2022 Machine Learning Final Project
Group 46

Data Process

資料處理方式將 images 資料夾的圖片做 720x1280 的 resize，images_real_world 資料夾的圖片做 1080x1920 的 resize，本組有嘗試使用 Data augmentation 中的 Random Flip 方法，將 images 和 images_real_world 的圖片在分別的 data loader 中加入圖片訓練時期的 Random Flip 方法，但由於效果不佳（Loss 收斂速度緩慢），因此資料前處理，只使用 resize 的方法。

Network Design

Network 架構為 input size = 256*256，channel = 13、depth = 7，output size = 1080*1920，Backbone 為 Conv2d+BN+Relu。其中，架構中的 BN 層有共用參數 b（BN 後加上一個須被訓練的參數 b），另外 network 中也有類似 shortcut 的結構，會將 model 前面 backbone 的 output 作為 model 後面 backbone 的 input。

```
inputs = tf.placeholder(tf.float32,shape=(None,None, None, 3))
y_ = tf.placeholder(tf.float32, [None,None, None,6]) # 6 class
x = tf.image.resize_images(inputs, (256, 256))
x = x/255.0
y = tf.image.resize_images(y_, (256, 256))
ch=13
depth=7 # 5
xn = []
b=tf.Variable(0.0)
x = tf.layers.conv2d(x,ch,3,1,'same')
x = tf.layers.batch_normalization(x)
x = tf.nn.relu(x)
for i in range(depth):
    xn.append(x)
    x = tf.layers.conv2d(x,ch*(2**(i+1)),3,1,'same')
    x = tf.layers.batch_normalization(x,center=False,scale=False)+b
    x = tf.nn.relu(x)
    x = tf.layers.conv2d(x,ch*(2**(i+1)),3,1,'same')
    x = tf.layers.batch_normalization(x,center=False,scale=False)+b
    x = tf.nn.relu(x)
    if i <depth-1:
        x = tf.nn.avg_pool(x,[1,2,2,1],[1,2,2,1],'SAME')
for i in range(depth):
    if i>0:
        x = tf.keras.layers.UpSampling2D((2,2))(x)
    x = tf.layers.conv2d(x,ch*(2**(depth-i-1)),3,1,'same')+xn[-i-1]
    x = tf.layers.batch_normalization(x,center=False,scale=False)+b
    x = tf.nn.relu(x)
out = tf.layers.conv2d(x,6,3,1,'same')
outputs = out
outputs = tf.image.resize_images(outputs, (1080, 1920))
```

Optimizer, Hyperparameters and Training Method

優化器使用 Adam，初始 learning rate 為 0.0005，另外有嘗試使用 SGD+momentum 優化器，但效果不佳，因此維持使用 Adam，Loss function 使用 cross entropy，batch_size 為 25，epochs 數約為 80，總參數數量為 66,458,802，FLOPs 為 132,901,067。每 Train 300 個 batch 的 images，則 train images_real_world，然後儲存 check point。

Post-training Quantization

比較 dynamic range quantization、float16 quantization、float32 quantization。發現 float16 的 mIOU 表現最好，但 latency 也是最高的，模型大小則是 126.8MB。dynamic range 雖然 mIOU 降低 20%，但同時 latency 也減少 30%，模型大小是 63.4MB，最後選擇使用 mIOU 較高的 float16 quantization。

```
converter = tf.lite.TFLiteConverter.from_frozen_graph(
        graph_def_file = '/content/drive/Shareddrives/ml2022_share_data/lab3_model.pb',
        input_arrays = ['Placeholder'],
        input_shapes = {'Placeholder':[1, 1080, 1920, 3]},
        output_arrays = ['ArgMax']
)

converter.optimizations = [tf.lite.Optimize.DEFAULT]
converter.target_spec.supported_types = [tf.float16]

tflite_quant_model = converter.convert()
```

Result

Model 最後在聯發科晶片的執行結果數據如下，MIOU 為 0.807，Latency 為 670048 (ms)，power 為 1142.94 (mA)。

```
*** avg latency: 670048 ***
/sdcard/Android/data/com.mediatek.simpleP...ulled. 27.3 MB/s (201657 bytes in 0.007s)
*** avg power: 1142.94 ***

*** calculating mIOU...
mIOU progress: 130/130
*** Mean mean Iou: 0.807 ***
*** overall results [ latency: 670048, power: 1142.939, mIOU: 0.807 ] ***
gropu number:  46
```

Work Distribution Chart

吳啓玄: model training
張繼哲: model training
詹前駒: post-training quantization
陳允欽: research and development