IMPLEMENTATION OF EFFICIENT, LOW POWER DEEP NEURAL NETWORKS ON NEXT-GENERATION INTEL CLIENT PLATFORMS

What is GNA?
Low power neural co-processor for continuous inference at the “edge”
Designed for Intel® Quark™, Intel Atom™, and Intel® Core™ based devices
Runs while application processor is in low power sleep state
Interfaces to system memory or private memory avoiding CPU cache pollution

How does GNA work?
Neural network topology stored in memory as list of layer descriptors
Layer types: affine, diagonal affine, Gaussian mixture model, recurrent, convolutional1D, transpose, copy
Activation function: piecewise linear (PWL) approximation
All-integer math (inputs, outputs, weights, biases)
Batching of layer inputs for better throughput
Optional on-the-fly pruning (affine layer)
Complex graphs (e.g., LSTM, GRU, TDNN) constructed from basic layer types
Stream processing model
• App processor configures memory, starts GNA, and sleeps or does useful work
• GNA signals when forward propagation is complete

How is GNA used?
Start with floating point neural network trained in framework of choice
Import using Intel® Deep Learning SDK Deployment Tool or Kaldi example
Link with Intel® Deep Learning SDK Inference Engine or GNA native library

GNA native library options
• Firmware API (for Intel® Quark™ running real-time operating system)
• Middleware API (for Intel Atom™ and Intel® Core™ running Linux, Windows)

API Function | Description
--- | ---
GNAOpen | acquire handle to GNA device
GNAInfo | return release handle to GNA device
GNARes | allocate memory (and pin it if cannot be swapped out)
GNAFree | free GNA memory (after unpinning)
GNAFFIC | propagate inputs through all layers of network
GNAWait | wait until propagation request completes or timeout

Complexity is hidden in layer descriptor list construction
• Handled transparently by Intel® Deep Learning SDK model optimizer
• Full control via GNA native library API

Intel® Deep Learning SDK Deployment Tool
Enables full utilization of all inference while abstracting HW from developers
Optimize:
• Imports trained models from all popular DL framework regardless of training HW
• Model Canonicalization, Compression and Quantization Deploy:
  • One API across all Intel HW and systems
  • Friendly Inference solution: low footprint, easy API, control monitoring
  • Functional Safety
  • Optimizes Inference execution per target hardware under-the-hood

Erase of use • Embedded friendly • Extra performance boost

What You Are Seeing: Live ASR Demonstration
Video playing TED talk by Aimee Mullens: The opportunity of adversity
Customer speech recognizer performing local real-time transcription
Acoustic model: 6-layer DNN w/ 2048 hidden nodes trained on 3000 hrs
Acoustic likelihood scoring is selectable:
• Native (customer optimized), GNA SW emulation (CPU), or GNA (HW)
Performance monitor shows drop in CPU residency with GNA HW
Corresponding drop in power consumption not shown

What You Are Seeing: Performance Demonstration
Real-time visualization of acoustic log-likelihoods
• DNN: additional layer added mapping outputs to phones
• Ordered by class: non-speech, unvoiced fricatives, voiced stops, unvoiced stops, voiced fricatives, liquids & glides, nasals, front vowels, mid vowels, back vowels, diphthongs
• LSTM: CTC-trained phone outputs in natural order
Repeating cycle: score on CPU, score on GNA, ...
Shows difference in scoring speed between CPU and GNA HW

Batch scoring of 7-layer 2048 hidden node DNN
Log-likelihoods color coded
CPU utilization in Perfmon
Scoring on GNA is ~3 times faster than 1.1GHz Atom CPU in this configuration**

**Note that other configurations (e.g., higher CPU clock, use of more CPU cores, etc.) may improve performance at cost of higher power consumption