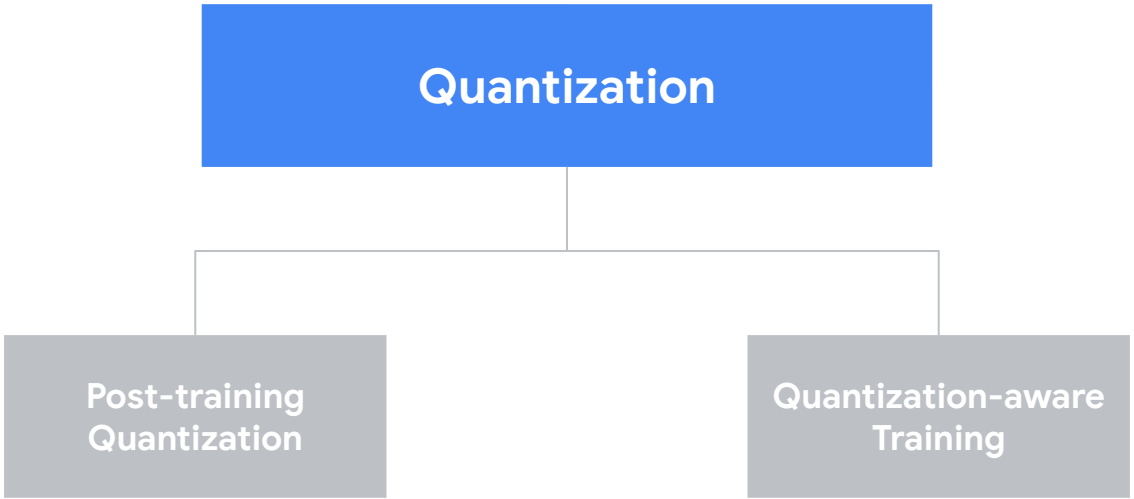
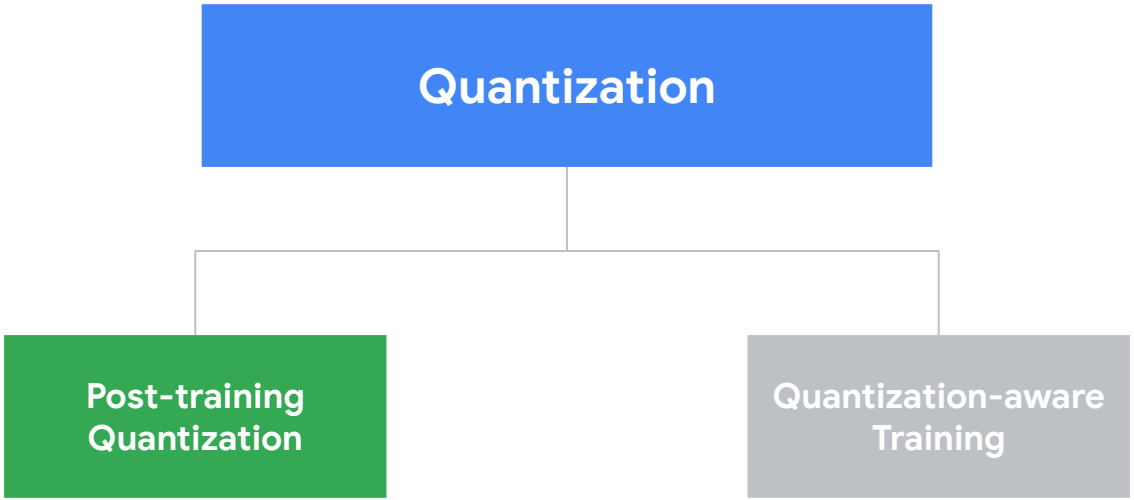
Quantization-aware Training (QAT)


Quantization
Post-training Quantization
Quantization-aware Training

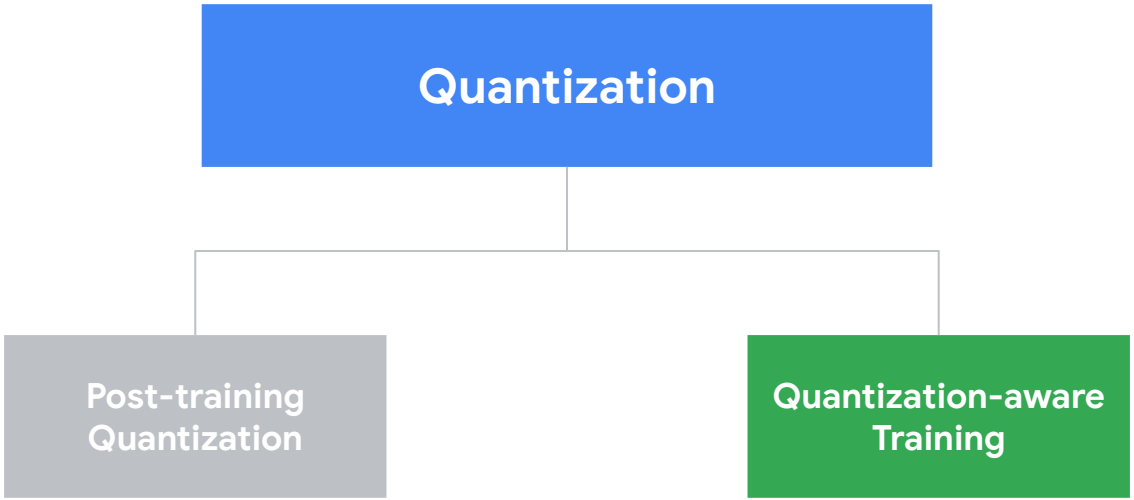

Quantization
Post-training Quantization
Quantization-aware Training

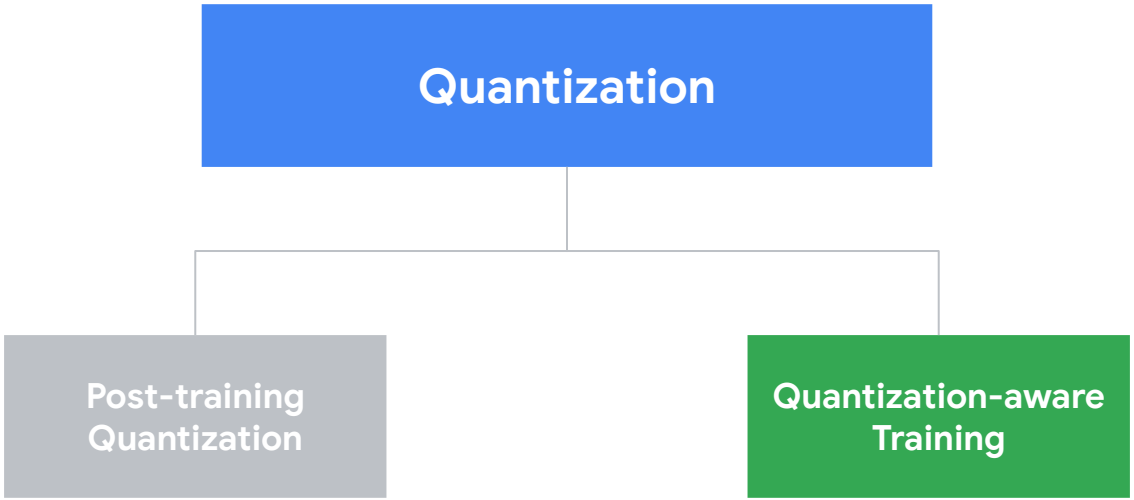
	Floating-point Baseline
MobileNet v1 1.0 224	71.03%
MobileNet v2 1.0 224	70.77%
Resnet v1 50	76.30%

	Floating-point Baseline	Post-training Quantization (PTQ)
MobileNet v1 1.0 224	71.03%	69.57%
MobileNet v2 1.0 224	70.77%	70.20%
Resnet v1 50	76.30%	75.95%

	Floating-point Baseline	Post-training Quantization (PTQ)	Accuracy Drop
MobileNet v1 1.0 224	71.03%	69.57%	▼1.46%
MobileNet v2 1.0 224	70.77%	70.20%	▼0.57%
Resnet v1 50	76.30%	75.95%	▼0.35%

Is this accuracy drop tolerable?


Quantization
Post-training Quantization
Quantization-aware Training

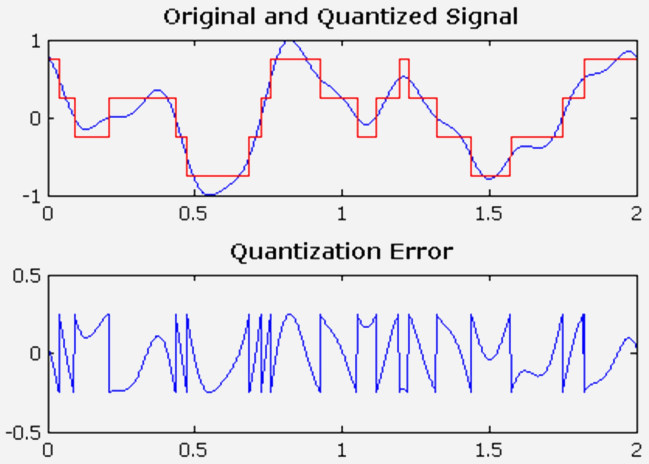


Quantization aware training ***emulates inference-time quantization***, creating a model that downstream tools will use to produce actually quantized models. The quantized models use lower-precision (e.g. 8-bit instead of 32-bit float), leading to benefits during deployment.

Why does the **accuracy drop**?

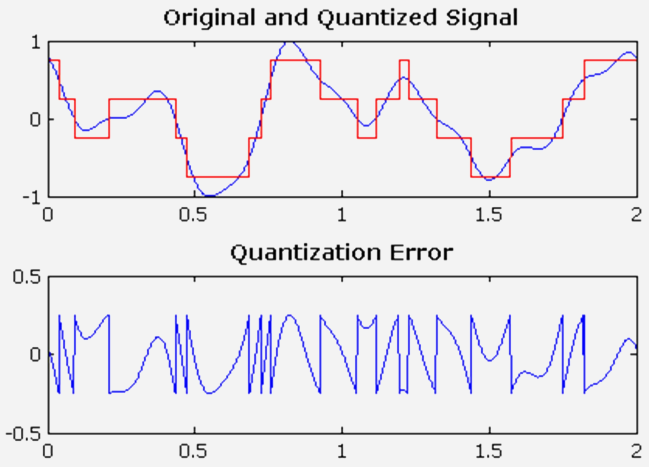
Why does the **accuracy drop**?

- Introducing **error** by ***discretizing*** the values



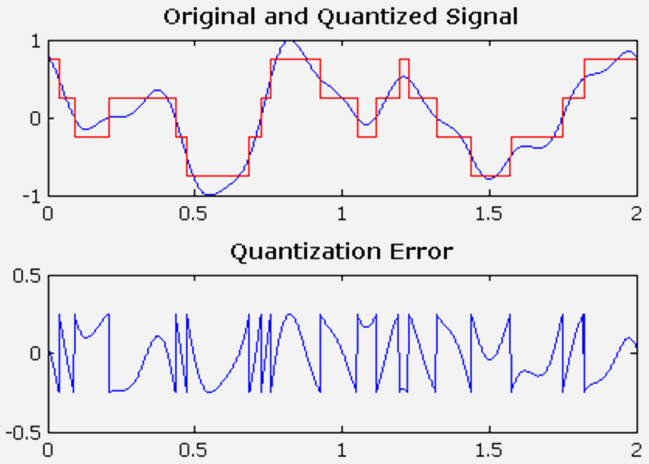
Why does the **accuracy drop**?

- Introducing **error** by ***discretizing*** the values
- Quantized weights are in **int8** instead of **fp32**



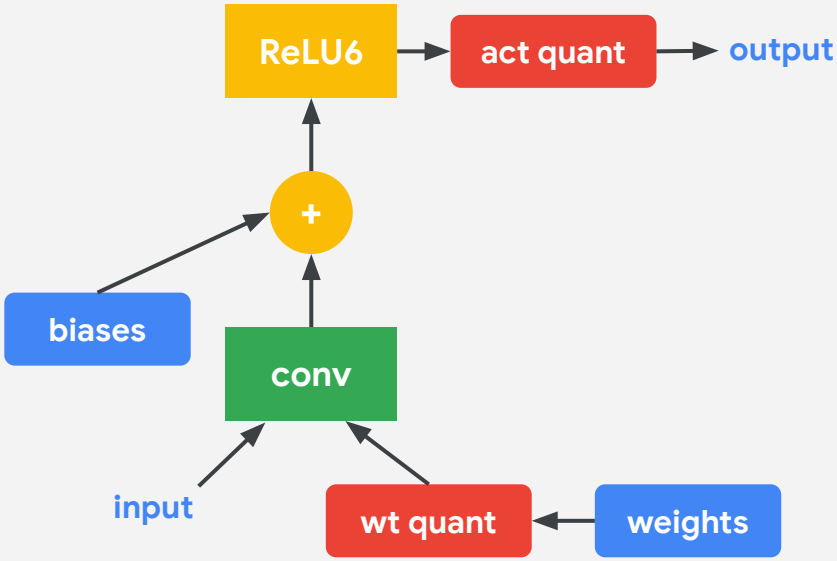
Why does the **accuracy drop**?

- Introducing **error** by ***discretizing*** the values
- Quantized weights are in **int8** instead of **fp32**
- **Many** different **conversions** (computation: int8, accumulations: int32, rescaling: int8)



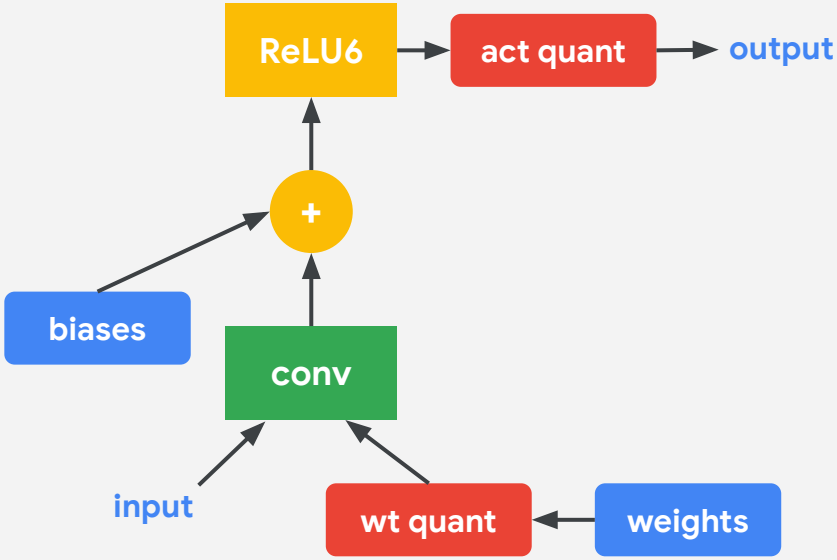
Quantization-aware Training

- **Mimic** the **inference path during** the **training** phase



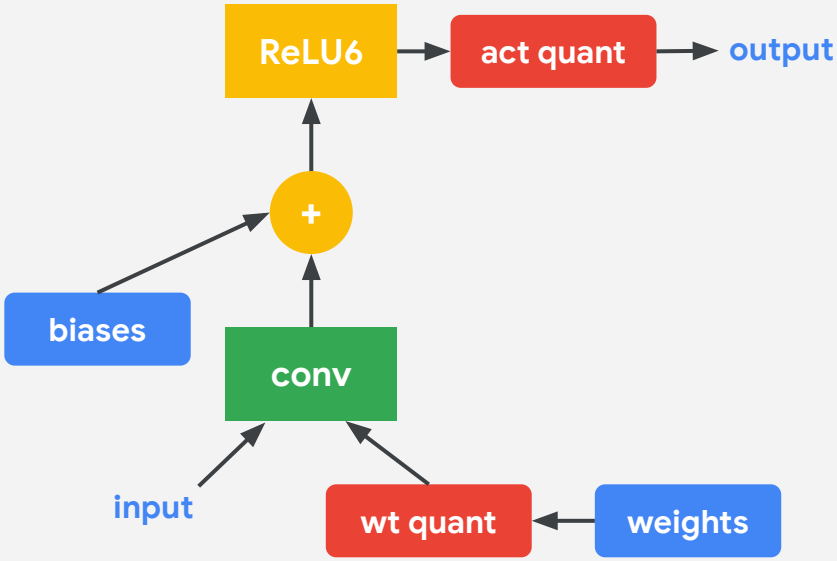
Quantization-aware Training

- Mimic the inference path during the training phase
- Expose the training pipeline to the errors observed



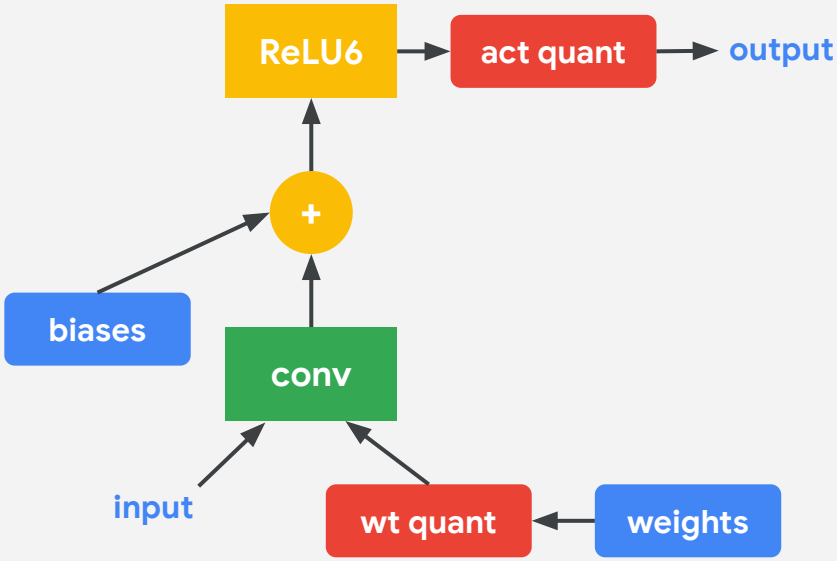
Quantization-aware Training

- **Mimic** the **inference path during** the **training** phase
- **Expose** the **training pipeline to** the **errors** observed
- Allow the **training phase** to **recover** the **error** ***"naturally"***



Quantization-aware Training

- **Mimic** the **inference path during** the **training** phase
- **Expose** the **training pipeline to** the **errors** observed
- Allow the **training phase** to **recover** the **error** ***"naturally"***
- **Weights** *and* **inputs** use the same int8—mimic int8 MAC



	Floating-point Baseline	Post-training Quantization (PTQ)	Quantization-Aware Training (QAT)
MobileNet v1 1.0 224	71.03%	69.57%	71.06%
MobileNet v2 1.0 224	70.77%	70.20%	70.01%
Resnet v1 50	76.30%	75.95%	76.10%

	Floating-point Baseline	Post-training Quantization (PTQ)	Quantization-Aware Training (QAT)
MobileNet v1 1.0 224	71.03%	69.57%	71.06%
MobileNet v2 1.0 224	70.77%	70.20%	70.01%
Resnet v1 50	76.30%	75.95%	76.10%

	Floating-point Baseline	Post-training Quantization (PTQ)	Quantization-Aware Training (QAT)
MobileNet v1 1.0 224	71.03%	69.57%	71.06%
MobileNet v2 1.0 224	70.77%	70.20%	70.01%
Resnet v1 50	76.30%	75.95%	76.10%

:(

```
import tensorflow_model_optimization as tfmot

quantize_model = tfmot.quantization.keras.quantize_model

# q_aware stands for for quantization aware.
q_aware_model = quantize_model(model)

# `quantize_model` requires a recompile.
q_aware_model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

Leave space for headshot

```
import tensorflow_model_optimization as tfmot

quantize_model = tfmot.quantization.keras.quantize_model

# q_aware stands for for quantization aware.
q_aware_model = quantize_model(model)

# `quantize_model` requires a recompile.
q_aware_model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

Leave space for headshot

```
import tensorflow_model_optimization as tfmot

quantize_model = tfmot.quantization.keras.quantize_model

# q_aware stands for for quantization aware.
q_aware_model = quantize_model(model)

# `quantize_model` requires a recompile.
q_aware_model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

Leave space for headshot

```
import tensorflow_model_optimization as tfmot

quantize_model = tfmot.quantization.keras.quantize_model

# q_aware stands for for quantization aware.
q_aware_model = quantize_model(model)

# `quantize_model` requires a recompile.
q_aware_model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

Leave space for headshot

```
import tensorflow_model_optimization as tfmot

quantize_model = tfmot.quantization.keras.quantize_model

# q_aware stands for for quantization aware.
q_aware_model = quantize_model(model)

# `quantize_model` requires a recompile.
q_aware_model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

Leave space for headshot

```
loss: 0.2724 - accuracy: 0.9244 -
val_loss: 0.1085 - val_accuracy: 0.9695
```

Leave space for headshot

```
loss: 0.2724 - accuracy: 0.9244 -
val_loss: 0.1085 - val_accuracy: 0.9695
```

FP32

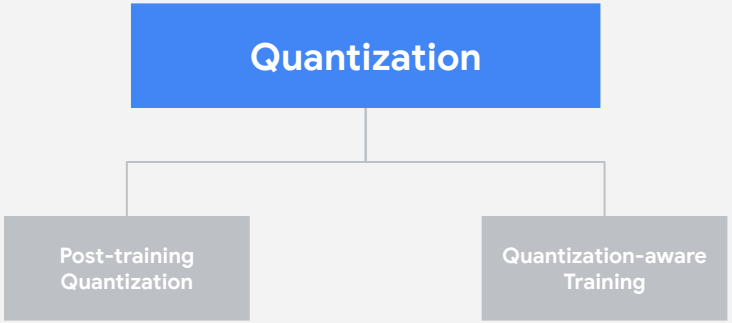
```
loss: 0.1315 - accuracy: 0.9589 -
val_loss: 0.1360 - val_accuracy: 0.9600
```

INT8

Leave space for headshot

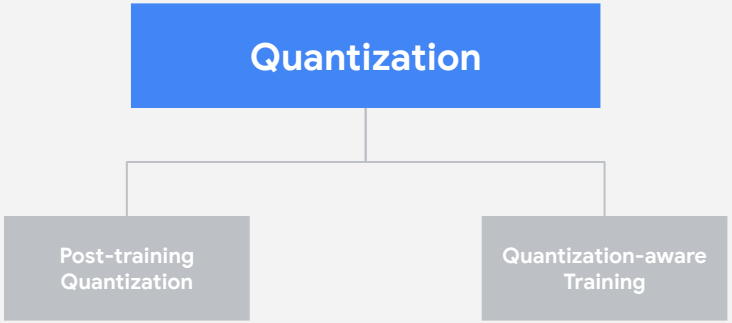
Summary

1. What is **quantization**?



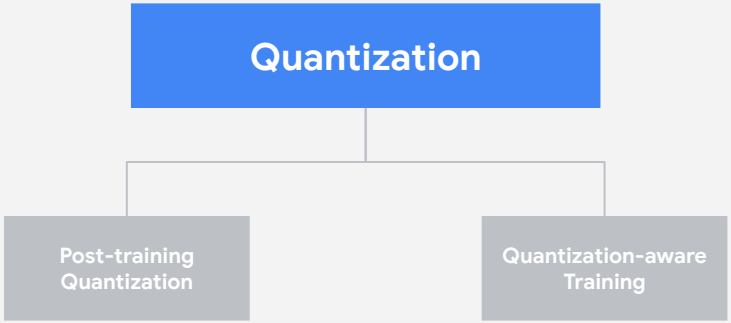
Summary

1. What is **quantization**?
2. ***Why*** is quantization important for **TinyML**?



Summary

1. What is **quantization**?
2. ***Why*** is quantization important for **TinyML**?
3. Understand **PTQ** and its **accuracy loss** reasons.



Summary

1. What is **quantization**?
2. ***Why*** is quantization important for **TinyML**?
3. Understand **PTQ** and its **accuracy loss** reasons.
4. How does **QAT** use **training** to **reduce** quantization **loss**?