the design of deep learning models, and its proposed residual structure has become a fundamental component of modern convolutional neural networks, widely used in tasks such as image classification and object detection.

3 Method

In the pre-processing process of the image, we define the transformation process for data augmentation and standardization: random flip (horizontal/vertical), random rotation, random color change, random crop scaling, conversion to tensor, standardized operation.

```
# GRADED FUNCTION: Add five data augmentation methods,
# Normalizating and Tranform to tensor
### START SOLUTION HERE ###
transforms.RandomHorizontalFlip(p=0.5),
```

,.

```
    transforms.RandomVerticalFlip(p=0.5),
    transforms.RandomRotation(degrees=45),
    transforms.ColorJitter(brightness=0.2, contrast=0.2,
    saturation=0.2, hue=0.1),
    transforms.RandomResizedCrop(size=224, scale=(0.8, 1.0)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
    std=[0.229, 0.224, 0.225])
    ### END SOLUTION HERE ###
```

For the setup of the fully connected layer network, we need to adjust the number of output features to the number of our classification categories. At the same time, the commonly used cross-entropy loss function is selected for the loss function, the SGD optimizer is selected for the optimizer, and the learning rate is set to 0.001. This completes the basic setup of the model.

```
# GRADED FUNCTION: Modify the last fully connected layer of model
### START SOLUTION HERE ###
# Modify the last fully connected layer of model
num_ftrs = model.fc.in_features
model.fc = nn.Linear(num_ftrs, len(class_names))
### END SOLUTION HERE ###

# GRADED FUNCTION: Define the loss function
### START SOLUTION HERE ###
criterion = nn.CrossEntropyLoss()
### END SOLUTION HERE ###

# GRADED FUNCTION: Define the optimizer
### START SOLUTION HERE ###
optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)
### END SOLUTION HERE ###
```

The backpropagation calculation gradient is performed during training, and the parameters are updated.

```
# GRADED FUNCTION: Backward pass and optimization
### START SOLUTION HERE ###
# Backward pass and optimization
loss.backward()
optimizer.step()
### END SOLUTION HERE ###
```

Save the best model and output the save information.

```
# GRADED FUNCTION: Save the best model
### START SOLUTION HERE ###
torch.save(model.state_dict(),
os.path.join(save_dir, 'best_model.pth'))
print(f'Model saved to {os.path.join(save_dir, "best_model.pth")}')
### END SOLUTION HERE ###
```

4 Experiments

Running after completing code supplementation can cause multi-threaded issues, leading to program errors. The reason is that Windows uses the spawn method to create new processes, requiring multi-process code to be placed in the "if __name__=='__main__': "block. Therefore, it is necessary to modify the code body by placing the entire training code into the main function and adding "if __name__=='__main__':" protection block.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi:10.1109/CVPR.2016.90.