# Introducción a la Cuantización

Jose Luis Rodríguez, Gonzalo Martínez y Alenxandre Muñoz

October 2023

## 1 Introducción

La cuantización se refiere al proceso de convertir valores de números reales en valores de números enteros o fraccionarios, reduciendo así la cantidad de bits necesarios para representar esos valores. En el contexto del aprendizaje automático y la inferencia en dispositivos de recursos limitados, como dispositivos móviles o sistemas integrados, la cuantización es esencial para reducir el tamaño de los modelos y mejorar la eficiencia computacional. Dos enfoques comunes para la cuantización son la Cuantización Posterior al Entrenamiento (PTQ) y el Entrenamiento Consciente de la Cuantización (QAT).

### 1.1 Bases de la Cuantización

La cuantización se basa en el principio de reducción de la precisión de los números para reducir la complejidad y el costo computacional. Se representan números de punto flotante de alta precisión en una forma más compacta, como enteros o fraccionarios con menos bits. La cuantización puede llevarse a cabo en varios niveles, desde la cuantización de parámetros (pesos del modelo) hasta la cuantización de activaciones (resultados intermedios de capas). Los niveles más comunes de cuantización incluyen:

Qunatización de Parámetros: En este nivel, los valores de los parámetros del modelo (pesos) se cuantizan para reducir su precisión.

Qunatización de Activaciones: Aquí, las activaciones generadas durante la inferencia se cuantizan antes de ser procesadas por las capas subsiguientes del modelo.

### 1.2 Diferencias entre PTQ y QAT

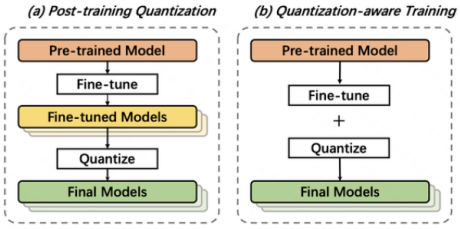


Figure 1: PTQ y QAT

1. **Cuantización Posterior al Entrenamiento (PTQ)**

   -En PTQ, la cuantización se realiza después del entrenamiento del modelo.

   -El modelo se entrena en números de punto flotante de alta precisión y luego se cuantiza para la inferencia.

   -Es un proceso de postprocesamiento que no afecta el proceso de entrenamiento inicial.

   -PTQ puede aplicarse a modelos pre-entrenados sin necesidad de un nuevo proceso de entrenamiento, pero puede ser menos preciso que QAT.

2. **Entrenamiento Consciente de la Cuantización (QAT)**

   -QAT es un enfoque en el que la cuantización se tiene en cuenta durante el entrenamiento del modelo.

   -El modelo se entrena directamente con cuantización.

   -Durante el entrenamiento, se simula la cuantización en tiempo real, lo que permite que el modelo se adapte mejor a la cuantización y se optimice para el rendimiento en números cuantizados.

   -QAT tiende a producir modelos cuantizados más precisos, pero puede requerir un proceso de entrenamiento más largo y costoso en comparación con PTQ.

En resumen, la cuantización es esencial para implementar modelos de aprendizaje automático eficientes en dispositivos con recursos limitados. La elección entre ellos depende de las necesidades específicas del proyecto y de si se requiere una precisión máxima o una implementación más rápida. PTQ es más sencillo y adecuado para la postprocesamiento de modelos pre-entrenados, mientras que QAT es más preciso pero puede ser más costoso en términos de entrenamiento.

# 2 Representación de Coma Flotante

La representación de números de coma flotante se utiliza para representar números reales en sistemas de cómputo, como computadoras y dispositivos de procesamiento. Las representaciones de números de coma flotante se utilizan en una amplia variedad de aplicaciones, desde cálculos científicos y de ingeniería hasta aprendizaje automático y gráficos computacionales. Tres formatos comunes de números de coma flotante son FP32 (32 bits de precisión simple), FP16 (16 bits de precisión media) y BF16 (16 bits de coma flotante bajas), y cada uno tiene sus propias características y aplicaciones.

## 2.1 FP32 (32 bits de precisión simple)

- FP32 es ampliamente utilizado en aplicaciones de cómputo de propósito general.
- Se compone de un bit para el signo, ocho bits para el exponente y 23 bits para la mantisa.

## 2.2 FP16 (16 bits de precisión media)

- FP16 utiliza 16 bits para representar un número real.
- Incluye un bit para el signo, cinco bits para el exponente y diez bits para la mantisa.
- Se emplea en redes neuronales profundas (DNN) y gráficos 3D en tiempo real.

## 2.3 BF16 (16 bits de coma flotante bajas)

- BF16 utiliza 16 bits para representar un número real.
- Incluye un bit para el signo, ocho bits para el exponente y siete bits para la mantisa.

## 2.4 Ventajas y Desventajas de cada tipo en términos de precisión, rango y uso de memoria.

Produndizando en términos de precisión, rango y uso de memoria encontramos:

## 2.5 Precisión

### 2.5.1 FP32 (32 bits de precisión simple)

**Ventajas:**

- Ofrece alta precisión y puede representar números con una gran cantidad de decimales.

- Adecuado para aplicaciones donde la precisión es crítica, como cálculos científicos y financieros.

**Desventajas:**

- Requiere más bits para representar un número, lo que consume más memoria y ancho de banda.
- Ineficiente en aplicaciones donde la precisión de 32 bits no es esencial.

### 2.5.2 FP16 (16 bits de precisión media)

**Ventajas:**

- Ofrece una representación más compacta en comparación con FP32, lo que ahorra memoria y reduce los requisitos de ancho de banda en cálculos intensivos.
- Adecuado para aplicaciones donde una precisión de 32 bits no es crítica.

**Desventajas:**

- Propenso a errores de desbordamiento y pérdida de precisión en operaciones matemáticas avanzadas.
- No es adecuado para aplicaciones que requieren una alta precisión, como cálculos científicos de alta fidelidad.

### 2.5.3 BF16 (16 bits de coma flotante bajas)

**Ventajas:**

- Ofrece un equilibrio entre la precisión de FP32 y la eficiencia de almacenamiento de FP16.
- Adecuado para aplicaciones de inteligencia artificial y aprendizaje automático, donde se requiere un rendimiento eficiente sin comprometer significativamente la precisión.

**Desventajas:**

- Aunque ofrece una mayor precisión en comparación con FP16, todavía puede ser menos preciso que FP32.
- No es adecuado para aplicaciones de alta precisión, como simulaciones científicas de alta fidelidad.

## 2.6 Rango:

- **FP32:** Tiene un rango amplio y puede representar números con una gran magnitud y variedad de valores.

  $1.18 \times 10^{-38}$ ... $3.40 \times 10^{38}$ con precisión de 6 a 9 dígitos decimales significativos.

- **FP16:** Tiene un rango más limitado en comparación con FP32, lo que puede llevar a desbordamientos en cálculos con números muy grandes o muy pequeños.

  $5.96 \times 10^{-8}$ $(6.10 \times 10^{-5})$ ... $6.5504 \times 10^{4}$ con precisión de 4 dígitos decimales significativos.

- **BF16:** Tiene un rango similar al de FP16 y, por lo tanto, también es más limitado que FP32.

  $1.18 \times 10^{-38}$ ... $\sim 3.40 \times 10^{38}$ con 3 dígitos decimales significativos.

## 2.7 Uso de Memoria:

- **FP32:** Requiere más memoria que FP16 y BF16 debido a su mayor número de bits.

- **FP16:** Ahorra memoria en comparación con FP32, lo que es beneficioso en aplicaciones con restricciones de memoria.

- **BF16:** Ofrece un equilibrio entre la eficiencia de memoria de FP16 y una precisión mejorada en comparación con FP16.

En resumen, FP32 ofrece alta precisión, pero consume más recursos, mientras que FP16 y BF16 son formatos de menor precisión que ahorran recursos de memoria y ancho de banda. La elección de un formato de coma flotante depende de las necesidades específicas de la aplicación y la capacidad de hardware disponible, y es importante equilibrar la precisión y la eficiencia en cada caso.