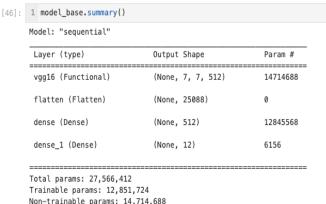
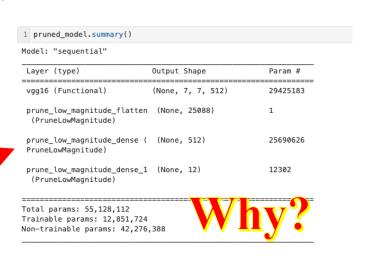
EEP 596 Quiz 2 - Week 3, Apr 10 (Wed.) Spring 2024

1. How does the number of parameters change when pruning is applied to a neural network, and what is the reason for this change?



Non-trainable params: 14,714,688

```
1 import tensorflow as tf
 2 import tensorflow_model_optimization as tfmot
 3 import numpy as np
 5 # Load the original pre-trained model
 6 model = tf.keras.models.load_model('vgg16_plant_leaves.h5')
 8 # Define the pruning parameters
 9 pruning_schedule = tfmot.sparsity.keras.PolynomialDecay(initial_sparsity=0.5,
                                                           final_sparsity=0.75,
10
11
                                                           begin_step=0,
12
                                                           end_step=len(train_ds)*5)
13 pruning_params = {'pruning_schedule': pruning_schedule}
15 # Apply pruning to the whole model
16 prune_low_magnitude = tfmot.sparsity.keras.prune_low_magnitude
17 pruned_model = prune_low_magnitude(model, **pruning_params)
19 # Compile the pruned model
20 pruned_model.compile(optimizer='adam',
21
                        loss='sparse_categorical_crossentropy',
22
                        metrics=['accuracy'])
23
24 # Fit the pruned model
25 pruning_callbacks = [tfmot.sparsity.keras.UpdatePruningStep()]
26 pruned_model.fit(train_ds, epochs=5, validation_data=test_ds, callbacks=pruning_callbacks)
28 # After training, remove the pruning wrappers and use this for quantization-aware training
29 final_model = tfmot.sparsity.keras.strip_pruning(pruned_model)
```



<pre>1 final_model.summary()</pre>		
Model: "sequential"		
Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 7, 7, 512)	14714688
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 512)	12845568
dense_1 (Dense)	(None, 12)	6156
Total params: 27,566,412 Trainable params: 12,851,72	4	

Non-trainable params: 14.714.688

```
metrics=['accuracy'])
 8 # Evaluate the pruned model
 9 pruned_accuracy = final_model.evaluate(test_ds)[1]
10 print(f'Pruned Model Test Accuracy: {pruned_accuracy}')
11
12 # After pruning and fine-tuning, convert the model to TensorFlow Lite
   converter = tf.lite.TFLiteConverter.from keras_model(final_model)
14 tflite_model = converter.convert()
16 # Save the TensorFlow Lite model
17 with open('vgg16_plant_leaves_pruned.tflite', 'wb') as f:
18
    f.write(tflite_model)
19
20 # Calculate the size of the TensorFlow Lite model
21 tflite_model_size = os.path.getsize('vgg16_plant_leaves_pruned.tflite') / (1024 * 1024)
22 print(f'Pruned Model Size (in TFLite format): {tflite_model_size:.2f} MB')
23
```

Pruned Model Size (in TFLite format): 105.17 MB

```
1 import os
2 import tensorflow as tf
 4 # Compile the stripped model before evaluation
 5 final_model.compile(optimizer='adam',
                      loss='sparse_categorical_crossentropy',
                      metrics=['accuracy'])
9 # Evaluate the pruned model
10 pruned_accuracy = final_model.evaluate(test_ds)[1]
11 print(f'Pruned Model Test Accuracy: {pruned_accuracy}')
12
13 # After pruning and fine-tuning, convert the model to TensorFlow Lite
   converter = tf.lite.TFLiteConverter.from keras model(final model)
15 converter.optimizations = [tf.lite.Optimize.EXPERIMENTAL_SPARSITY] # Enable sparsity optimization
17 # Convert the model
18 tflite_model = converter.convert()
19
20 # Save the TensorFlow Lite model
21 tflite_model_path = 'vgg16_plant_leaves_pruned.tflite'
22 with open(tflite_model_path, 'wb') as f:
23
      f.write(tflite_model)
25 # Calculate the size of the TensorFlow Lite model
26 tflite_model_size = os.path.getsize(tflite_model_path) / (1024 * 1024)
27 print(f'Pruned Model Size (in TFLite format): {tflite_model_size:.2f} MB')
```

Pruned Model Test Accuracy: 0.945555567741394

Pruned Model Size (in TFLite format): 38.97 MB

2. Which quantization methods does TensorFlow Lite support?

TensorFlow Lite supports several quantization methods, including post-training quantization (Dynamic range quantization, full integer quantization, float16 quantization) and quantization-aware training.

3. Does TensorFlow Lite support structured or unstructured pruning, and what is the key difference between these methods?

TensorFlow Lite primarily supports unstructured pruning, which involves individually removing weights based on their magnitude or importance, as opposed to structured pruning, which removes entire filters or channels.

- 4. What distinguishes feature-based knowledge distillation from relation-based knowledge distillation?
 - Feature-based knowledge distillation focuses on transferring knowledge by matching features (such as intermediate layer activations) between the teacher and student networks.
 - In contrast, relation-based knowledge distillation aims to transfer relational knowledge by aligning the relationships (e.g., pairwise distances Euclidean distance or angles between data points Cosine Similarity) between the layers or outputs of the teacher and student networks.

