CS 171: Introduction to Machine Learning, Section 1

Instructor: Fabio Di Troia

Final Project

Chee Jun Wong (013844545)

Introduction

In these days of modernization and industrialization, it's critical to quantify how human activities like logging, mining, and agriculture affect our protected natural areas in order to determine the impact of climate change on Earth's flora and wildlife. As a result, VIGIA project was developed by Mexican researchers with the goal of creating an automated surveillance system for protected areas. The ability to distinguish plants within protected zones is the first step in such an endeavor. Therefore, it is true to say that this project is an image classification problem.

For this particular final project, I have downloaded the dataset from Kaggle in order to identify a specific type of cactus in aerial imagery. The file contains a *csv* file and all the *jpg* images. Thus, I am going to perform image classification using machine learning method based on the image files given. In other words, I am going to build a classifier capable of determining whether an aerial image contains a columnar cactus or not.

Data preprocessing

Before we can feed our data to a model, it must first be converted into a format that the model can understand. This dataset is a modified version of the cactus aerial image dataset as Kaggle has resized each image to 32 x 32 pixels.

Figure 1. Example image with cactus.

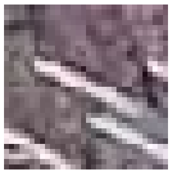

Figure 2. Example image with cactus.

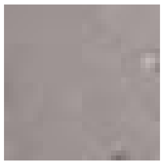
Figure 3. Example image without cactus.

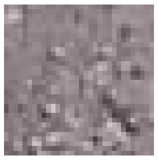
Figure 4. Example image without cactus.

On top of that, I have split the data into training, validation and test sets. During the split, I used 80 percent of the samples for training and the remaining 10 percent for validation and 10 percent for testing. I then created respective *tf.data.Dataset* objects for each training, validation and test set. The dataset, however, still just contains the image filenames, not the actual images. As a result, I created a function that can read files and conduct any necessary preparation. Nonetheless, I shuffled and batch-processed the datasets.

<u>**Tuning**</u>

Transfer learning is the method of choice for this project. This is because of transfer learning allows me to reuse previously trained image classification models and just retrain the top layer of the network that decides which classes an image can belong to, which speeds up training. In the project, I have used a pre-trained MobileNetV2 model as the feature detector. Google introduced MobileNetV2 as the second iteration of MobileNet, which is a convolutional neural network architecture, with the goal of being smaller and lighter than ResNet and Inception for use on mobile devices. I have loaded the MobileNetV2 model pre-trained on ImageNet without the top layer, frozen its weights, and added a new classification head using Tensorflow's Keras.

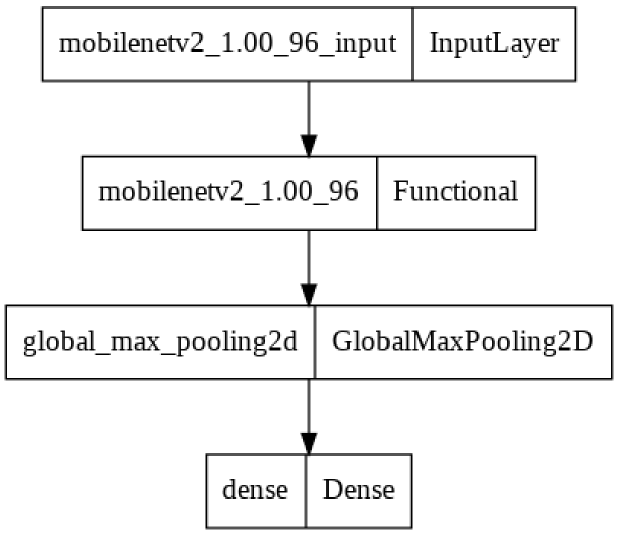

Figure 5. The model I used in this project.

Since I used transfer learning in this model, hyperparameter optimization is not necessary as the MobileNetV2 has been trained on the ImageNet dataset. However, I then fine-tuned the model in order to improve the accuracy of the model. I only trained the new classification head of the model, freezing the weights from MobileNetV2. Because the new classification head was randomly initialized, the model would "forget" all of the knowledge it started with if I didn't freeze those weights at the start.

Experiments

Experiment on base model

At first, I trained the model with only 30 epochs before any fine tuning of the model. Below are the results I have got with the model.

```
Epoch 30/30
437/437 [==============================] - 17s 38ms/step - loss: 0.0178 -
accuracy: 0.9945 - val_loss: 0.0134 - val_accuracy: 0.9953
```

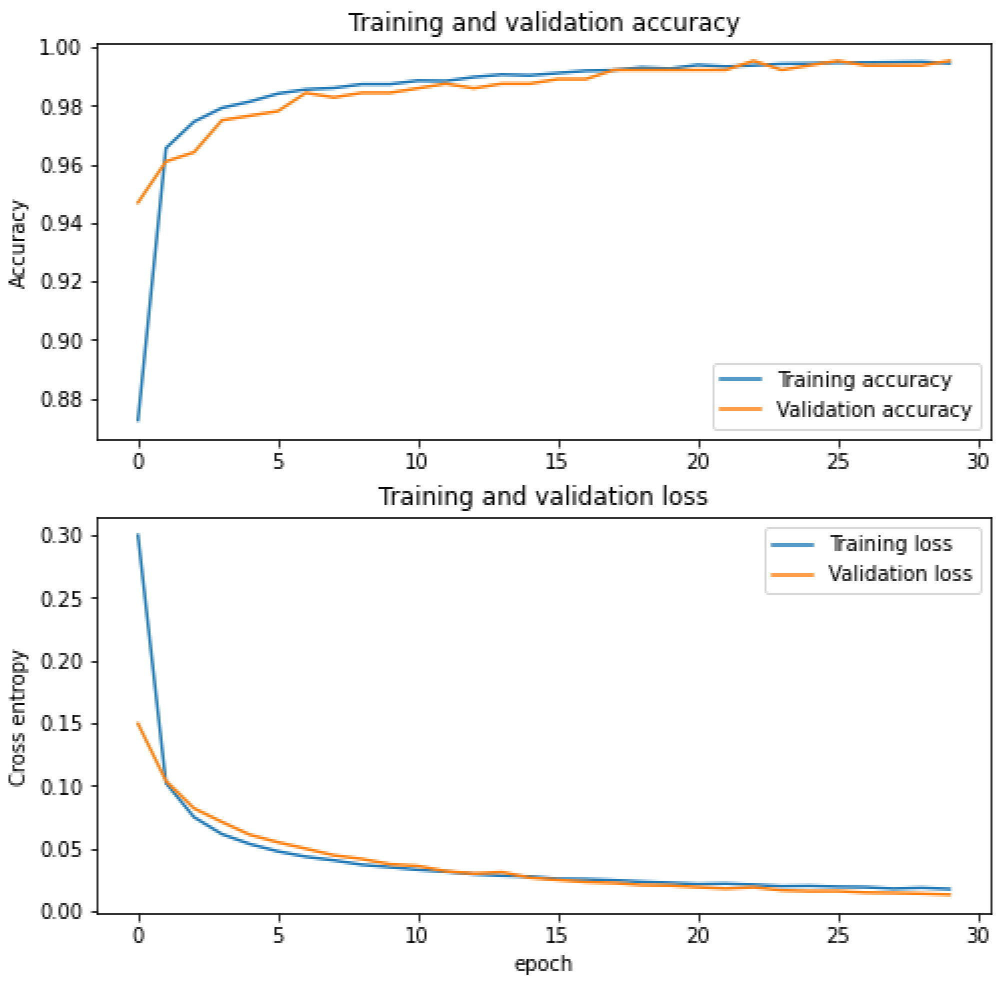

Figure 6. The top part is the graph of training and validation accuracy between accuracy against epoch. The bottom part is the graph of training and validation loss between cross entropy against epoch.

From the figure above, the validation accuracy of the model increases from 0.9469 to 0.9953 after 30 epochs.

```
Evaluation of base model on test data
274/274 [==============================] - 10s 26ms/step - loss: 0.0256 -
accuracy: 0.9920
Test loss, test accuracy: [0.0255572646856308, 0.9919999837875366]
```

From the result of test data, it is true to say that the test accuracy is quite high.

<u>Experiment on fine-tuned model</u>

```
Epoch 60/60
437/437 [==============================] - 21s 48ms/step - loss: 0.0016 -
accuracy: 0.9993 - val_loss: 2.3443e-05 - val_accuracy: 1.0000
```

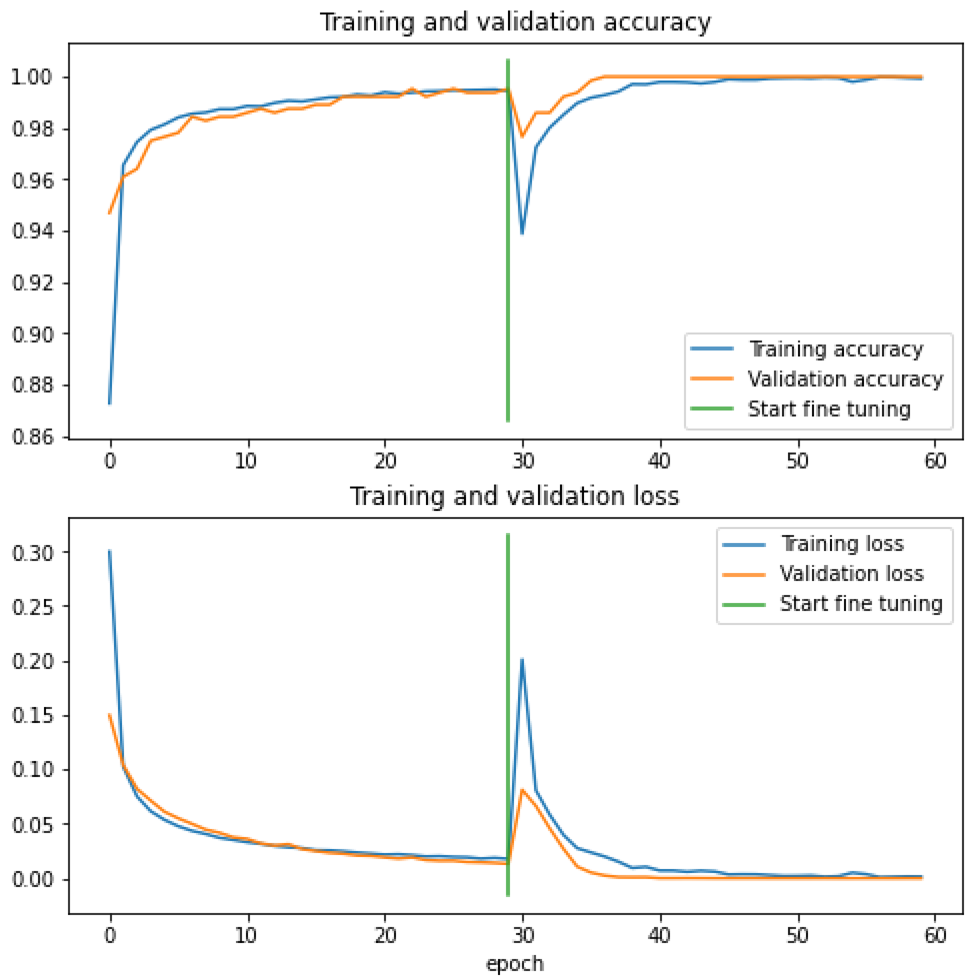

Figure 7. The top part is the graph of training and validation accuracy between accuracy against epoch. The bottom part is the graph of training and validation loss between cross entropy against epoch.

From the figure above, it is obvious that the validation accuracy of the fine-tuned model increases from 0.9953 to 1.0000 after another 30 epochs.

```
Evaluation of fine-tuned model on test data
274/274 [==============================] - 10s 25ms/step - loss: 0.0144 -
accuracy: 0.9960
Test loss, test accuracy: [0.014437632635235786, 0.9959999918937683]
```

Model	Test accuracy	Test loss	Validation accuracy	Validation loss
Base model	0.9920	0.0256	0.9953	0.0134
Fine-tuned model	0.9960	0.0144	1.0000	2.3443e-05

From the result of test data, the test accuracy has increased to 0.9960 from 0.9920. Also, the test loss has dropped from 0.0256 to 0.0144. Therefore, more epochs may result in even higher improvements, based on the accuracy and loss graphs.

Conclusions

In the nutshell, from the obtained results, all the validation accuracies are quite close to the train accuracies. It is true to say that the overfitting of the model is prevented. I can say the fine-tuned model gave the best result, which is 0.9960 in test accuracy. Apart from that, it is also true that transfer learning has saved me a lot of time as I don't have to build a new model from scratch and spent a lot of time just training the model due to scarcity of time.

Hence, for future work, if the time permits, I plan to build a model from scratch with a hyperparameter optimizer such as Optuna.

Reference

1. Efren López-Jiménez, Juan Irving Vasquez-Gomez, Miguel Angel Sanchez-Acevedo, Juan Carlos Herrera-Lozada, Abril Valeria Uriarte-Arcia, Columnar Cactus Recognition in Aerial Images using a Deep Learning Approach. Ecological Informatics. 2019.
2. TensorFlow.org. https://www.tensorflow.org/alpha/tutorials/images/transfer_learning. Transfer Learning Using Pretrained ConvNets. 2021.
3. Mark Sandler, Andrew Howard. Google Research. https://ai.googleblog.com/2018/04/mobilenetv2-next-generation-of-on.html. MobileNetV2: The Next Generation of On-Device Computer Vision Networks. 2018.