Incremental low-precision matrix multiplication

Pre-requisites:

1. **Floating-Point Types**:
    - **Floating-point types** are numerical data types used to represent real numbers with a fractional component. They are stored in a format that includes a sign bit, an exponent, and a mantissa (or significand).
    - **32-bit (float32)**: Uses 1 bit for the sign, 8 bits for the exponent, and 23 bits for the mantissa.
    - **16-bit (float16)**: Uses 1 bit for the sign, 5 bits for the exponent, and 10 bits for the mantissa.
    - **8-bit (float8)**: Uses 1 bit for the sign, 4 bits for the exponent, and 3 bits for the mantissa.
2. **Quantization**:
    - **Quantization** is the process of mapping a large set of input values to a smaller set, often used to reduce the precision of numerical data.
    - **Uniform Quantization**: All quantization intervals are of equal size.
    - **Non-uniform Quantization**: Quantization intervals vary in size, often used to preserve more precision for frequently occurring values.
3. **K-Fold Precision**:
    - **K-Fold Precision** refers to the technique of using multiple lower-precision operations to achieve higher precision results. For example, using multiple float8 operations to approximate a float16 operation.
4. **Normalized Matrix**:
    - A **normalized matrix** is a matrix where the values are scaled to a specific range, typically [0, 1] or [-1, 1], to ensure numerical stability and consistency in computations.

Goal:

Multiply two normalized matrices with higher precision (e.g., float16) using multiple matrix multiplications of lower precision (e.g., float8). Ensure that the result of the first multiplication is a good proxy for the actual multiplication.

Methodology:

Build on prior work on vector and matrix multiplication using lower precision, with the additional constraint of achieving better approximation after each multiplication. Some prior works are:

- ULTIMATELY FAST ACCURATE SUMMATION
- Error-free transformations of matrix multiplication by using fast routines of matrix multiplication and its applications

Next Step:

Test this approach for training and inference of neural networks to provide incremental accuracy based on resource requirements.

Tasks:

1. Read the two papers and try to make a simple demo in python for a small matrix multiplication
2. Evaluate for Neural Network Training and Inference:
    - Integrate the incremental precision GEMM into a neural network training loop.
    - Measure the accuracy and resource utilization for different precision levels and number of steps.

NOTE. Please start with doing all tasks in Python.

Some Code Repos:
https://www.r-ccs.riken.jp/labs/lpnctrt/projects/ozblas/