NYCU Introduction to Machine Learning, Final Project

0716225, 洪瑋廷

Part. 1, Environment details (5%):

Python version

Python 3.10.12

Framework

Framework: TensorFlow with Keras API.
Libraries: I use tensorflow.keras modules for building and training the neural network.

Hardware

I was using the Colab free version with T4 GPU for development.
GPU Model: Tesla T4
NVIDIA Driver Version: 535.104.05
CUDA Version: 12.2
Persistence Mode: Off
Compute Mode: Default

I am using the Colab Pro for fine-tuning with the following hardware provided.
GPU Model: Tesla V100-SXM2-16GB
Driver Version: 535.104.05
CUDA Version: 12.2
Persistence Mode: Off
Compute Mode: Default

Part. 2, Implementation details (15%):

Model architecture

Base Model: ResNet50 pre-trained on ImageNet. ResNet50 is a popular convolutional neural network (CNN) known for its deep architecture and residual connections.
Input Shape: The input shape for the images is set to (224, 224, 3), which is standard for ResNet50.
Custom Top Layers:

- After the base ResNet50 model (with its top layers excluded), a GlobalAveragePooling2D layer is added to condense feature maps to a single vector per map.
- This is followed by a fully connected Dense layer with 1024 neurons and ReLU activation.
- The final output layer is another Dense layer with a softmax activation function, having num_classes (200) neurons for multi-class classification.

Hyperparameters

Optimizer: Adam with a learning rate of 0.001 for the first 14 epochs and 0.0001 for 3 epochs of fine-tuning.

Loss Function: Categorical cross-entropy, which is standard for multi-class classification tasks.
Metrics: Accuracy, Precision, and Recall. These metrics provide insights into the model's performance in terms of overall accuracy and class-wise prediction accuracy. The same set of metrics are also used for validation.

Training strategy
Training Execution: The model is trained for 14 and fine-tuning 3 epochs, with the number of steps per epoch determined by the size of the training dataset and the batch size (256).

Firstly, I increased the batch size to meet the RAM size to better utilize the available RAM. Larger batch sizes can lead to faster training because they allow for more efficient utilization of the parallel processing capabilities of GPUs.

Secondly, I decreased the learning rate. It can lead to more stable and precise convergence to the minimum of the loss function.

Also, I increased the epoch size. Increasing the number of epochs allows the model more iterations to learn from the data, potentially leading to better performance.

Lastly, I examined the difference between the accuracy, precision, and recall rate of the training and the validation. Finally, I decided to split the training into the first 14 epochs and 3 fine-tuning epochs respectively.

Part. 3, Experimental results (15%):

Evaluation metrics
The model's performance is tracked using several metrics: loss, accuracy, precision, and recall. Here is the evaluation based on the final epoch (Epoch 17/17):

Loss: The model's loss is 0.0929.

Accuracy: An accuracy of 98.92% implies that the model correctly classifies a high percentage of the input samples.

Precision: A precision of 99.45% indicates that when the model predicts a class correctly.

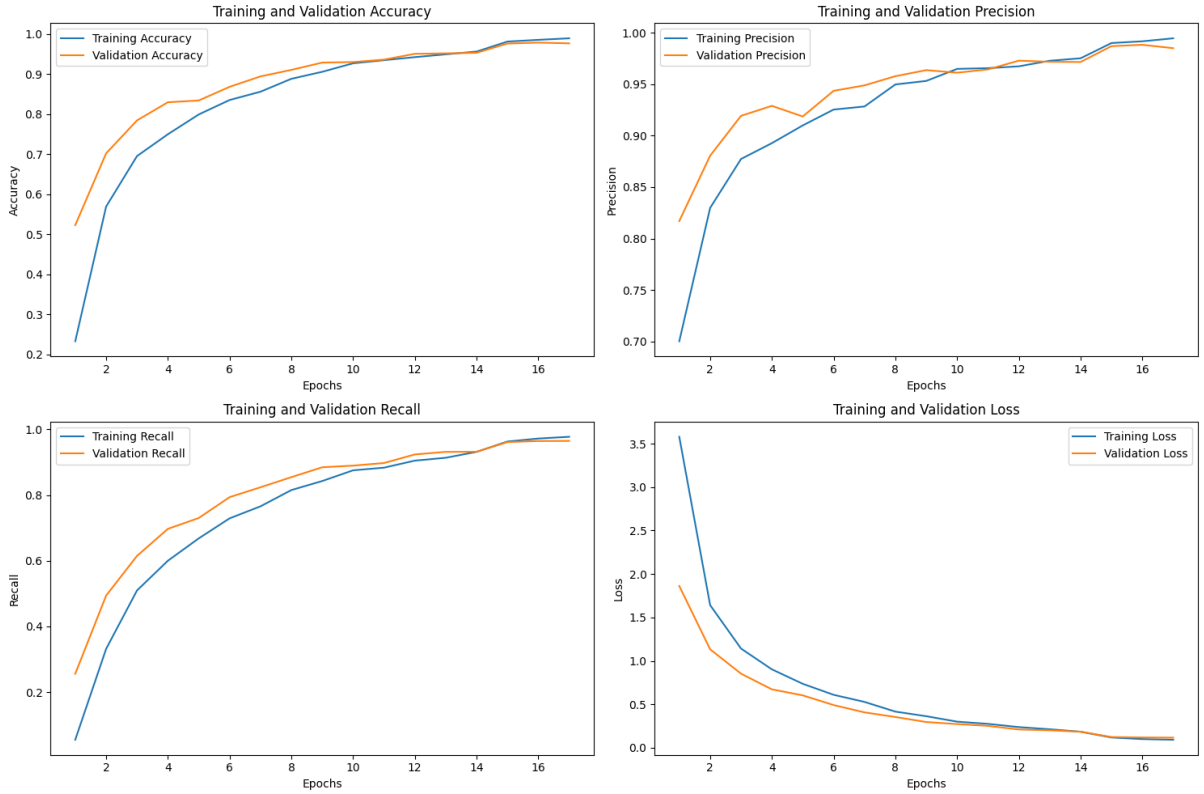
Recall: A recall of 97.73% implies that the model correctly identifies 97.73% of all relevant cases.

In terms of validation, it shows the same level of accuracy, precision, and recall rate.
val_accuracy: 0.9766 - val_precision: 0.9849 - val_recall: 0.9648
This indicates that the model has a good capacity to generalize its ability to unseen datasets.

Learning curve

The loss consistently decreases over epochs, which is a positive indicator that the model is learning effectively.

Accuracy, precision, and recall all increase as training progresses on the training and validation dataset, suggesting that the model is improving its ability to classify correctly even for the case of unseen data.



Ablation Study

Batch Size Ablation: I tried different batch sizes (32, 64, 128, 256) and chose 256 and 512 based on the outcomes and available GPU RAM size.

Epoch and Learning Rate Ablation: I experimented with various combinations of epochs and learning rates.

To be specific,

- Trial 1 (Batch Size 32): Started with a smaller batch size.
- Trial 2 (Batch Size 64): Increase the batch size to improve the computational efficiency and stability of gradient updates.
- Trial 3 (Batch Size 128)
- Trial 4 (Batch Size 256)
- Trial 5 (10 Epochs)
- Trial 6 (20 Epochs)
- Trial 7 (20 Epochs, Learning Rate 0.001): Adjusting the learning rate to see if a faster learning rate impacts model convergence over the same number of epochs.
- Trial 8 (20 Epochs, Learning Rate 0.0005): Lower learning rate for potentially finer convergence.
- Trial 9 (30 Epochs)

- Trial 10 (Learning Rate 0.0001)
- Trial 11 (30 Epochs, Learning Rate 0.0005)
- Trial 12 (50 Epochs, Learning Rate 0.0005)
- Trial 13 (14 Epochs, Learning Rate 0.001 + fine-tuning 3 Epochs, Learning Rate 0.0001)