Medical Image Processing

CHALLENGE

Processing medical images can be time-consuming, especially when dealing with large datasets. Therefore, the challenge is to optimise medical image processing using SYCL to improve the accuracy and speed of diagnosis.

END GOAL

The end goal is to build a medical image processing application that demonstrates the performance benefits of using SYCL for parallel computing. The application should process medical images faster and with higher accuracy than existing medical imaging applications.

1. Use SYCL's parallelism capabilities to speed up medical image processing.
2. Improve the accuracy of diagnosis through efficient feature extraction from medical images.
3. Benchmark the application against existing medical imaging applications in terms of performance, accuracy, and scalability.
4. Use a publicly available medical imaging dataset, such as the NIH Chest X-ray dataset or the BraTS dataset for brain tumour segmentation.

IMPORTANT NOTE

You can explore the SYCL/DCP++ Libraries to find the related libraries and optimizations for improving performance which will make your solution stand out.

ADDITIONAL REQUIREMENTS

Participants should explore the following options for SYCL to optimise medical image processing:

1. Device Selection: Selecting the best device for the use case, such as a CPU, GPU, or FPGA, to achieve the best performance.
2. Memory Management: Managing memory efficiently using SYCL to reduce data transfer times and improve overall performance.
3. Parallel Processing: Parallelizing image processing tasks using SYCL to speed up the processing time of medical images and improve the accuracy of diagnosis.
4. Feature Extraction: Using SYCL to extract important features from medical images, such as textures, shapes, and colours, to aid in the diagnosis of diseases and conditions.

SUGGESTED SYCL AND OTHER LIBRARIES

1. SYCL/DPC++: The SYCL/DPC++ libraries provide a C++ API for programming heterogeneous systems using SYCL and DPC++.
2. ComputeCpp: ComputeCpp is a SYCL implementation that allows developers to write standard C++ code for heterogeneous systems.
3. Eigen: Eigen is a C++ library for linear algebra that provides GPU acceleration through SYCL.
4. ArrayFire: ArrayFire is a library that provides GPU acceleration for scientific computing and machine learning using SYCL and other backends.
5. hipSYCL: hipSYCL is a SYCL implementation for AMD and NVIDIA GPUs, and supports both OpenCL and CUDA backends.

SAMPLE DATASETS AND RESOURCES

1. NIH Chest X-ray dataset: https://www.kaggle.com/nih-chest-xrays/data
2. The Cancer Imaging Archive (TCIA): https://www.cancerimagingarchive.net/
3. The Alzheimer's Disease Neuroimaging Initiative (ADNI): http://adni.loni.usc.edu/data-samples/
4. RSNA Pneumonia Detection Challenge: https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/data

DELIVERABLES (Idea Submission Phase)

1. Presentation.pdf: Participants will have to submit a PPT presentation of their proposed idea which should clearly explain their idea, emphasising on the usage of SYCL/DPC++ Libraries.

2. The submitted PPT must include a **Process Flow Diagram** and an **Architectural Diagram**

3. Exploring SYCL/DPC++ Libraries and their usage in the product is an essential aspect.

4. Products/Projects without oneAPI as the core component will not qualify for the hackathon

DELIVERABLES (Prototype Development Phase)

1. Participants are required to fork this repository and update the README file, filling in the required details. Then upload their code in the forked repository and create a pull request to submit the code.

2. Presentation.pdf: Participants will have to submit a PPT presentation of their prototype, showcasing their prototype workings along with a **Process Flow Diagram** and an **Architectural Diagram**. It should clearly mention the usage of SYCL/DPC++ Libraries

3. Participants are required to submit a comprehensive write-up that details their chosen problem statement, approach to the problem, and the code used to build their solution. This write-up should be in the form of a technical article posted on medium.com. It will be evaluated by the judges for the functionality and creativity of the submitted solution. Therefore, it is crucial to submit a complete and clear write-up to increase your chances of winning the competition.

4. Participants are also required to submit a prototype demo video.

5. Exploring the SYCL/DPC++ Libraries and their usage in the product is an essential aspect.

6. Products/Projects without oneAPI as the core component will not qualify for the hackathon.

7. The prototype submission must lay down equal emphasis on the deployment/inference benchmarking for both, with and without Intel® oneAPI.

CODE SUBMISSION GUIDELINES

1. The **prototype submission** will be made by creating a pull request on this repository.

2. Participants are required to fork this repository, Push their prototype code and update its README file with all the required information to count as a valid submission. **The required information is mentioned below.**

    a. Team Name,
    b. Problem Statement,
    c. Team Leader Email,
    d. A Brief of the Prototype,
    e. Tech Stack the prototype is Built Upon (Clearly mentioning SYCL/DCP++ Libraries used.)
    f. Step-by-Step Code Execution Instructions
    g. What I Learned

*Here is a link to a Sample Submission

CRITERIA FOR SUCCESS

The success of the project will be evaluated based on the following criteria:

1. Improved accuracy and speed of medical image processing compared to existing applications.
2. Effective use of SYCL's parallelism capabilities for medical image processing.
3. Efficient memory management using SYCL.
4. Successful feature extraction from medical images using SYCL.
5. Successful benchmarking against existing medical imaging applications in terms of performance, accuracy, and scalability.

JUDGING CRITERIA

1. Code quality (9 points)
2. Technical implementation (9 points)
3. Creativity & originality (9 points)

RESOURCES

VIDEOS
What is oneAPI
Introduction to oneDnn
oneAPI Deep Neural Networks Library Programming Model and Samples
oneAPI Video Processing Library Programming Model and Code Samples
oneAPI Collective Communications Library
oneAPI Video Processing Library Programming Model and Code Samples
oneAPI Threading Building Blocks (oneTBB)
The oneAPI Math Kernel Library (oneMKL)
oneAPI Collective Communications Library | oneCCL

oneDPL | oneAPI DPC++ Library
Direct Programming with SYCL
(SETUP) The Easiest, The Simplest C++ Parallel Library, oneTBB - SpinScoped MutexLock
Making banking secure via bio metrics application built using oneAPI and DPC++ based on
SYCL/C++
Parallel C++: Concurrent Containers
CUDA to SYCL Migration Tool and Method
Data Parallel C++ (DPC++) Programming Model
YouTube channel from an Intel innovator follow to know more

TOOLKITS & LIBRARY
oneAPI Deep Neural Network Library
Intel® oneAPI Threading Building Blocks
Intel® oneAPI Data Analytics Library
Intel® oneAPI Video Processing Library
Intel® oneAPI Collective Communications Library
Intel® oneAPI DPC++ Library
Intel® Optimization for TensorFlow*
Intel® Optimization for PyTorch*
Intel® Distribution for Python*
Intel® Extension for Scikit-learn
Intel® Neural Compressor

HANDS-ON WORKSHOPS

Live Hands-on Workshops	Date

Accelerate AI workloads with Intel® oneDNN & oneDAL	19th April (5-7 PM)
Solving Operating System Concepts with SYCL	26th April (5-7 PM)