Optimization and Quantization

Minimizing compression loss



Laurence Moroney, Google





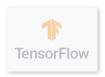
Train a model

Convert model

Optimize model

Deploy model at inferences at Edge

Make inferences at Edge





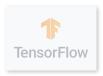
Train a model

Convert model

Optimize model

Deploy model at Edge

inferences at Edge



TensorFlow Lite

Train a model

Convert model

Optimize model

Deploy model at model inferences at Edge

at Edge





Train a model

Convert Make inferences at Edge

Convert Make inferences at Edge





Train a model

Convert
model

Optimize
model

Deploy
model at
Edge

Make
inferences
at Edge





Train a model

Convert model

Optimize model

Deploy Make inferences at Edge

at Edge





Train a model

Convert model

Optimize model at inferences at Edge

Deploy Make inferences at Edge

Converter Optimization

```
converter = tf.lite.TFLiteConverter.from_saved_model(CATS_VS_DOGS_SAVED_MODEL)

tflite_model = converter.convert()

tflite_model_file = 'converted_model.tflite'
```

```
with open(tflite_model_file, "wb") as f:
    f.write(tflite_model)
```

```
converter = tf.lite.TFLiteConverter.from_saved_model(CATS_VS_DOGS_SAVED_MODEL)

converter.optimizations = [tf.lite.Optimize.DEFAULT]

tflite_model = converter.convert()

tflite_model_file = 'converted_model.tflite'
```

```
with open(tflite_model_file, "wb") as f:
    f.write(tflite_model)
```

```
converter = tf.lite.TFLiteConverter.from_saved_model(CATS_VS_DOGS_SAVED_MODEL)

converter.optimizations = [tf.lite.Optimize.DEFAULT]

tflite_model = converter.convert()

tflite_model_file = 'converted_model.tflite'
```

```
converter = tf.lite.TFLiteConverter.from_saved_model(CATS_VS_DOGS_SAVED_MODEL)

converter.optimizations = [tf.lite.Optimize.DEFAULT]

tflite_model = converter.convert()

tflite_model_file = 'converted_model.tflite'

with open(tflite_model_file, "wb") as f:
```

```
rith open(tflite_model_file, "wb") as f:
f.write(tflite_model)
```

```
[tf.lite.Optimize.DEFAULT]
[tf.lite.Optimize.OPTIMIZE_FOR_SIZE]
```

[tf.lite.Optimize.OPTIMIZE_FOR_LATENCY]

```
converter = tf.lite.TFLiteConverter.from_saved_model(CATS_VS_DOGS_SAVED_MODEL)

converter.optimizations = [tf.lite.Optimize.DEFAULT]

tflite_model = converter.convert()

tflite_model_file = 'converted_model.tflite'

with open(tflite_model_file, "wb") as f:
```

```
[tf.lite.Optimize.DEFAULT]
```

f.write(tflite_model)

[tf.lite.Optimize.OPTIMIZE_FOR_SIZE]

```
[tf.lite.Optimize.OPTIMIZE_FOR_LATENCY]
```

```
converter = tf.lite.TFLiteConverter.from_saved_model(CATS_VS_DOGS_SAVED_MODEL)

converter.optimizations = [tf.lite.Optimize.DEFAULT]

tflite_model = converter.convert()

tflite_model_file = 'converted_model.tflite'
```

```
with open(tflite_model_file, "wb") as f:
    f.write(tflite_model)
```

```
[tf.lite.Optimize.DEFAULT]
[tf.lite.Optimize.OPTIMIZE_FOR_SIZE]
```

[tf.lite.Optimize.OPTIMIZE_FOR_LATENCY]

Representative Data

```
converter = tf.lite.TFLiteConverter.from_saved_model(CATS_VS_DOGS_SAVED_MODEL)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
tflite_model = converter.convert()
tflite_model_file = 'converted_model.tflite'
with open(tflite_model_file, "wb") as f:
```

```
converter = tf.lite.TFLiteConverter.from_saved_model(CATS_VS_DOGS_SAVED_MODEL)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
def representative_data_gen():
    for input_value, _ in test_batches.take(100):
        yield [input_value]
converter.representative_dataset = representative_data_gen
converter.target_spec.supported_ops = [tf.lite.OpsSet.TFLITE_BUILTINS_INT8]
tflite_model = converter.convert()
tflite_model_file = 'converted_model.tflite'
```

```
converter = tf.lite.TFLiteConverter.from_saved_model(CATS_VS_DOGS_SAVED_MODEL)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
def representative_data_gen():
    for input_value, _ in test_batches.take(100):
        yield [input_value]
converter.representative_dataset = representative_data_gen
converter.target_spec.supported_ops = [tf.lite.OpsSet.TFLITE_BUILTINS_INT8]
tflite_model = converter.convert()
tflite_model_file = 'converted_model.tflite'
```

```
converter.optimizations = [tf.lite.Optimize.DEFAULT]
def representative_data_gen():
    for input_value, _ in test_batches.take(100):
        yield [input_value]
converter.representative_dataset = representative_data_gen
converter.target_spec.supported_ops = [tf.lite.OpsSet.TFLITE_BUILTINS_INT8]
tflite_model = converter.convert()
```

tflite_model_file = 'converted_model.tflite'

with open(tflite_model_file, "wb") as f:

f.write(tflite_model)

converter = tf.lite.TFLiteConverter.from_saved_model(CATS_VS_DOGS_SAVED_MODEL)

```
converter.optimizations = [tf.lite.Optimize.DEFAULT]
def representative_data_gen():
    for input_value, _ in test_batches.take(100):
        yield [input_value]
converter.representative_dataset = representative_data_gen
converter.target_spec.supported_ops = [tf.lite.OpsSet.TFLITE_BUILTINS_INT8]
tflite_model = converter.convert()
tflite_model_file = 'converted_model.tflite'
```

f.write(tflite_model)

converter = tf.lite.TFLiteConverter.from_saved_model(CATS_VS_DOGS_SAVED_MODEL)

```
converter = tf.lite.TFLiteConverter.from_saved_model(CATS_VS_DOGS_SAVED_MODEL)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
def representative_data_gen():
    for input_value, _ in test_batches.take(100):
        yield [input_value]
converter.representative_dataset = representative_data_gen
converter.target_spec.supported_ops = [tf.lite.OpsSet.TFLITE_BUILTINS_INT8]
tflite_model = converter.convert()
tflite_model_file = 'converted_model.tflite'
with open(tflite_model_file, "wb") as f:
    f.write(tflite_model)
```

Your turn!