**UNIVERSIDAD MAYOR, REAL Y PONTIFICIA DE**

**SAN FRANCISCO XAVIER DE CHUQUISACA**

**Facultad de Ciencias de la Tecnología**

**A logo of a city

AI-generated content may be incorrect.**

LoRA y QLoRA

**ESTUDIANTE:** Polo Orellana Brayan Simón

**CARRERA:** Ingeniería en Ciencias de la computación

**MATERIA:**  Inteligencia Artificial 2

**SIGLA:**  SIS-421

Introduccion

Recientemente, los LLM(Large Lenguage Models) han demostrado un rendimiento sin precedentes en un amplia gama de tareas de comprensión de lenguaje y sirvió como base par sistemas de chat. La diversidad de aplicaciones en el mundo real exige un proceso en el que los LLM puedan ajustarse con precisión para adaptarse a diferentes escenarios y cuantificarse para su implementación en dispositivos periféricos (por ejemplo, teléfonos moviles). La clave resude en eliminar la pesada carga computacional que supone la gran cantidad de parámetros de los LLM.

Con el rápido crecimiento de los modelos de lenguaje grandes (LLMs), el fine-tuning de estos modelos se ha convertido en un desafío crítico. Los métodos tradicionales de fine-tuning demandan recursos computacionales significativos, haciéndolos imprácticos para muchas organizaciones. La Adaptación de Bajo Rango Cuantizada (QLoRA) es un enfoque nuevo que reduce las necesidades de memoria mientras conserva la calidad del modelo, permitiendo la afinación de LLMs en hardware de consumo.

Que es Fine-Tunning

La primera es la eficiencia de parámetros con **Fine-tunning,** el cual introduce un numero pequeño de parámetros entrenables mientras mantiene sin cambios la mayoría de los parámetros entrenados previamente.

**LoRA (Low-Rank Adaptation)**

Es un algoritmo PEFT popular, propuesto **para ajustar matrices de bajo rango** y complementar los pesos preentrenados. A pesar de su rendimiento comparable con el **fine-tunning** de parámetros completos, el uso de memoria de LoRA sigue siendo elevado, especialmente cuando el LLM base es grando (ejemplo LLaMA).

LoRA de las Siglas **Low-Rank Adaptation (Adaptacion de Bajo Rango),** es un método de reentrenamiento que reutiliza un modelo de lenguaje básico para una tarea específica. Explora cómo LoRA te permite aprovechar la tecnología de un LLM y entrenarlo de forma rápida y eficiente para las necesidades de uno mismo.

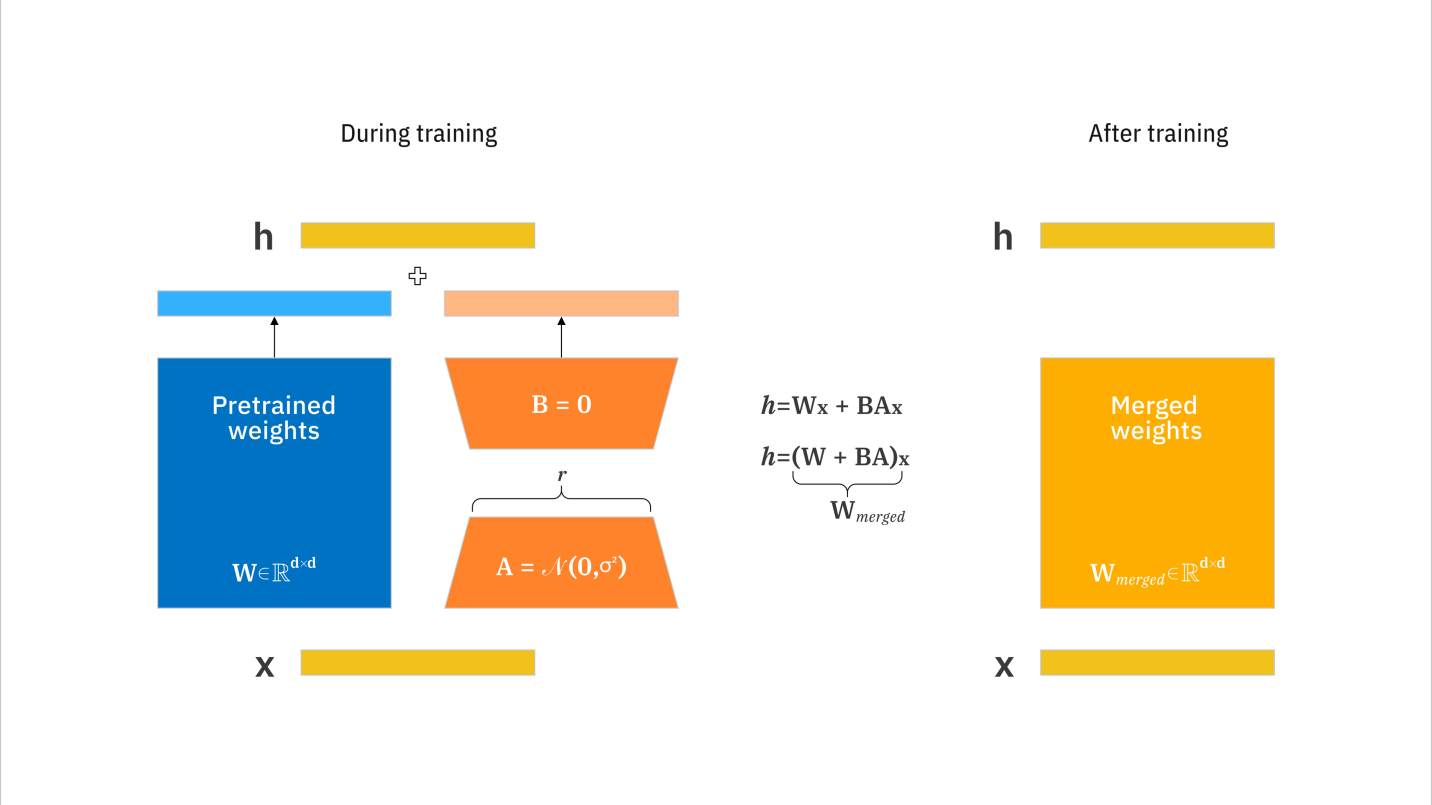
Su función principal es adaptar un modelo base a una tarea modificando los pesos de un subconjunto representativo de los parámetros del modelo, denominados adaptadores de bajo rango, en lugar de los pesos del modelo base durante el ajuste. En la Inferencia, los pesos de los adaptadores ajustados se suman a los pesos del modelo base para generar un resultado ajustado a la tarea.

**Como funciona LoRA**

En lugar de reentrenar todo el modelo, LoRA congela los pesos y parámetros originales del modelo tal como están. Luego, sobre este modelo original, añade una adición ligera llamada una matriz de bajo rango, que luego se aplica a nuevas entradas para obtener resultados específicos del contexto. La matriz de bajo rango ajusta los pesos del modelo original para que las salidas coincidan con el caso de uso deseado.

LoRA aprovecha el concepto de matrices de menor rango para hacer que el proceso de entrenamiento del modelo sea extremadamente eficiente y rápido. Tradicionalmente, la microajuste de LLMs requiere ajustar todo el modelo. LoRA se centra en modificar un subconjunto más pequeño de parámetros (matrices de menor rango) para reducir el sobrecosto computacional y de memoria.

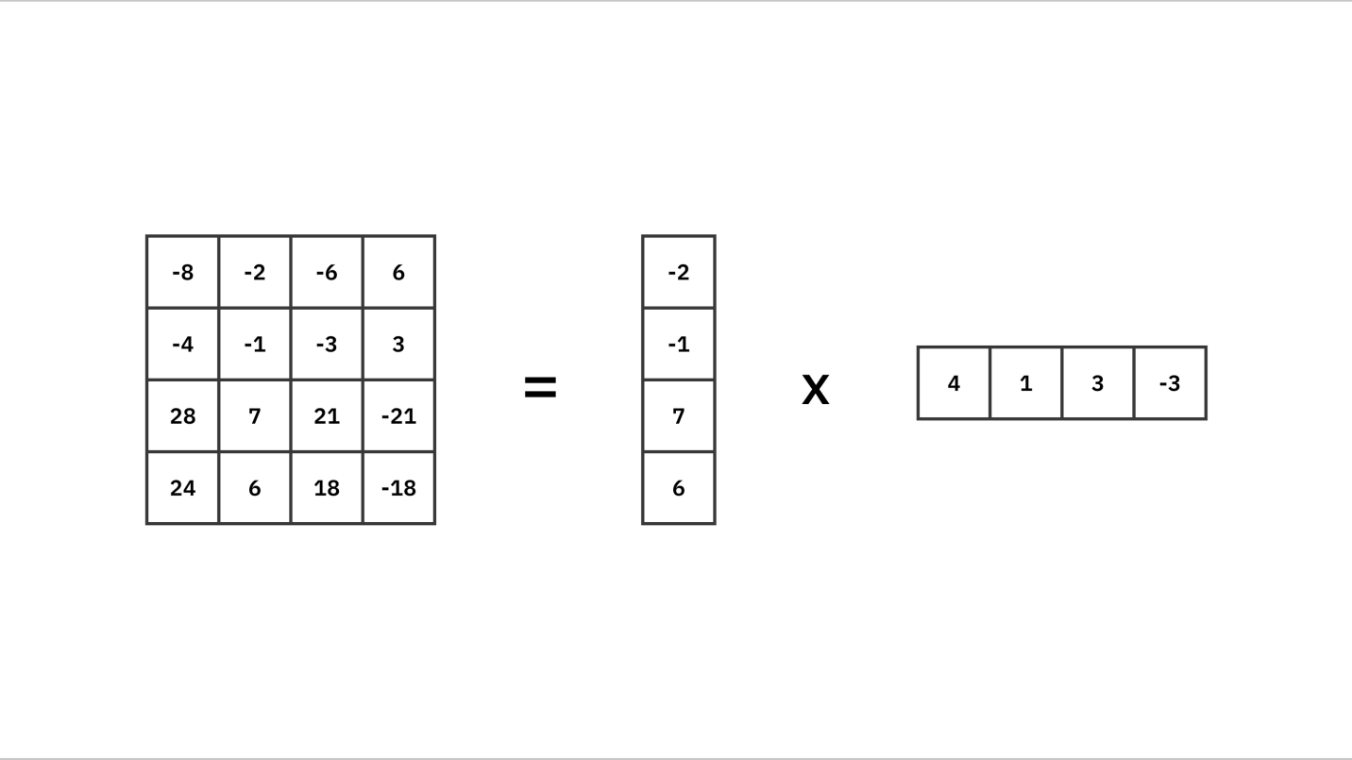
El diagrama muestra cómo LoRA actualiza las matrices A y B para rastrear los cambios en los pesos preentrenados utilizando matrices más pequeñas de rango r. Una vez completada la capacitación de LoRA, los pesos más pequeños se fusionan en una nueva matriz de pesos, sin necesidad de modificar los pesos originales del modelo preentrenado.



LoRA funciona añadiendo pares de matrices de descomposición de rangos a las capas de transformador, centrándose generalmente en los pesos de atención. Durante la inferencia, estos pesos de adaptador pueden fusionarse con el modelo base, lo que elimina la sobrecarga de latencia adicional. LoRA es particularmente útil para adaptar modelos de lenguaje extensos a tareas o dominios específicos, manteniendo.

Las matrices son una parte importante de cómo funcionan los modelos de aprendizaje automático y las redes neuronales. Las matrices de rango bajo son más pequeñas y tienen muchos menos valores que las matrices más grandes o de rango más alto. No ocupan mucha memoria y requieren menos pasos para sumar o multiplicarse entre sí, lo que las hace más rápidas para que los ordenadores las procesen.

Una matriz de rango alto puede descomponerse en dos matrices de rango bajo, una matriz de 4x4 puede descomponerse en una matriz de 4x1 y una matriz de 1x4.



LoRA añade matrices de bajo rango al modelo original de aprendizaje automático congelado. Las matrices de bajo rango se actualizan a través del descenso de gradiente durante el ajuste fino, sin modificar los pesos del modelo base. Estas matrices contienen nuevos pesos para aplicar al modelo al generar resultados. La matriz de cambio multiplicada se suma a los pesos del modelo base para obtener el modelo final ajustado. Este proceso altera las salidas que el modelo produce con mínima potencia de cómputo y tiempo de entrenamiento.

En esencia, LoRA mantiene el modelo original sin cambios y añade pequeñas partes variables a cada capa del modelo. Esto reduce significativamente los parámetros entrenables del modelo y la necesidad de memoria GPU para el proceso de entrenamiento, lo que es otro desafío significativo cuando se trata de ajustar o entrenar modelos grandes.

**QLoRA(Quantized Low-Rank Adaptation)**

Consiste en la cuantificación de parámetros, donde los pesos entrenados se cuantifican en enteros de bajo número de bits o números de punto flotante. Si bien estos métodos pueden reducir la carga computacional, a menudo presentan una precisión deficiente, especialmente cuando el ancho de bits de cuantificación es bajo.

La idea central detrás de QLoRA es cargar el modelo base preentrenado en un formato cuantizado, típicamente 4-bit, reduciendo drásticamente su requisito de memoria. Crucialmente, mientras el modelo base se mantiene en este formato de baja precisión, los adaptadores LoRA se entrenan en una mayor precisión (por ejemplo, bfloat16). Este enfoque permite importantes ahorros de memoria sin una pérdida catastrófica en el rendimiento, haciéndolo factible ajustar fino modelos masivos (por ejemplo, 65 mil millones de parámetros) en hardware con limitada VRAM, como un solo GPU de consumo

**Como funciona QLoRA**

**a) Cuantización de pesos**

Antes de entrenar, los pesos del modelo () se **cuantizan** usando un formato especial llamado **NF4 (Normalized Float 4)**.

NF4 es una representación en 4 bits optimizada estadísticamente que conserva la distribución de valores de los pesos originales, reduciendo la pérdida de precisión.

Esto reduce el uso de memoria **4 veces**.

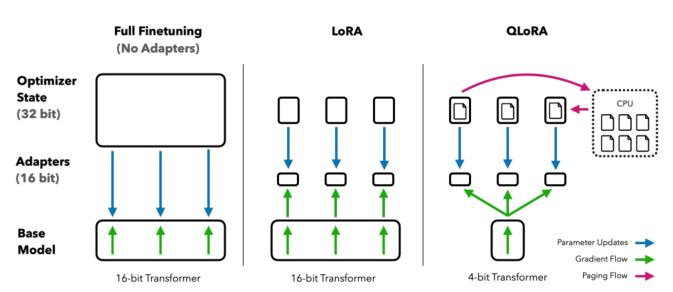
**b) Descuantización dinámica (Double Quantization)**

Durante el entrenamiento, QLoRA **no descuantiza completamente los pesos**, sino que los mantiene en forma comprimida y solo los convierte parcialmente cuando es necesario para operaciones matriciales.  
Esto se llama **double quantization** — se cuantizan tanto los pesos como los parámetros de cuantización, para ahorrar aún más memoria.

**c) Inserción de adaptadores LoRA**

Igual que en LoRA, no se modifican los pesos cuantizados.  
Se añaden matrices **A** y **B** de bajo rango (en FP16) para representar las actualizaciones.

Solo **A** y **B** se entrenan.  
El modelo base permanece cuantizado, congelado y eficiente.



**Idea matemática y formula básica**

Consideramos una capa lineal típica con peso . En el forward normal:

(o en notación batch )

En LoRA, la matriz de pesos se parametriza como:

donde

Donde A es Matriz de proyección Descendente, donde reduce la dimensionalidad de la entrada a un espacio de dimensión .

Donde B es Matriz de proyección Ascendente, donde reconvierte esa representación comprimida al tamaño de salida original.

Donde α es Factor de escalado (ajusta magnitud de actualizacion) que es un hiperparametro no entrenable

con y , siendo el rango bajo ().  
Durante el entrenamiento, los parámetros permanecen **congelados**, mientras que solo se optimizan y .

El término de escalado se introduce para controlar la magnitud de la actualización:

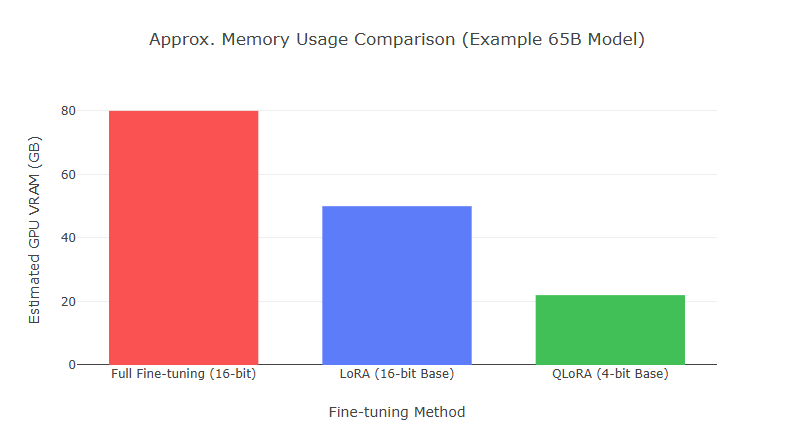
De esta manera, el número de parámetros entrenables se reduce de a , logrando una reducción significativa en el consumo de memoria.

Donde r es el rango de la descomposición de baja dimensión usado para representar

**Ventajas y Desventajas**

**Ventajas**

* **Reducción Masiva de Memoria:** QLoRA reduce drásticamente la VRAM necesaria para el ajuste fino, permitiendo la adaptación de modelos muy grandes (30B, 65B+) en hardware accesible (por ejemplo, GPUs con 24GB o 48GB de VRAM).
* **Preservación del Rendimiento:** A pesar de la cuantización de 4 bits del modelo base, QLoRA logra típicamente un rendimiento muy cercano al ajuste fino completo de 16 bits o a 16-bit LoRA en muchas métricas y tareas. El uso de NF4 y el cálculo en bfloat16 son esenciales para esto.
* **Compatibilidad:** Se integra bien con ecosistemas existentes como los bibliotecas transformers y peft de Hugging Face, a menudo requiriendo solo cambios en la configuración.



**Desventajas**

* **Reducción Potencial de Troughput:** La desquantización en tiempo real durante la pasada hacia adelante añade sobrecarga computacional. Si bien QLoRA ahorra memoria, podría resultar en pasos de entrenamiento más lentos (menor throughput) en comparación con el estándar LoRA si la memoria no era el cuello de botella principal.
* **Diferencias Menores de Rendimiento:** Si bien generalmente se logra un rendimiento comparable, podrían haber caídas ligeras en ciertos métricos en comparación con los métodos de 16 bits en ciertas tareas.
* **Dependencias de Implementación:** Requiere bibliotecas específicas como bitsandbytes para la cuantización NF4, DQ y Optimizadores Página.