Image Classification model

1. **Dataset**

   I noticed that in the given dataset, there is an imbalance between the classes. We have 108 images of ‘roads’, but we only have 45 images of ‘fields’. With a small dataset like this, the model would certainly be biased towards the class ‘roads’, which would cause our model to perform badly. Therefore I have added 35 new images into the class ‘fields’. These images are taken from Pexels, a free stock photo website. The final dataset that I trained the model with has 80 images of ‘fields’ and 108 images of ‘roads’. You can find the dataset here.

2. **Model**

   Since we have a limited dataset and a simple 2 classes image classification task, I thought it would be best to perform transfer learning based on a pre-trained model. For the pre-trained model, I have chosen to use the MobileNetV2 model that was trained on ImageNet. MobileNetV2 is designed to be efficient and can work well with small datasets, which is a perfect choice for my case, since I do not have a GPU and our dataset is very small. While it's true that other pre-trained models, such as ResNet and VGG, have superior accuracy in complex image classification tasks, their overkill complexity is not needed for our uncomplicated task and limited resources. Notably, the MobileNetV2 architecture can still deliver commendable performance within our context

   For the transfer learning part, I froze the convolutional layer of MobileNetV2, and added a classifier on top and trained this top layer classifier.

3. **Hyperparameters**
   - Learning rate: 0,0001
   - Epochs: 10
     - With a dataset this size, 10 epochs are enough for the model to converge, and also since we are doing transfer learning, the model converges faster.

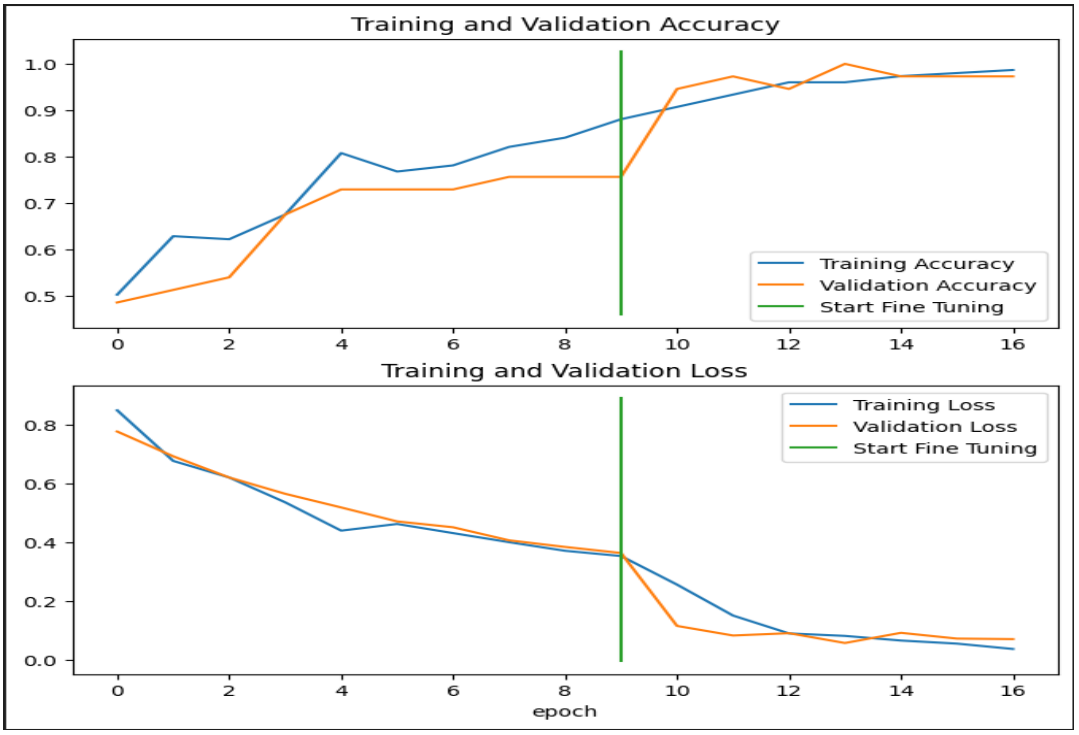
4. **Results**

After training, we have:

- training_loss : 0.3529
- training_accuracy : 0.8808
- validation_loss : 0.3632
- validation_accuracy : 0.7568

Here are the results after fine-tuning the model for another 10 epochs (however *EarlyStopping* stops the process after epoch 16 to prevent overfitting):

- training_loss : 0.0375
- training_accuracy : 0.9868
- validation_loss : 0.0709
- validation_accuracy : 0.9730



Here are the interferences of test_images and some new data with this model: