Lab 6 Exercise - Reflections on transfer learning

Amishapriya Singh
Student id: 33333904
as6u21@soton.ac.uk

1. Transfer Learning

1.1. Finetuning

For classification of our boat image dataset, we use a pretrained ResNet50 model from PyTorch model zoo. We tweak this model architecture for our usecase by replacing the last dense layer with a new one that has output dimension same as the number of classes (d=15) in our dataset.
To finetune this pretrained model to our classification task, we do the following:

1. For initial learning of weights and biases of the new layer, we first freeze the weights and biases for all layers except the final dense layer that we replaced. This is done to ensure that we do not overfit the pretrained layers while our model learns the parameters for the final layer. We train the learnable parameters of the final layer for 10 epochs with Adam optimiser with a learning rate of 1e-4. This gives us a **validation loss of 0.82** and **validation accuracy of 0.71**.

2. Once our final dense layer is trained, we unfreeze the weights and biases for all the previous layers and finetune them to our dataset using Adam optimiser with *ReduceLROnPlateau* scheduler. Scheduling allows us to adjust our learning rate based on model performance metrics so preventing the model from overfitting. The scheduler monitors our model validation loss with a patience of 5 epochs and reduces the learning rate by factor of 0.1. We start our learning with a rate of 1e-5. Post training, this model yields a **validation loss of 0.23** and **validation accuracy of 0.92**.

2. Reflect on the two different approaches

For the second method, we use CNN codes generated by ResNet50 as feature vectors and use an SVC for classification. To tune the hyperparameters of our classifier, we run grid search over {'kernel': ('linear', 'rbf'), 'C': [0.1, 1, 10]}. Using the results of the search, we choose the following hyperparameters for our model: C=0.1 and kernel='linear'. We use hinge loss to evaluate the loss of our SVC classifier.
To compare the two models, we use test accuracy and test loss as our metrics. Although the two models give quite similar performance, we find that finetuned ResNet50 just slightly outperforms our SCV model. However, the transfer learning approach takes quite a long time to train and tune. We summarise our findings in Table 1.
Of the two models, SVC on CNN codes is faster to fit. The model took 2.85 seconds to train while finetuned ResNet50 took 6 minutes to train the final dense layer and another 10 minutes to tune the entire model.

Model	Test accuracy	Test loss	Train time
Finetuned ResNet50	0.91	0.33	16m
SVC on CNN codes	0.89	0.33	2.85s

Table 1. Metrics of the two models.