National Science Foundation's Industry/University Cooperative Research (I/UCRC) Program

Multiarchitecture Hardware Acceleration of Hyperdimensional Computing Using oneAPI

Mission-Critical Computing
NSF CENTER FOR SPACE, HIGH-PERFORMANCE, AND RESILIENT COMPUTING (SHREC)

oneAPI August 2023 DevSummit

Dr. Alan George
Mickle Chair Professor of ECE
University of Pittsburgh

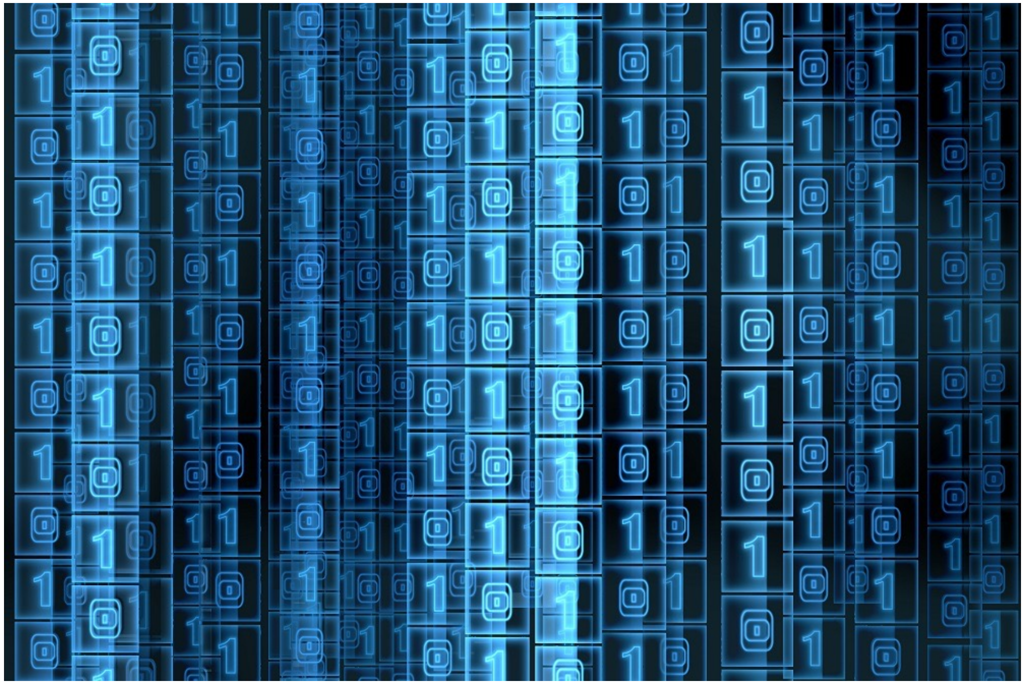
Ian Peitzsch
Research Student
University of Pittsburgh

University of Pittsburgh
BYU BRIGHAM YOUNG UNIVERSITY
VIRGINIA TECH
UF UNIVERSITY of FLORIDA

August 21, 2023

Agenda

- What is SHREC?
- Background
- System Overview
- Approach
- Results
- Conclusions



What is SHREC?

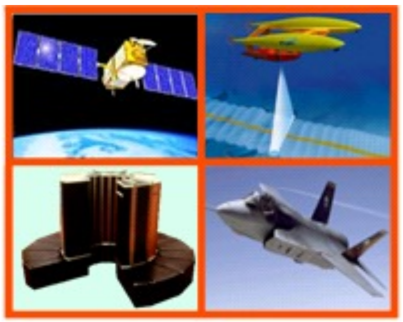
Mission-Critical Computing
NSF CENTER FOR SPACE, HIGH-PERFORMANCE, AND RESILIENT COMPUTING (SHREC)

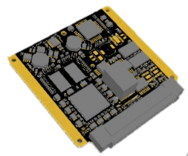
- **NSF Center for Space, High-Performance, & Resilient Computing**
  - Founded in Sep. 2017, replacing highly successful ***NSF CHREC Center***
  - Leading ECE research groups @ four major universities
    - **University of Pittsburgh** (lead)
    - **Brigham Young University** (partner)
    - **University of Florida** (partner)
    - **Virginia Tech** (partner)
- **Under auspices of IUCRC Program at NSF**
  - **Industry-University Cooperative Research Centers**
    - Fostering university, agency, and industry R&D collaborations
  - **SHREC is both National Research Center and Consortium**
    - University groups serve as research base (faculty, students, staff)
    - Industry & government organizations are research partners, sponsors, collaborators, advisory board, & technology-transfer recipients

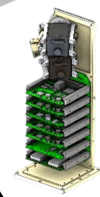
CHREC
NSF Center for High-Performance Reconfigurable Computing

NSF

Center Mission





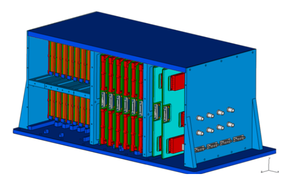


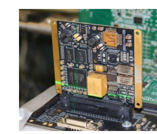
Space Computing

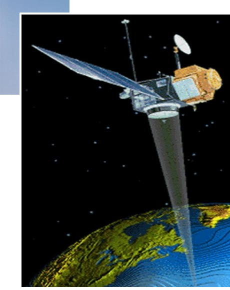
SHREC

High-Performance Computing

Resilient Computing







<u>Theme: Mission-Critical Computing</u>

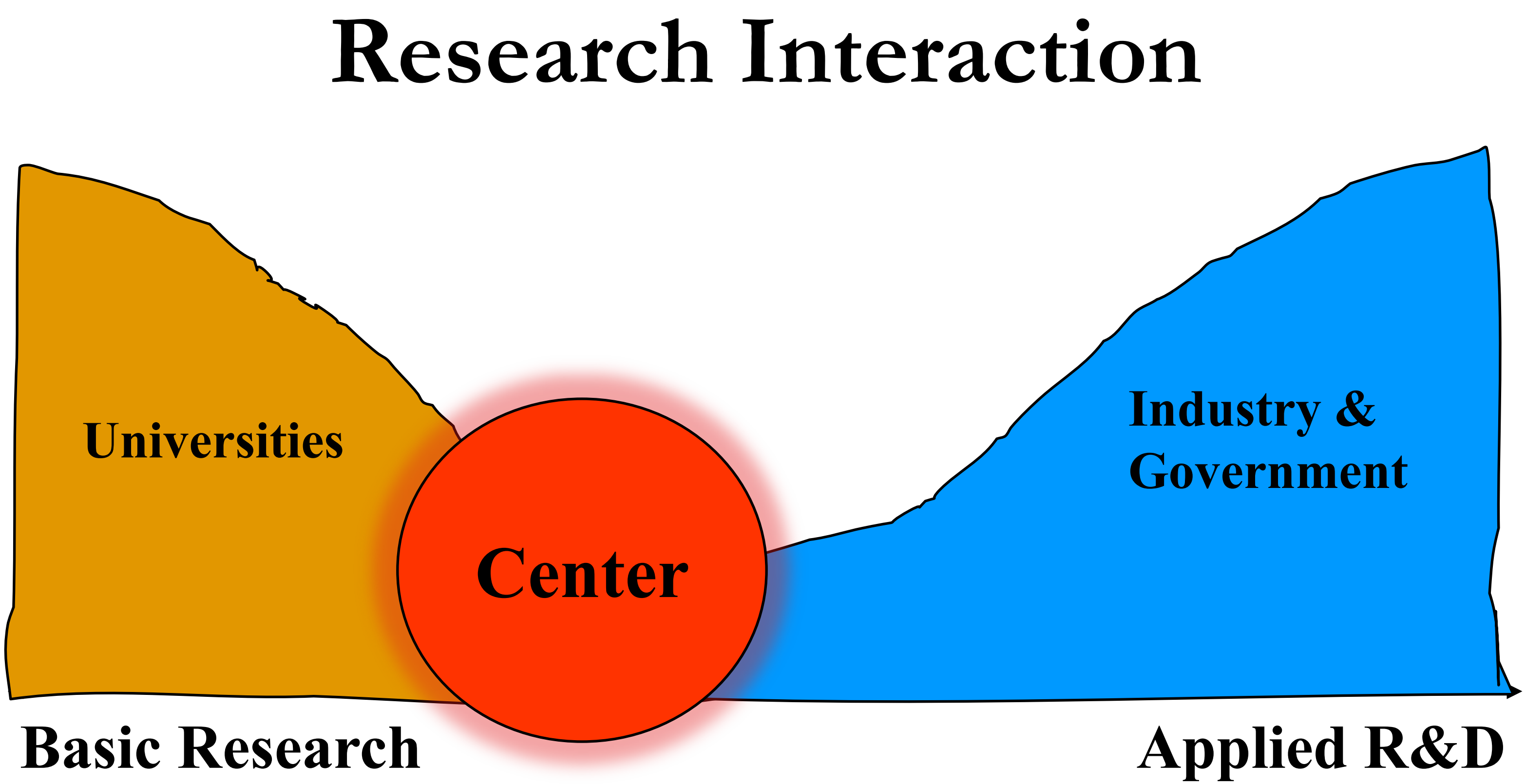
Basic and applied R&D to advance S&T on advanced computing.

Many common challenges, technologies, & benefits, in terms of performance, power, adaptivity, productivity, cost, size, etc.

From architectures to applications to design concepts and tools.

From spacecraft to supercomputers!

NSF Model for IUCRC Centers

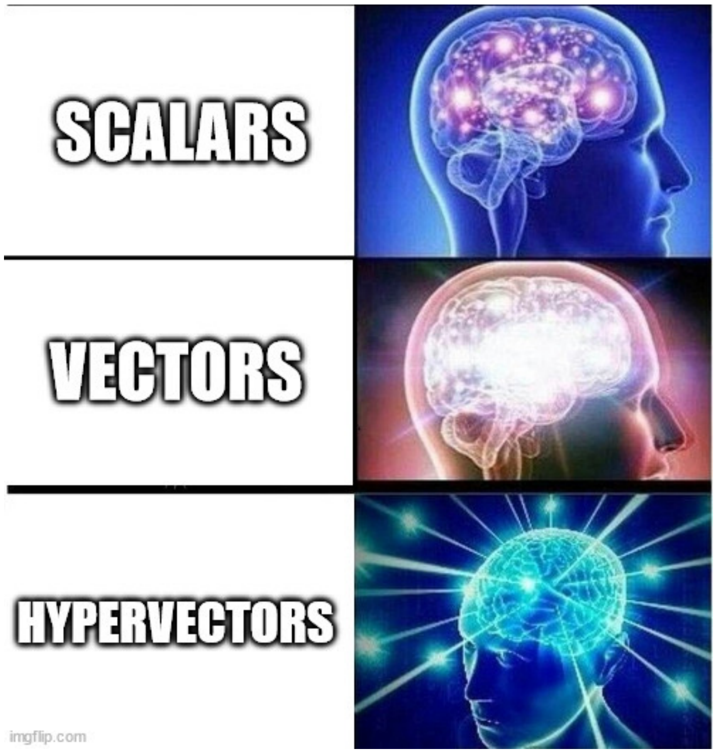


Hyperdimensional Computing

- Researchers observed brains represent and operate on data using randomly assorted neurons
- Hyperdimensional computing (HDC) is a machine learning paradigm that mimics this behavior using very large, randomly generated vectors
- HDC is easily pipelined and/or parallelized making it a good target for hardware acceleration

Hyperdimensional Computing

- Hypervectors
  - Very large vectors
  - Base representation of data in HDC
- "Curse of dimensionality"
  - In high-dimensional spaces, randomly generated vectors are nearly orthogonal
  - Can exploit this by representing semantically different things as different randomly generated vectors



HDC Operations

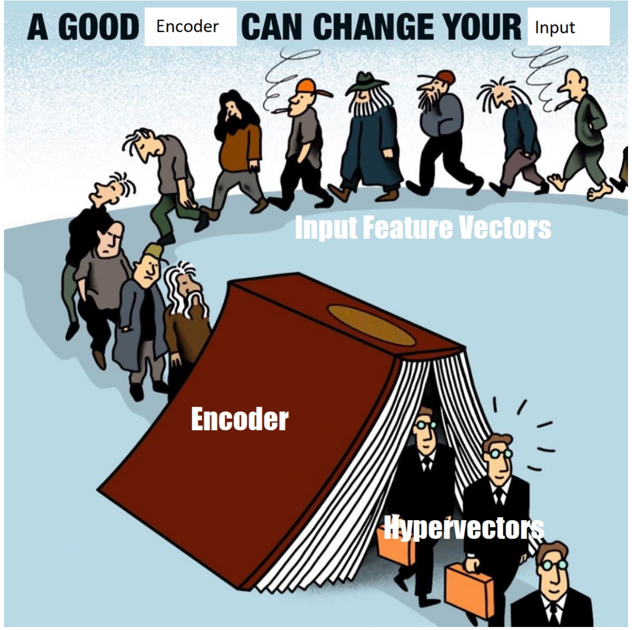
- Similarity
  - Measures "relatedness" of hypervectors
  - $\delta(A, B) \approx 0$ means A and B are unrelated
  - $\delta(A, B) \gg 0$ means A and B are related
  - Often implemented as cosine similarity $\delta(A, B) = \frac{A \cdot B}{|A||B|}$
- Bundling
  - Combines hypervectors into a hypervector that is similar to the inputs
  - Often implemented as elementwise addition

HDC Operations

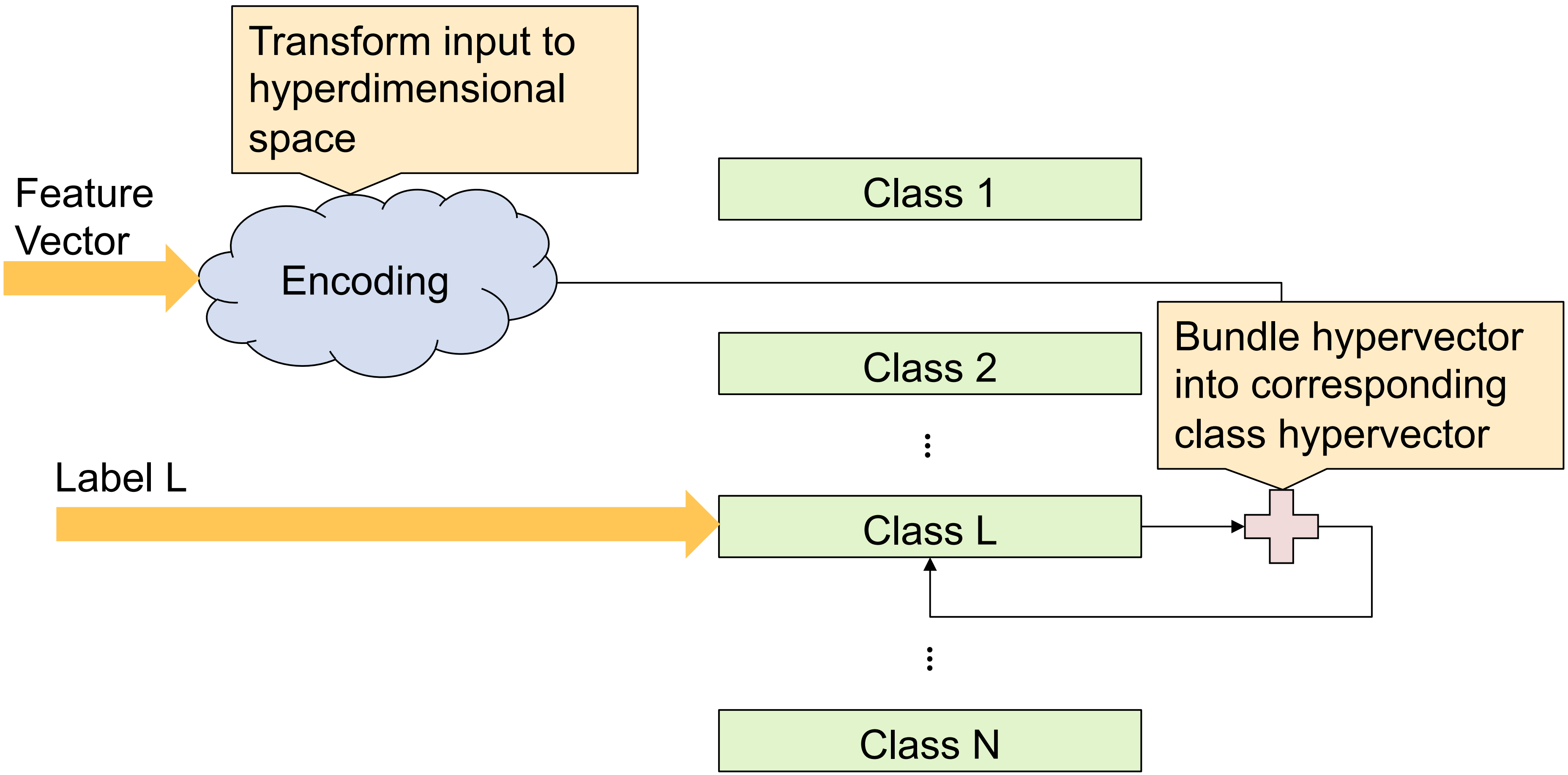
- Binding
  - Combines hypervectors to create a hypervector that is dissimilar to the inputs
  - Think of it as a generalized cross product
  - Often implemented as XOR for binary hypervectors
- Permutation
  - Rotates the elements in a hypervector
  - Creates a hypervector that is dissimilar to the inputs
  - Often used to encode positional or temporal data

HDC Learning

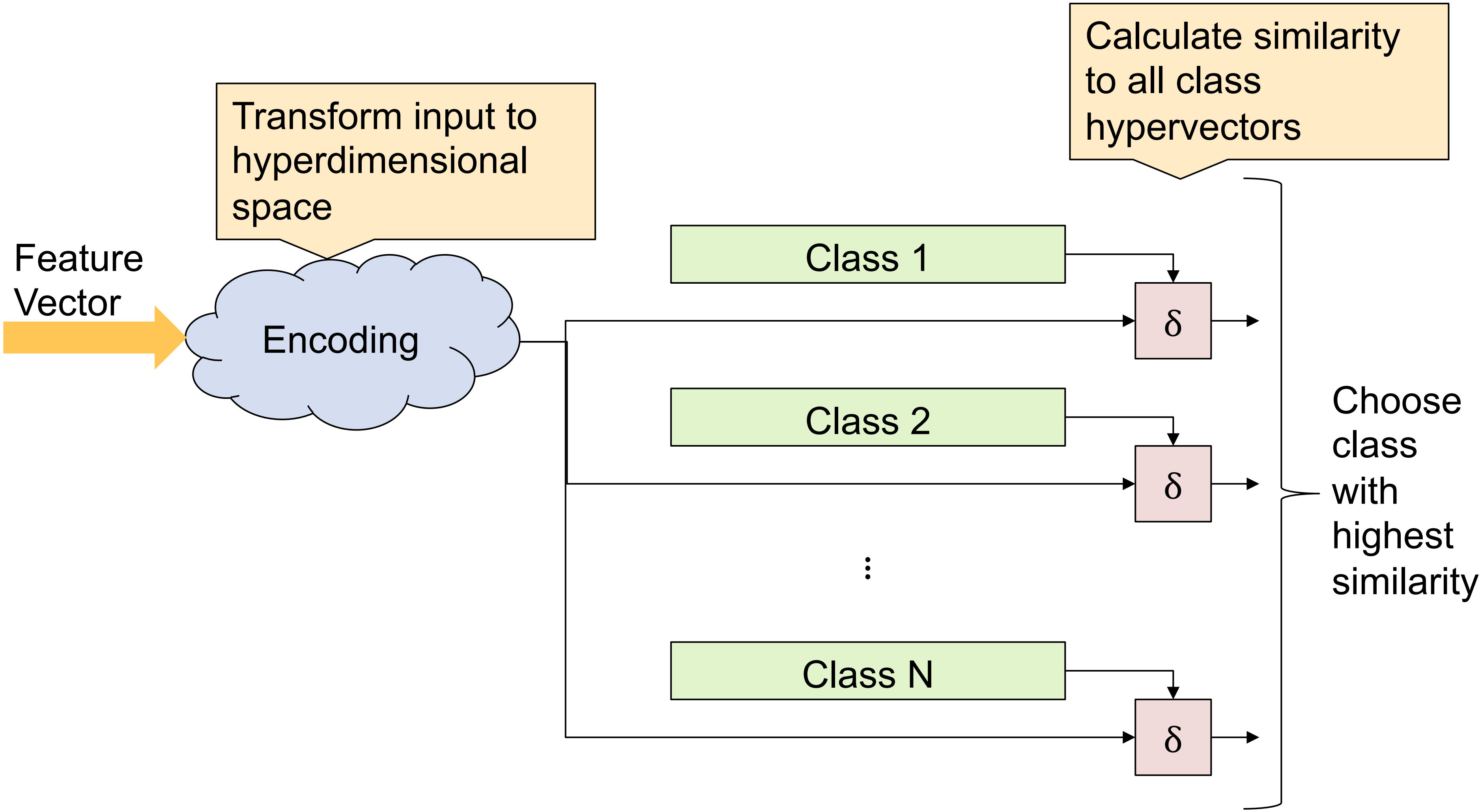
- Encoding
  - Uses randomly generated basis hypervectors to map feature vectors to hyperdimensional space
  - Implementation is dependent on the application
  - Implementations often include look-up tables or non-linear transforms
- Each class is represented as a hypervector



HDC Training



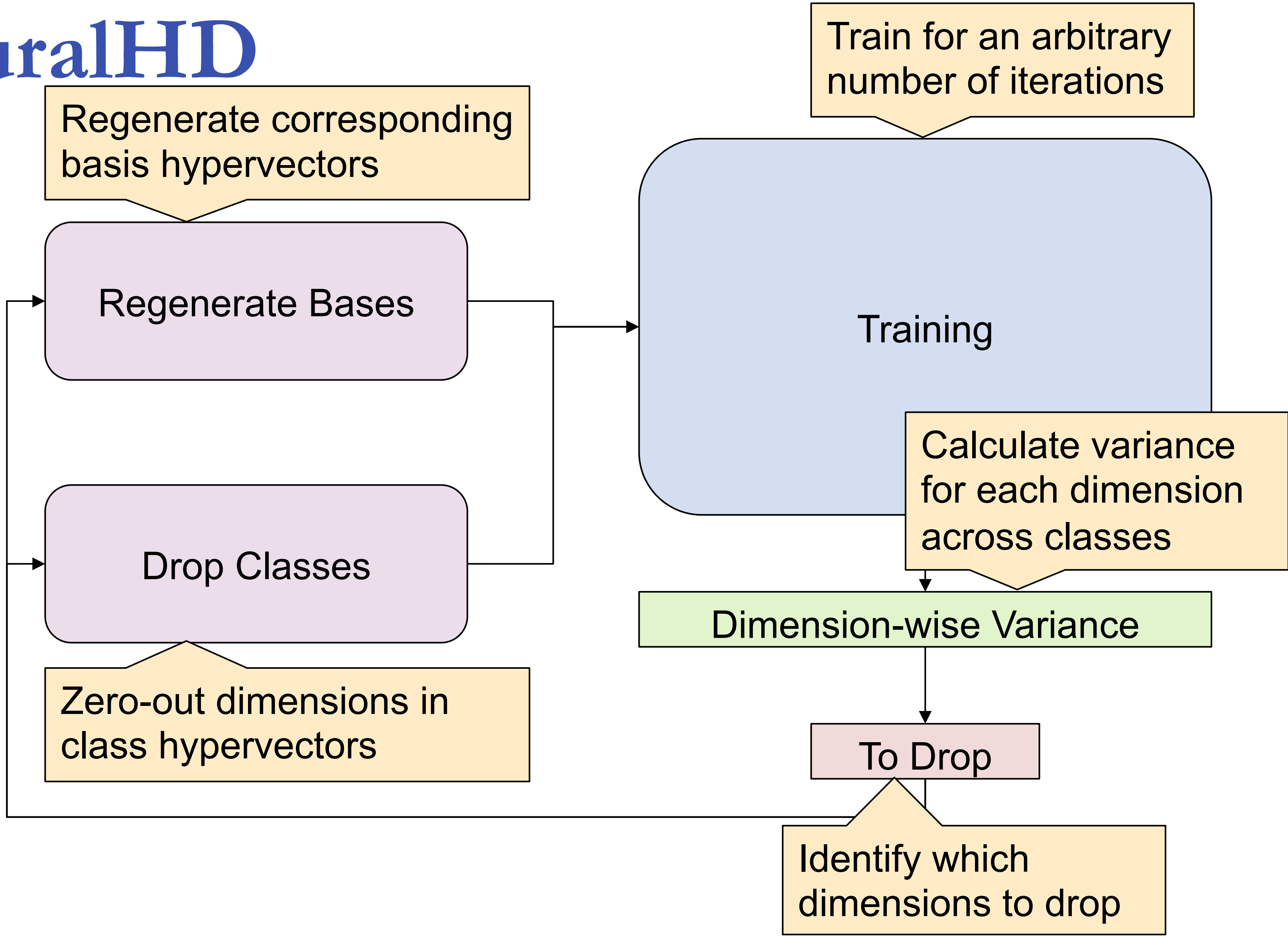
HDC Inference



NeuralHD

- Novel training algorithm
  - Updates encoding bases to optimize the encoding scheme for the desired application
  - Based on neural regeneration
- Encoder optimization
  - Calculates dimension-wise variance of class hypervectors
  - Lowest variance dimensions are zeroed in all classes
  - Bases corresponding to the lowest variance dimensions are regenerated

NeuralHD



Regenerate corresponding basis hypervectors
Train for an arbitrary number of iterations
Regenerate Bases
Training
Calculate variance for each dimension across classes
Drop Classes
Dimension-wise Variance
Zero-out dimensions in class hypervectors
To Drop
Identify which dimensions to drop

oneAPI

- Open, multiarchitecture accelerator framework
- Uses SYCL and C++
- Single-source development
- Offers various libraries with pre-optimized functions

oneAPI Execution

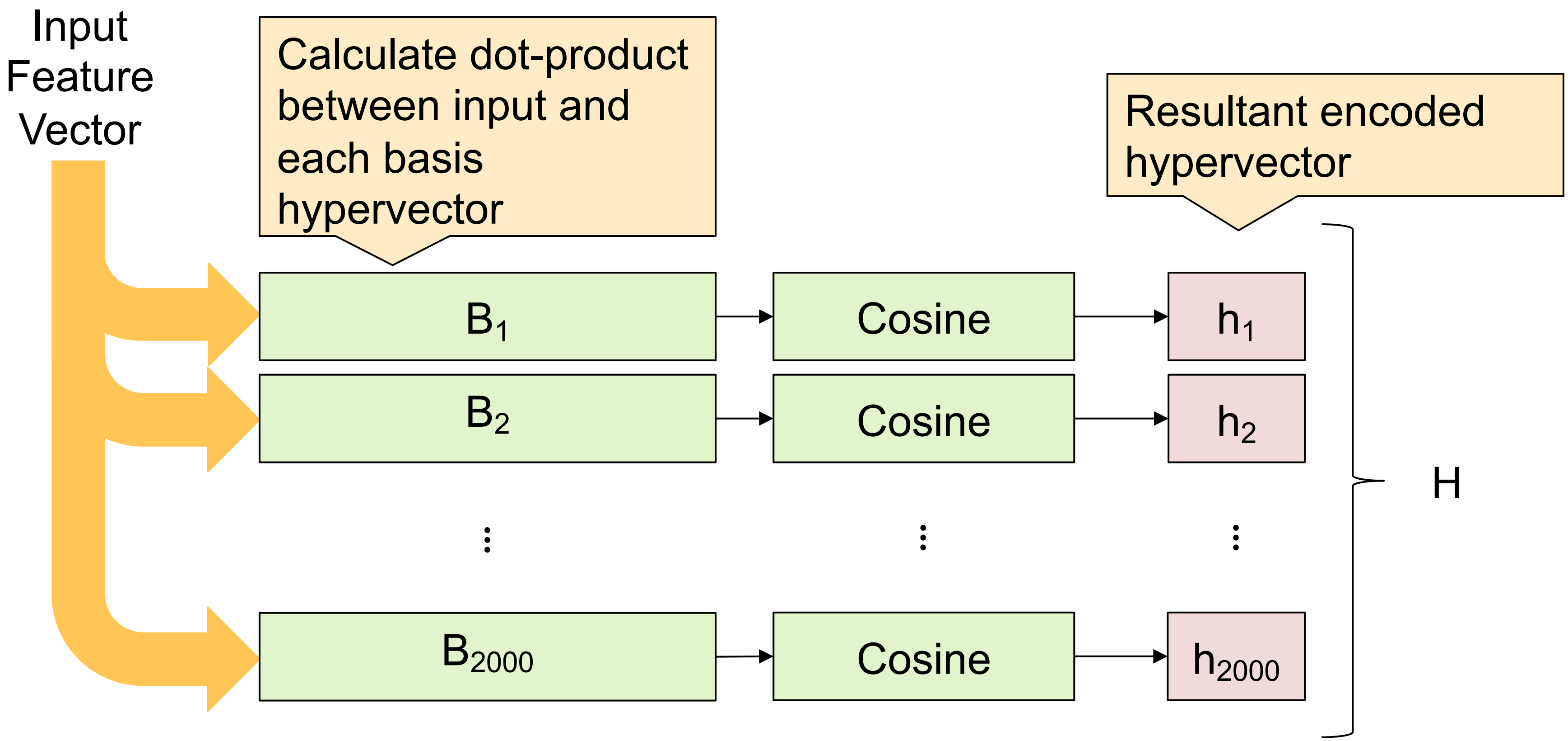
- Command queues
  - Host uses queues to send command groups to accelerator
- parallel_for
  - GPUs & CPUs: data-parallel execution of command group using threads
  - FPGAs: autopipelined loop
- single_task
  - Command group is executed as a single thread
  - Mostly used with FPGAs

oneAPI Memory Management

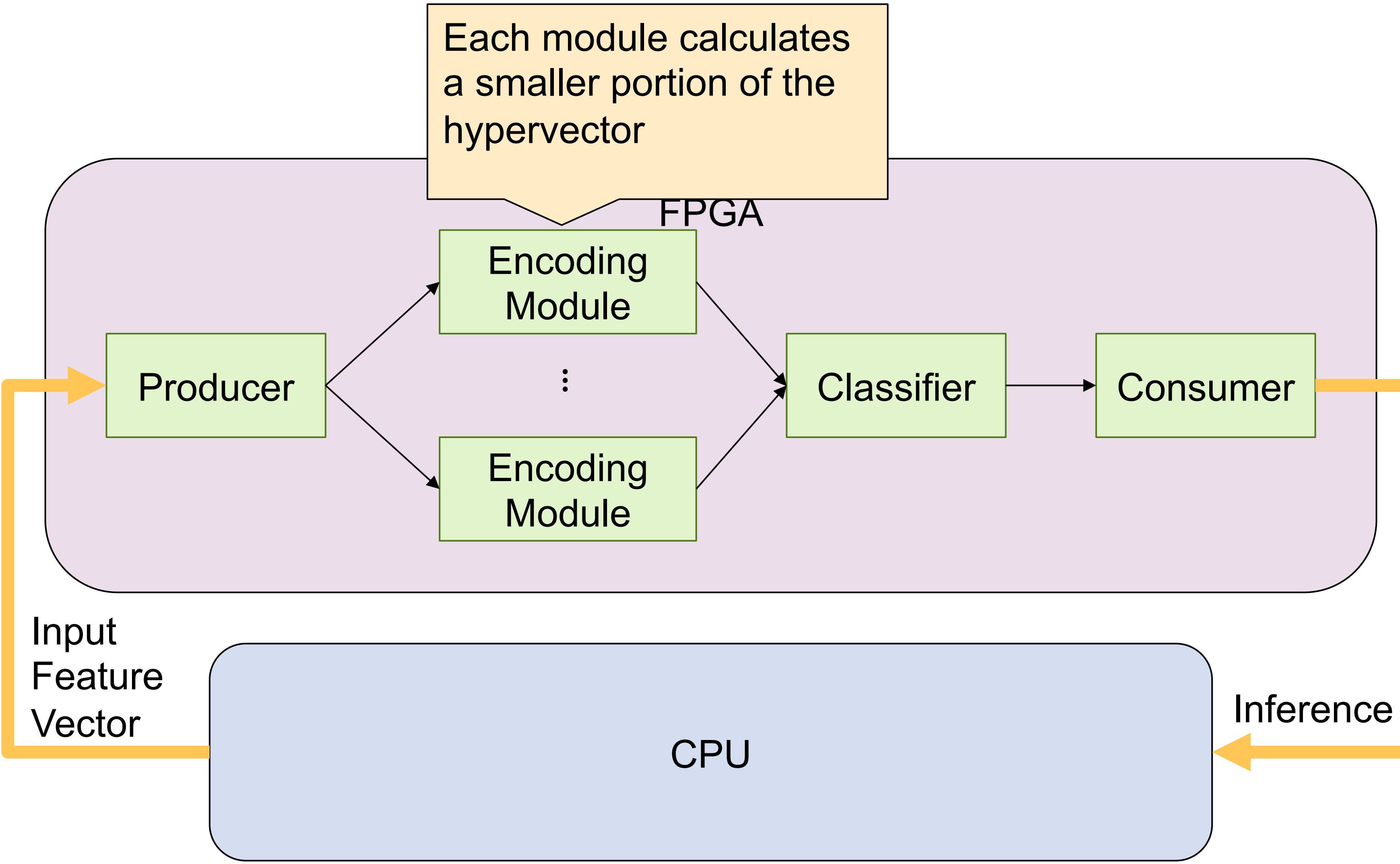
- Unified Shared Memory (USM)
  - Uses pointers
  - Explicit data movement: developer specifies when data is moved
  - Implicit data movement: data movement abstracted away
- Buffers and Accessors
  - Buffers are wrappers around data
  - Accessors are used to access the data in buffers
  - Abstracts away data movement
- Pipes
  - Only for FPGAs
  - Uses on-device FIFOs to pass data between kernels

HDC Model

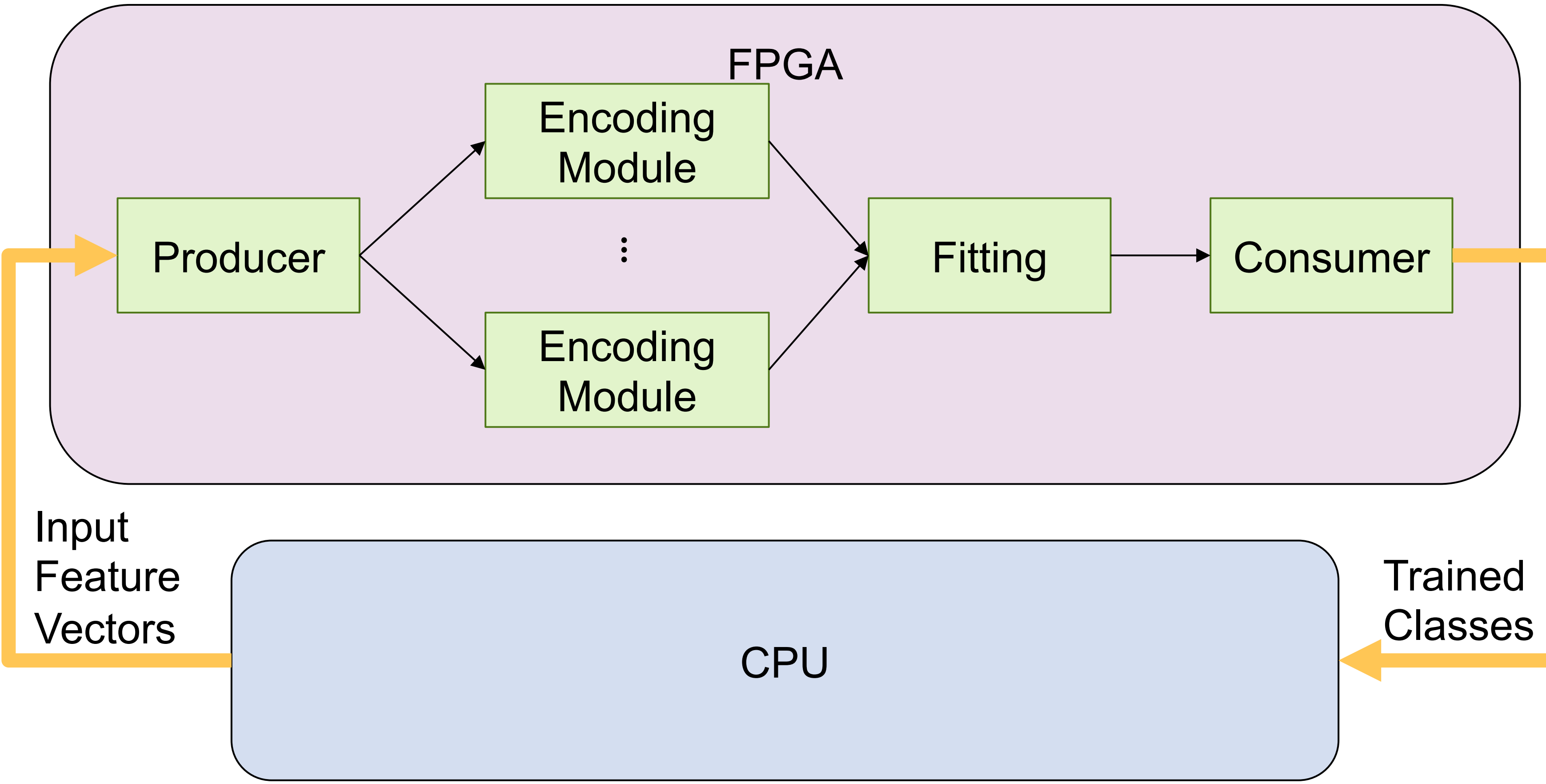
- 2000 hyperdimensions of FP32
- Encoding scheme



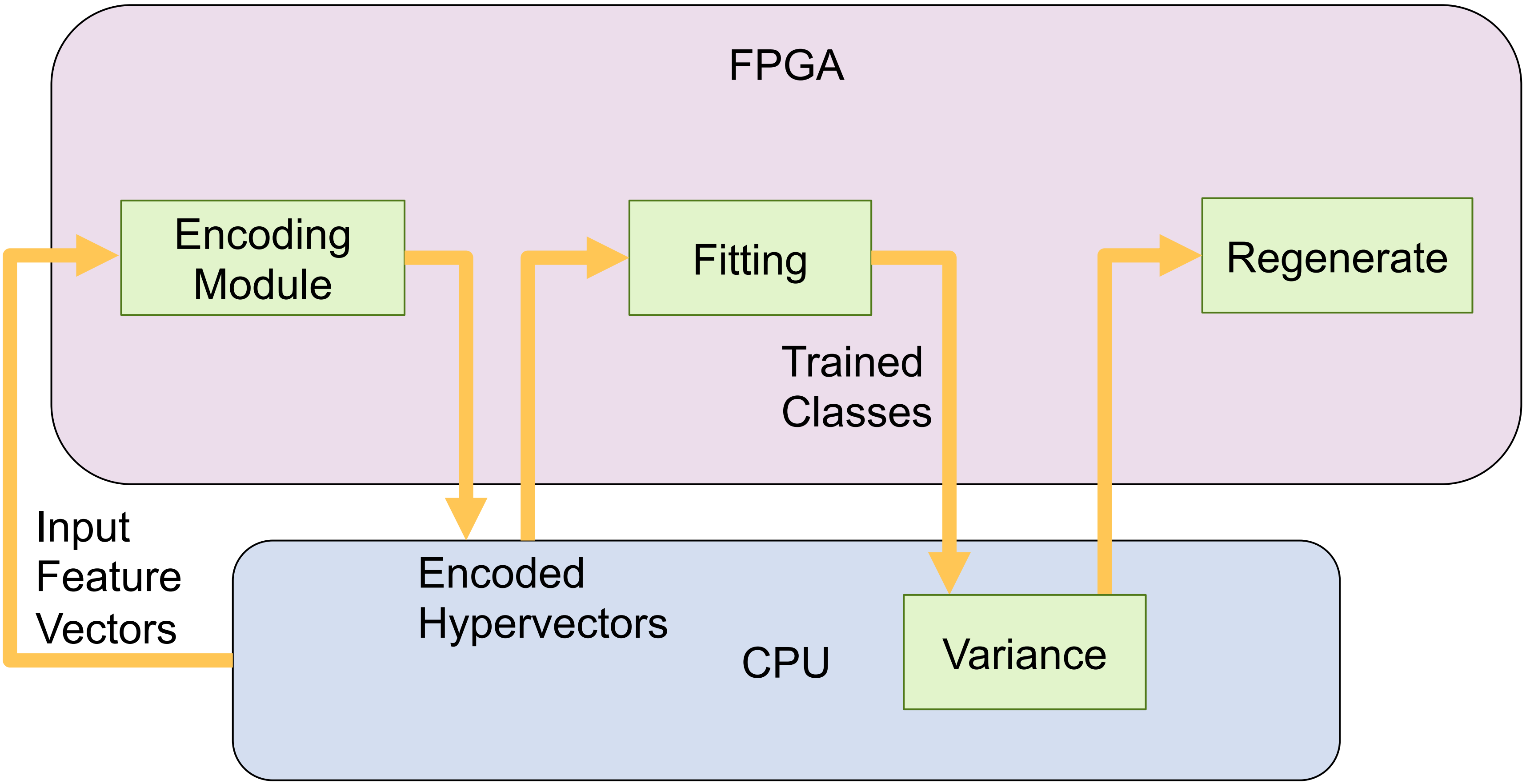
FPGA Inference Design



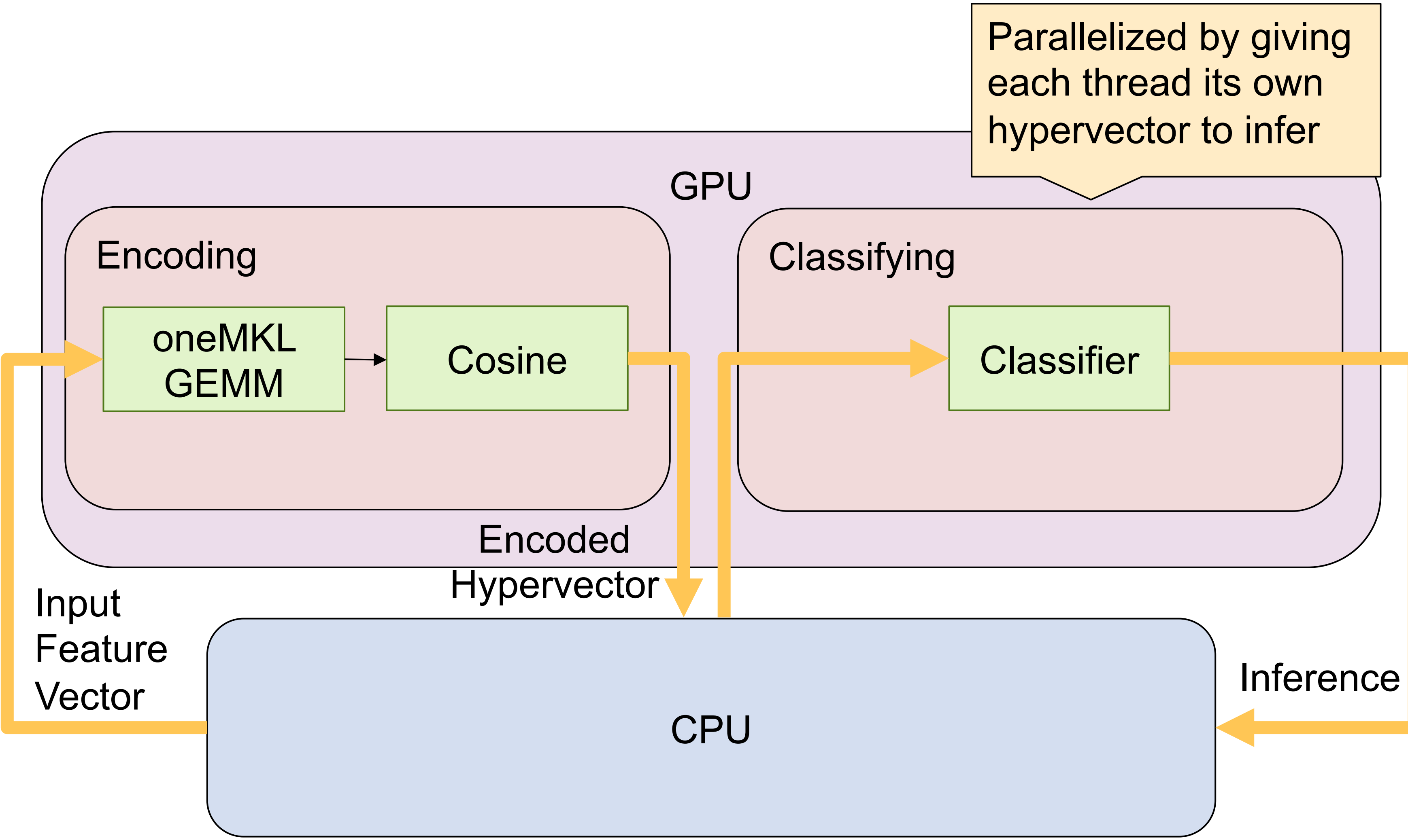
FPGA Single-Pass Training Design



FPGA NeuralHD Design

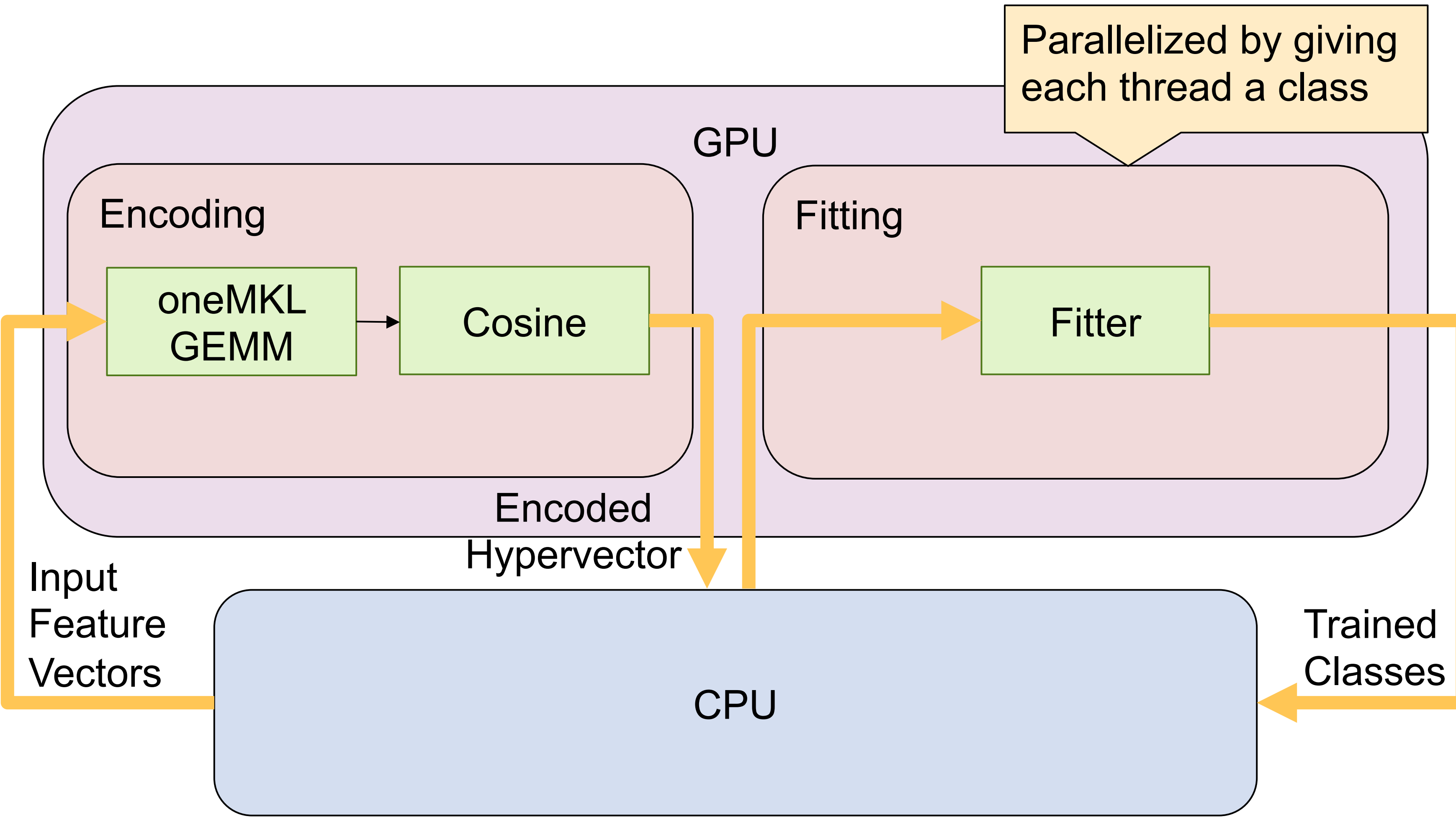


GPU Inference Design



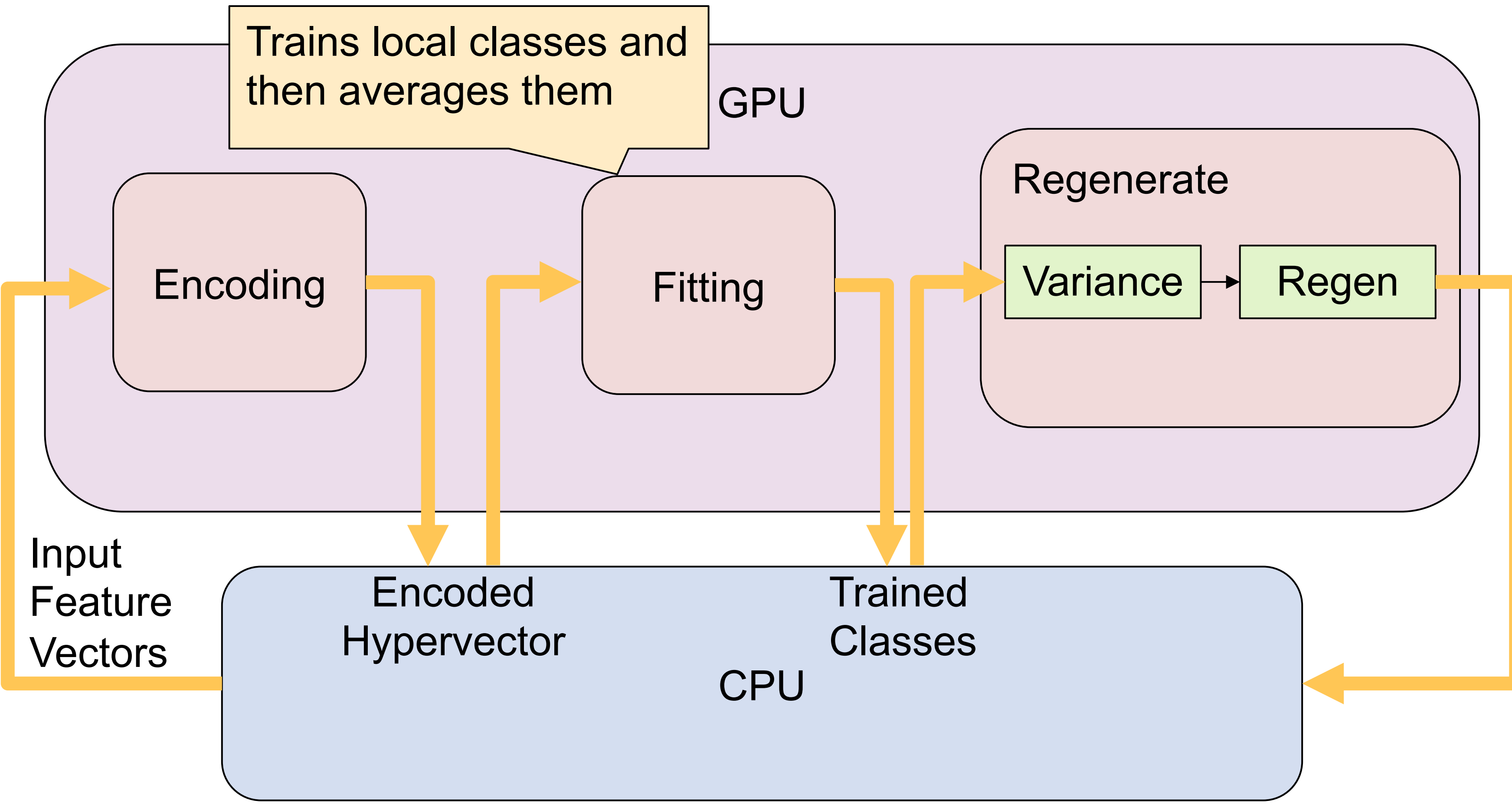
oneMKL: Intel® oneAPI Math Kernel Library
GEMM: GEneral Matrix Multiplication

GPU Single-Pass Training Design



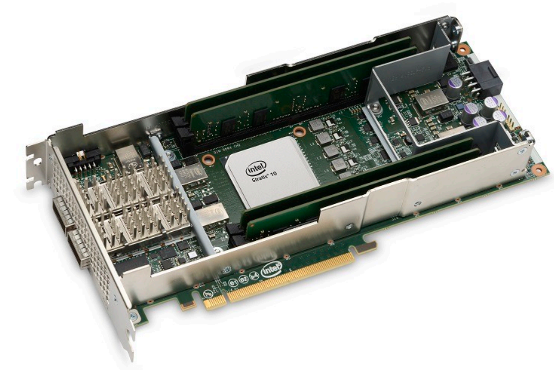
oneMKL: Intel® oneAPI Math Kernel Library
GEMM: GEneral Matrix Multiplication

GPU NeuralHD Design



Hardware

- Intel Stratix 10 (PAC D5005)
  - 16 GB DDR RAM
- Intel UHD 630
  - Integrated GPU
  - 32 GB DDR RAM
- Intel Xeon Platinum 8256
  - 3.8 GHz
  - 4 Cores
  - 192 GB DDR RAM

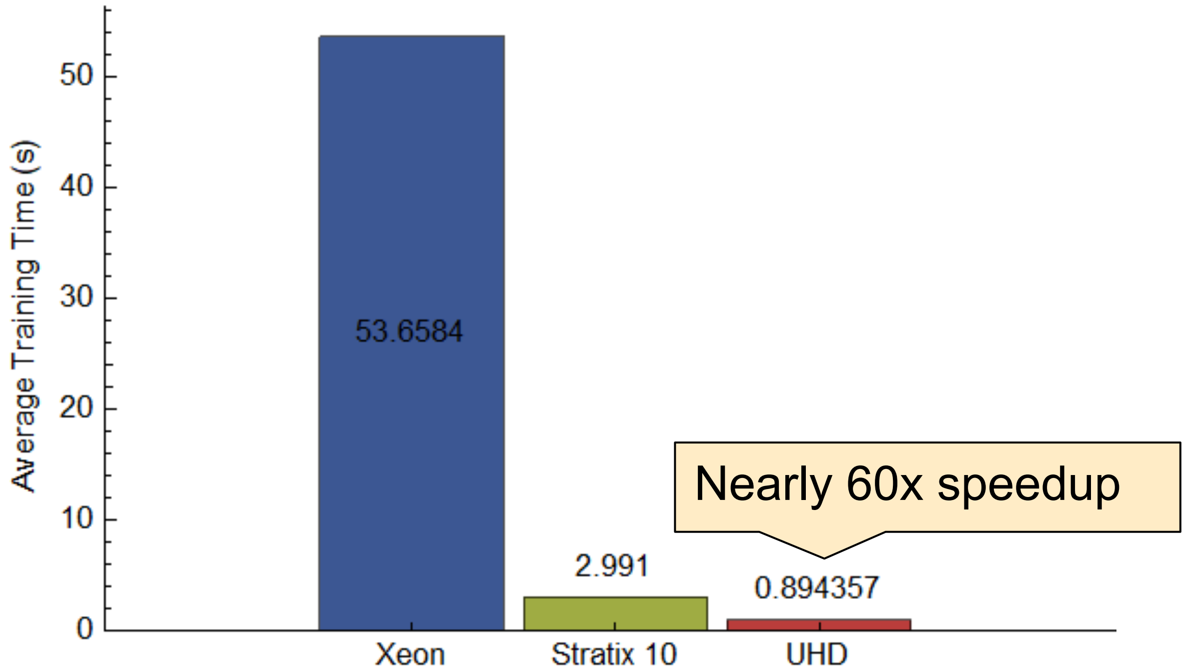


Experiment

- Benchmark single-pass training, NeuralHD training, inference latency, and throughput
- Used MNIST handwritten digits dataset
  - 60000 training images, 1000 test images
  - 28x28 monochrome images

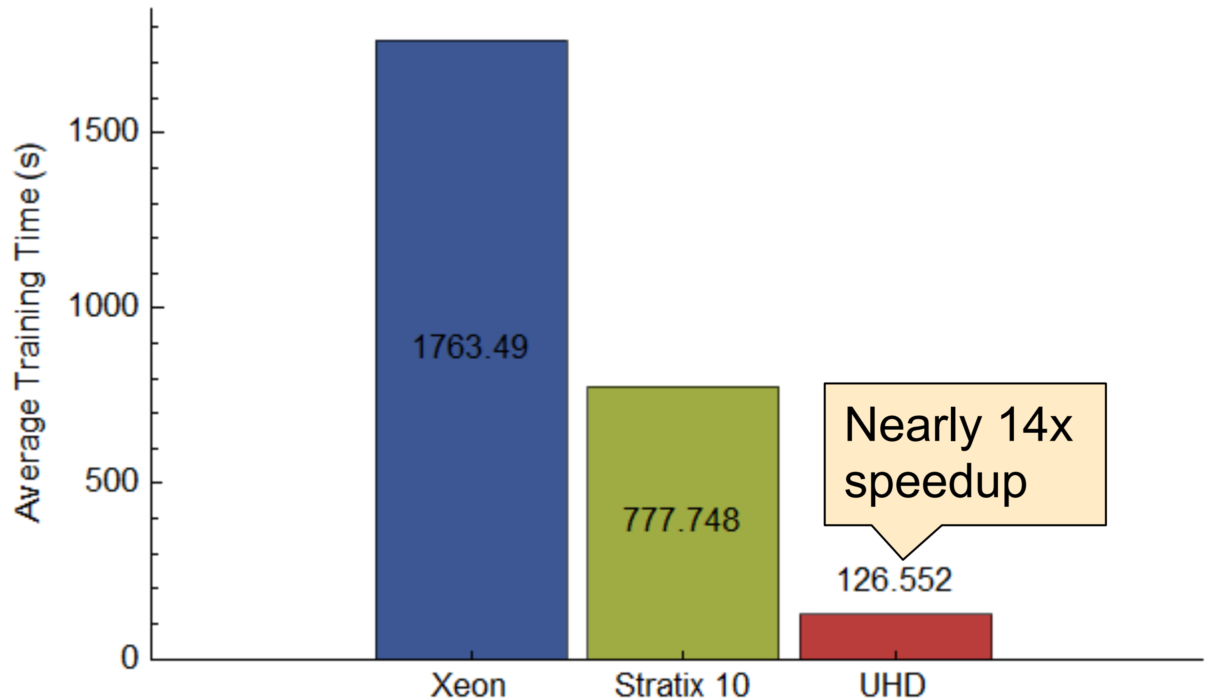
Single-Pass Training

- All models achieved ~89% accuracy



NeuralHD Training

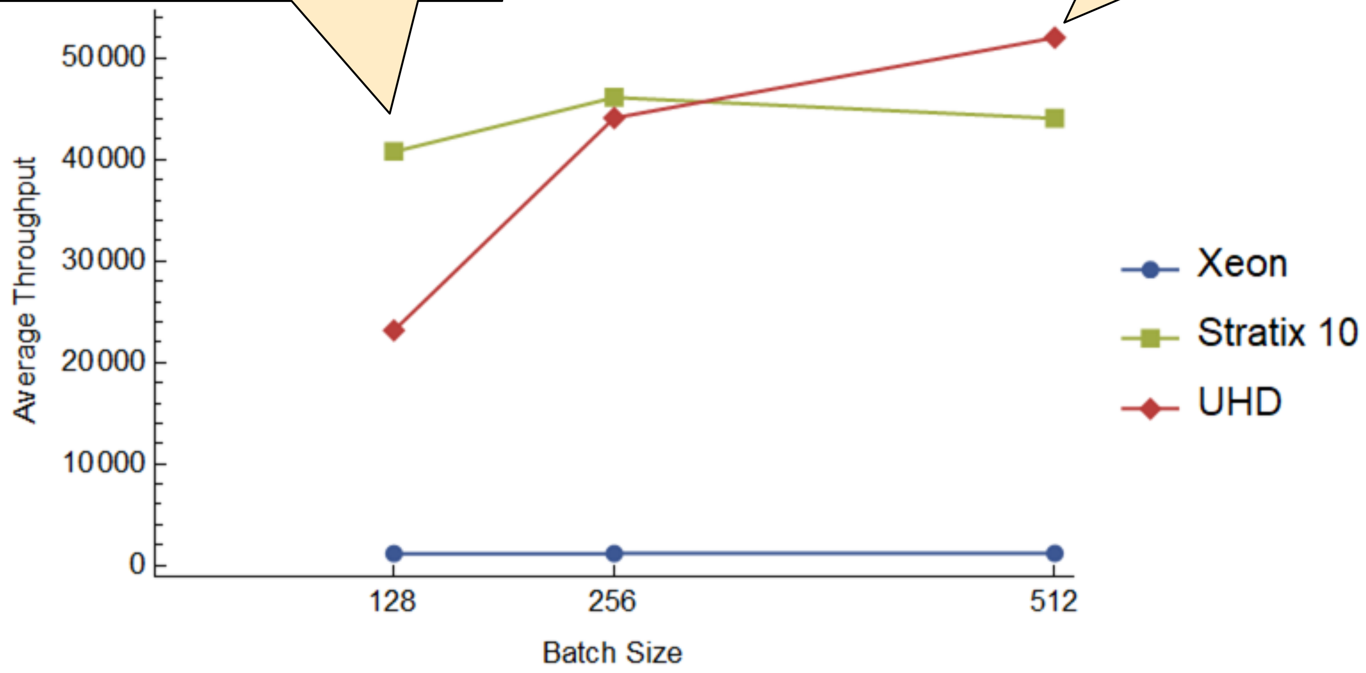
- FPGA and CPU achieved ~97% accuracy
- GPU achieved ~94% accuracy
- Not as great of speedup compared to single-pass



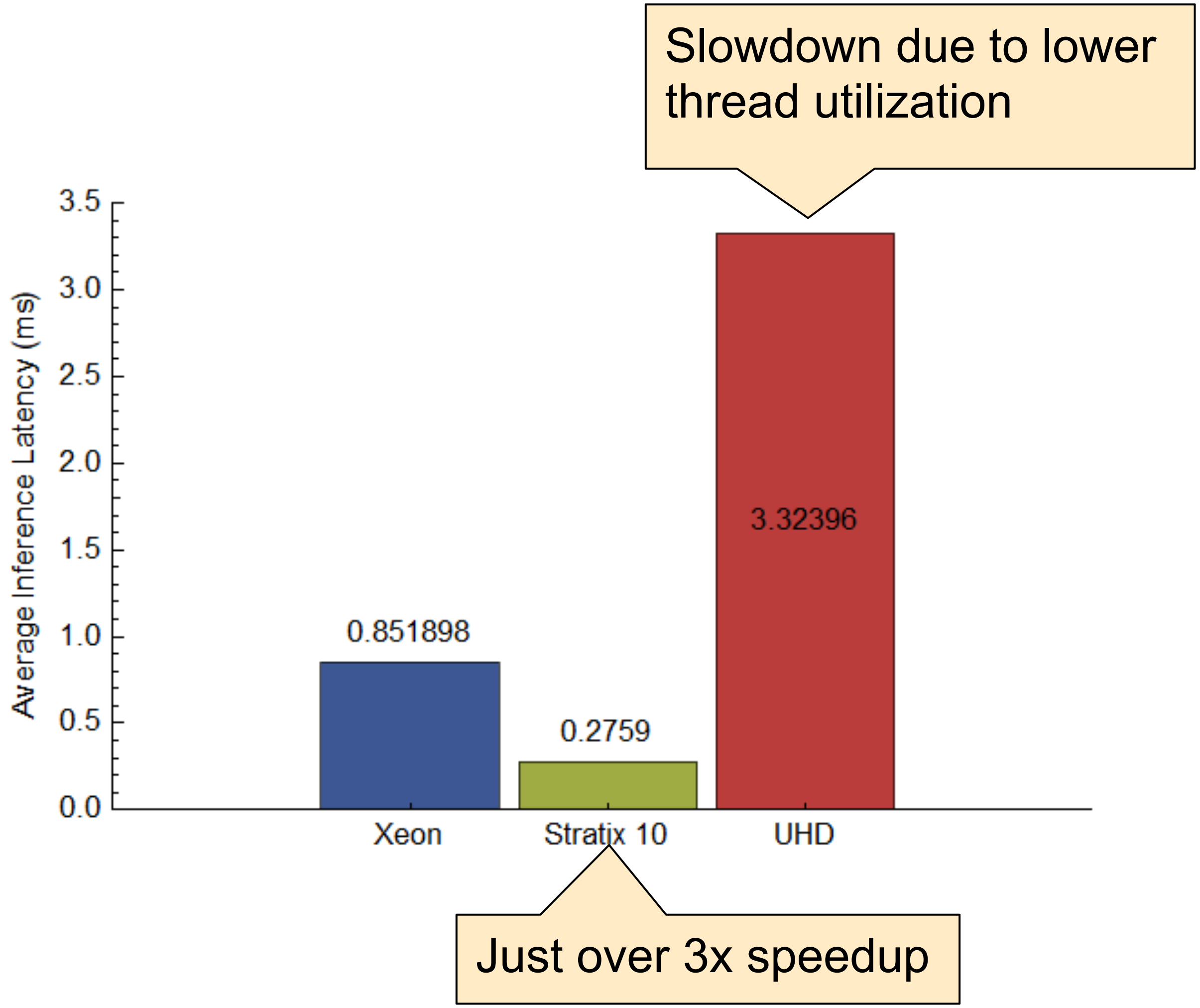
Inference Throughput

FPGA has highest throughput for smaller batch sizes

GPU achieves highest throughput once all threads are being used



Inference Latency



Conclusions

- GPU achieves greatest speedups for throughput, single-pass training, and NeuralHD training
  - All of these tasks demand high throughput
  - UHD architecture and GPU HDC implementation has higher degree of parallelism leading to greater throughput
- GPU exhibits slowdown for inference latency
- FPGA achieves greatest speedup for inference latency
  - Latency likely able to be improved further through quantization

Questions?