Ambient AI Bootcamp

Practice 5

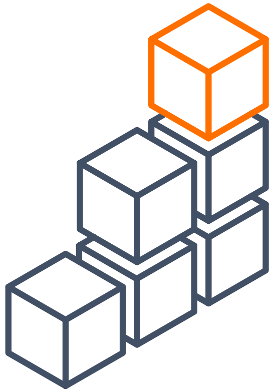
SNU Graduate School of Data Science

Table of Contents

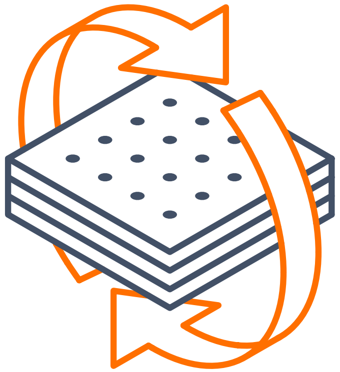
5-1. Introduction to TensorFlow Lite

TensorFlow Lite

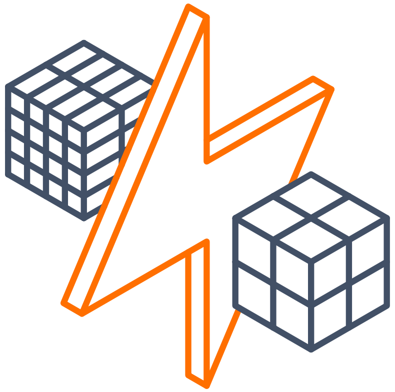
Library for deploying models on mobile, microcontrollers, and other edge devices



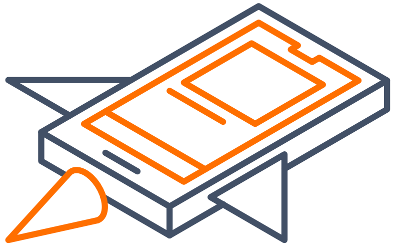
1. Build a model



2. Convert



3. Optimize



4. Deploy

TensorFlow Lite

Optimized for five core constraints

1. **Latency**
   - No round-trip to a server
2. **Privacy**
   - No personal data leaves the device
3. **Connectivity**
   - Internet connection not required
4. **Size**
   - Reduced model size, smaller download size
5. **Power consumption**
   - Efficient inference

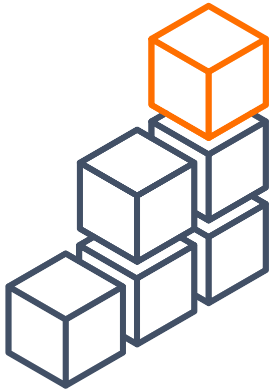
Tensorflow Lite: Getting Started

First, download and normalize the fashion MNIST dataset

```
import tensorflow as tf
import numpy as np

# Load MNIST dataset
fashion_mnist = tf.keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) =
fashion_mnist.load_data()

# Normalize the input image
train_images = train_images.astype(np.float32) / 255.0
test_images = test_images.astype(np.float32) / 255.0
```

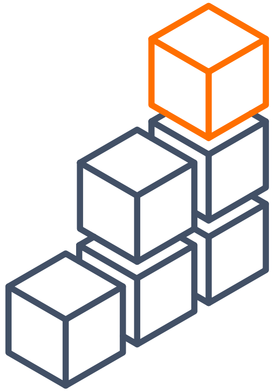


1. Build a model

Tensorflow Lite: Getting Started

Next, define the model architecture

```
model = Sequential([
    InputLayer(input_shape=(28, 28)),
    Reshape(target_shape=(28, 28, 1)),
    Conv2D(filters=16, kernel_size=3, padding='same', activation='relu'),
    MaxPool2D(pool_size=(2,2), strides=(2,2)),
    Conv2D(filters=32, kernel_size=3, padding='same', activation='relu'),
    MaxPool2D(pool_size=(2,2), strides=(2,2)),
    Flatten(),
    Dense(10, activation='softmax')
])
```



1. Build a model

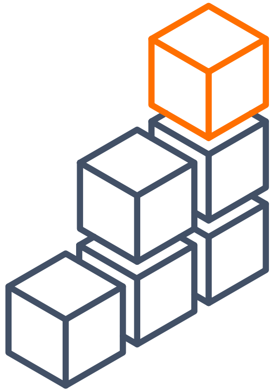
Tensorflow Lite: Getting Started

Next, train/optimize the model

- We can also add ***quantization aware training*** in this step

```
model.compile(optimizer='adam',
              loss=SparseCategoricalCrossentropy(
                  from_logits=False),
              metrics=['accuracy'])
model.fit(
  train_images,train_labels, epochs=10,
  validation_data=(test_images, test_labels)
)

metrics = model.evaluate(test_images, test_labels)
```



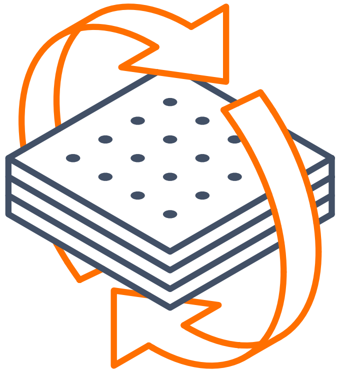
1. Build a model

```
Model validation loss: 0.254 | validation accuracy: 90.99%
```

Tensorflow Lite: Getting Started

To convert the model to TFLite, initialize a ***converter***

```
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()
```



2. Convert

We can now save the tflite model and deploy it on mobile!

```
with open('model.tflite', 'wb') as f:
    f.write(tflite_model)
```

Tensorflow Lite: Getting Started

Next, we have several options available to *optimize* the model

- Typically, we use [*Tensorflow Model Optimization Toolkit*]

Two methods:

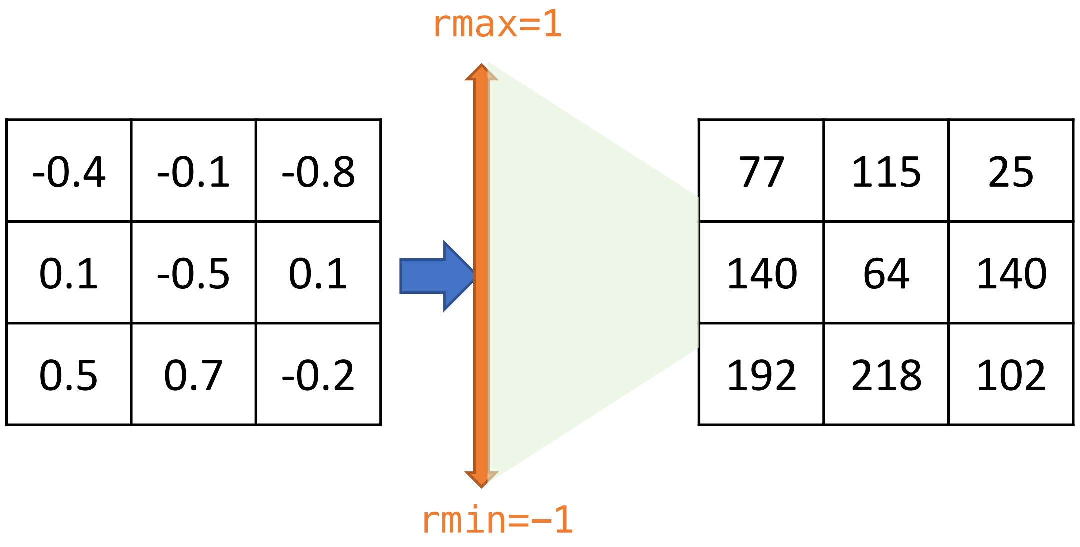
- Quantization
  - Post-Training Quantization (PTQ)
  - Quantization-Aware Training (QAT)
- Pruning

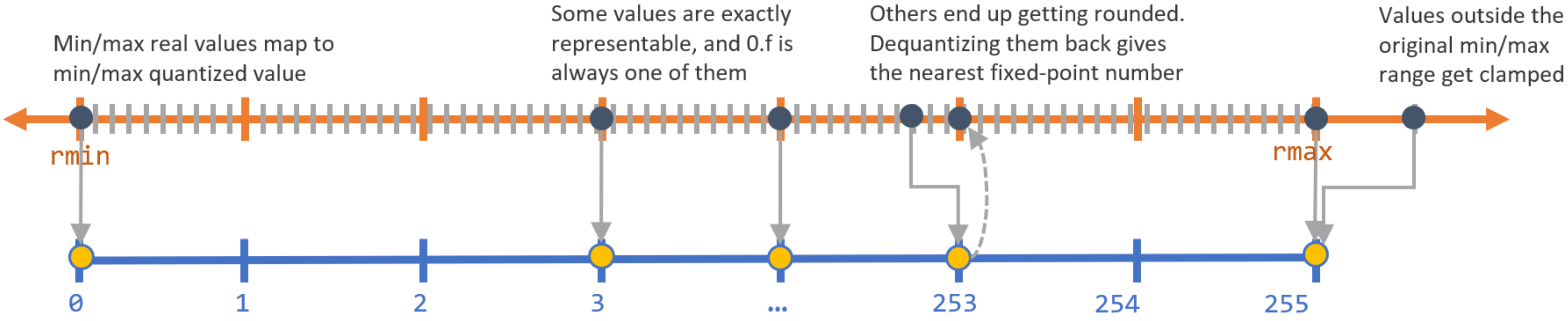
3. Optimize

Post-Training Quantization

PTQ는 학습을 완료한 다음에 Quantization 하는 방법

- Edge TPU가 있는 Coral Board를 사용할 때, 8-bit Integer 연산만 가능함
- Quantization을 통해 모든 weight와 activation은 0~255 또는 2's complement -128~127 의 8-bit 정수로 변환됨
- 32bit→8bit로, 모델의 크기는 75% 정도 작아짐

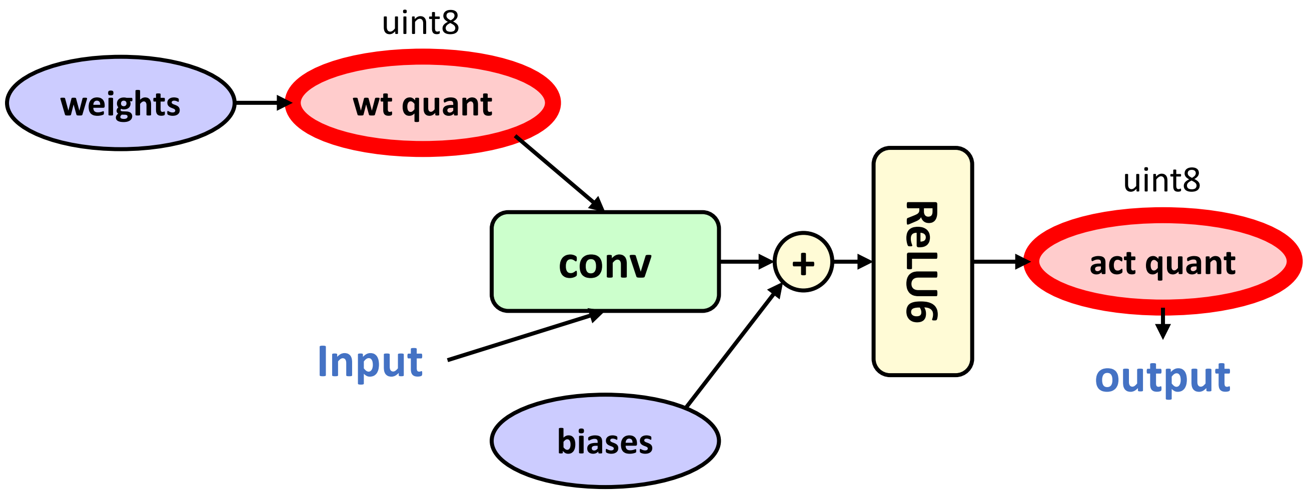




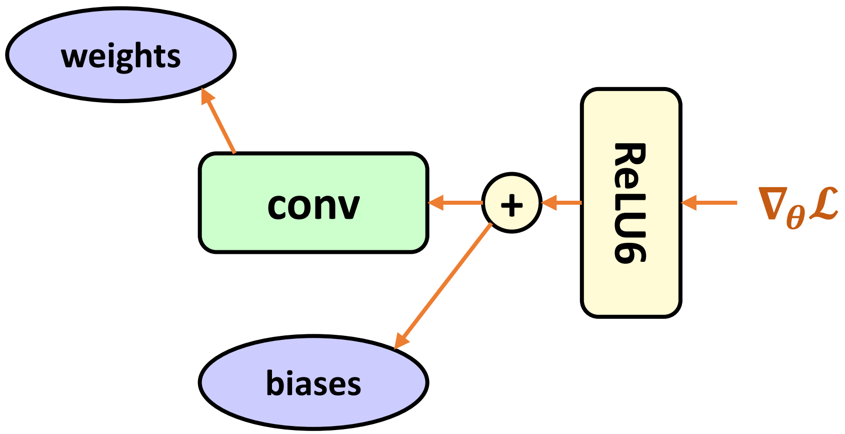
Quantization-Aware Training

QAT는 **학습 도중**에 이루어지고, inference할 때는 integer연산, backpropagation할 때에는 full-precision으로 모델을 학습함

- QAT 방식으로 학습하면, 최종 quantized 성능이 PTQ보다 좋다고 함



Inference



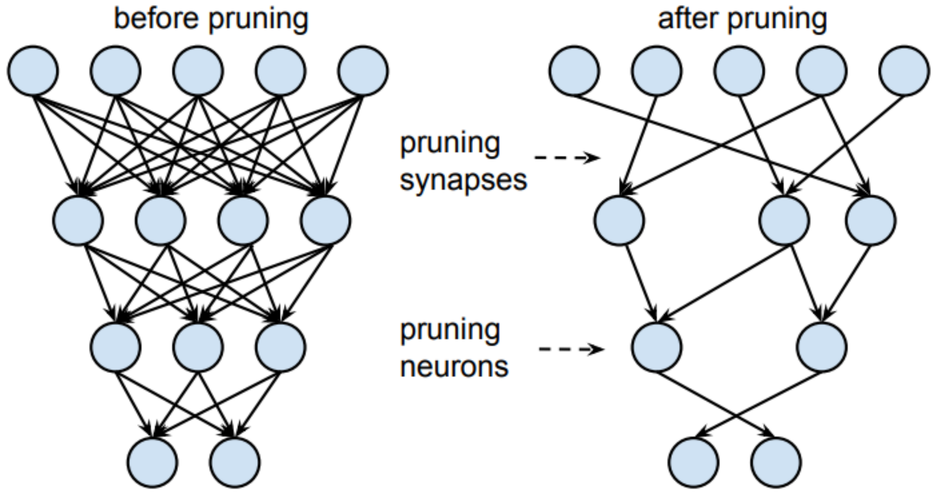
Backpropagation

Pruning

불필요한 (0에 가까운) weight들을 0으로 만들고 없애면서 모델 경량화

TFLite에서는 Gradual Pruning 방법론을 사용함

- `initial_sparsity`: pruning을 시작할 때의 sparsity를 몇으로 할지
- `final_sparsity`: pruning을 끝낼 때 sparsity를 몇으로 할지
- `begin_step`: pruning을 언제부터 진행할 지(batch 단위의 step)
- `end_step`: pruning을 언제 끝낼 지



To prune, or not to prune: exploring the efficacy of pruning for model compression [arXiv ‘17]

Thank You!