Hyper-Dimensional Analytics of Video Action at the Tactical Edge

Michael Isnardi[#*], Saurabh Farkya[#], Indu Kandaswamy[*], Aswin Raghavan, David Zhang, Gooitzen van der Wal, Joe Zhang, Zachary Daniels, Michael Piacentino
Center for Vision Technologies, SRI International, Princeton, NJ
{Michael.Isnardi, Saurabh.Farkya, Indhumathi.Kandaswamy, Aswin.Raghavan, David.Zhang, Gooitzen.Vanderwal, Yuzheng.Zhang, Zachary.Daniels, Michael.Piacentino}@sri.com

***Abstract*—We review HyDRATE, a low-SWaP reconfigurable neural network architecture developed under the DARPA AIE HyDDENN (Hyper-Dimensional Data Enabled Neural Network) program. We describe the training and simulated performance of a feature extractor free of multiply-accumulates (MAC) feeding a hyperdimensional (HD) logic-based classifier and show how performance increases with the number of hyperdimensions. Reconfigurability in the field is achieved by retraining only the feed-forward HD classifier without gradient descent backpropagation. We show performance for a video activity classification task and demonstrate retraining on this same dataset. Finally, we discuss a realized FPGA architecture that achieves 10x smaller memory footprint, 10x simpler operations and 100x lower latency/power compared to traditional deep neural networks.**

Keywords*—DARPA; HyDDENN; FPGA; hyperdimensional computing; video activity classification; training at the edge; reconfigurability

I. Introduction

We provide a description of SRI's Hyper-Dimensional Reconfigurable Analytics at the Tactical Edge (HyDRATE) system, a non-MAC (free of floating-point Multiply-ACcumulate operations) deep neural network (DNN) architecture combined with hyperdimensional (HD) computing. We show non-MAC results for video analytics with performance on par with MAC-based DNNs. We believe this is the first non-MAC HD architecture to perform 3D (video) analytics. We also show initial results for gradient-free reconfiguration by downstream tasks directly on the edge. Such a method could leverage the huge success of pre-trained encoders (e.g. BERT, ImageNet) that are easily fine-tuned to different target tasks by fine-tuning a decoder. We address the challenges of edge-friendly deployment (non-MAC operations), edge-friendly training of an HD decoder, and describe an underlying hardware implementation that can support video rates. In this paper, we present accuracy and performance metrics in comparison to a state-of-the-art (SoA) MAC-based NN and discuss details of its FPGA implementation.

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR00112090037. The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.
[#] indicates equal contribution; [*] indicates co-Principal Investigator

II. HyDRATE Architecture

The overall HyDRATE architecture is a novel sequence-to-sequence (seq2seq) model augmented with an attention mechanism (Fig. 1). The left side is the non-MAC encoder that acts as a feature extractor; the right side is an HD decoder that acts as a classifier. The network performs image- or video-based activity classification. The input is a sequence of video frames. The shallow Non-MAC Sparse Encoder compresses an image into a set of feature vectors which, along with predicted label of the previous frame, are converted to HD vectors through *bind* and *bundle* binary logic operations[1]. The output of the HD mapping in the decoder is compared with a set of class exemplars, hyper-vectors obtained in the training stage in the HD decoder. The architecture is designed so that fast, forward-only (gradient-free) retraining of the HD decoder can be performed for recognition of new activity classes.

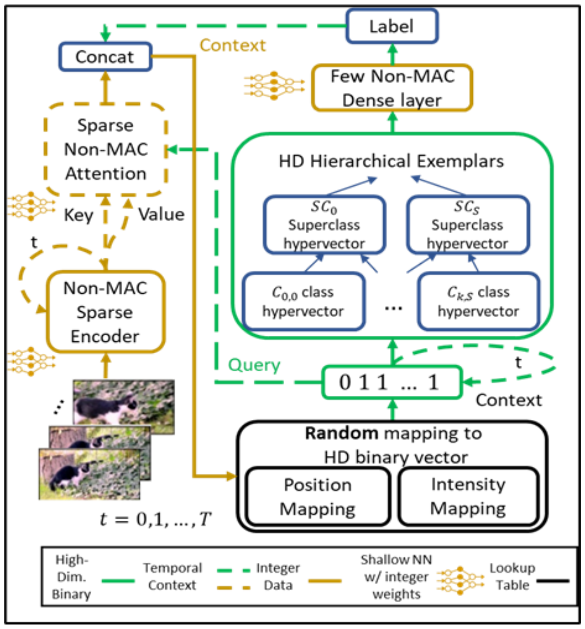


Fig. 1. Overall HyDRATE Architecture

The implemented HyDRATE architecture is illustrated in Fig. 3. A sequence of video frames is fed to the Non-MAC quantized neural network (NN) encoder. The 4-bit NN is a distilled network which learns all the weights from a 32-bit teacher network. It contains a spatial quantized ResNet50 and a temporal LSTM network. All weights are power-of-2, and multiplications are replaced by shift operations. This shallow Non-MAC Sparse Encoder compresses a frame into a vector of feature embedding, which, along with the feature vectors from the previous frames, are converted into HD vectors by encoding the position and value of each vector. The HD encoder uses binary XOR to *bind* position HD and value HD into a single HD vector, and then *bundle* (majority voting per bit) all the feature elements into one HD vector. The output of the HD mapping in the decoder is compared with a set of class HD exemplars obtained in the training stage from the HD classifier. The Hamming distance is calculated between the input HD and the HD exemplars stored in the associative memory, and the smallest Hamming distance determines the class of the input frame. For video-based activity recognition, we select an average of 12 temporal frames per time window and calculate the average Hamming distance. The result is used to classify the activity at the time.

Unlike traditional NN models, in which weights are learned by gradient descent back-propagation, HyDRATE only learns the feature extractor network based on this method and are fixed when the HD classifier is connected for classification. HD exemplars are generated using binary bundle operation. Since we use the UCF101 dataset, 101 HD exemplars are generated during training, and they are used in Hamming distance measure against new samples to classify each of them.

Based on this architecture, we will discuss our accomplishments to date, namely:

- Successful implementation of a method to convert a MAC-based DNN into a non-MAC DNN while minimizing degradation in accuracy [2].
- Successful training of an HD classifier whose input comes from the non-MAC encoder. *We discovered and demonstrated the need to train the HD classifier via an intermediate step of training a non-MAC classifier. In addition, our results show that learned low-bit quantization (HD Accuracy in Table 1) is better than any hand coded feature representation (Accuracy in Table 1).*
- Realization of a non-MAC field programmable gate array (FPGA) architecture whose low-SWaP metrics meet or exceed those listed in the BAA.
- Demonstration of the ability to reconfigure the HD classifier to recognize new classes while keeping the non-MAC encoder fixed. The FPGA design supports this feature as well.

We also discuss the key Phase 1 performance metrics shown in Table 2 and highlight progress towards a real-time FPGA breadboard demo that is being implemented in Phase 2.

III. Creating the Non-MAC DNN

We create a non-MAC shift-accumulate-only DNN by implementing a distilled teacher-student network as shown in Fig 2. The teacher NN is a 32-bit network, and the student NN is a quantized network with the low-precision weights containing the power-of-2 coefficients. We consider three loss functions, the standard 32-bit NN cross-entropy loss $L(W)$, the low-precision distillation loss $D(W,\theta)$, and the bit loss $B(\theta)$:

$$\text{Loss} = L(W) + \lambda_D D(W, \theta) + \lambda_{bwt} B(\theta) \qquad (1)$$

The quantization function used to convert weights into power-of-2 equations are as follows:

$$q(W;\, \theta) = \theta_1 + \theta_2 \log_2 |W| \qquad (2)$$

Quantized weights: $W_{LP} = sign(W) 2^{round(\theta_1 + \theta_2 \log_2 |W|)}$ (3)
Number of bits: $1 + \lceil \log_2(M - m + 1) \rceil$, where (4) $m = \min_W round(q(W;\theta))$, $M = \max_W round(q(W;\theta))$, and $q(W, \theta)$ is the transformation of the parameters W to the power-of-2 index.

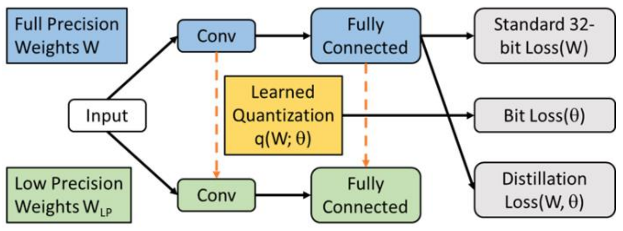


Fig. 2. Teacher student distilled network for quantized NN training. W is the weights of the 32-bit NN, and W_{LP} is of the quantized weights from Eq. 3. The dashed arrows indicate that the values of θ_1, θ_2 are learned for each layer. Both the number of bits per layer and the quantization levels are learned. Although the optimziation indicates that many layers can use fewer than 4 bits without the loss of performance, for FPGA design and implementation, we use 4 bits for all the layers.

This architecture learns both student and teacher models simultaneously from scratch. Once the training is done, we freeze the student network hence obtaining a non-MAC DNN (in power-of-2). This network is further used as feature extractor to perform HD computations. The first row in Table 1 is the baseline performance from the 32-bit NN. The rest of the rows are the quantized NN performance trained with different lambda. Note that the average bits of the NN is between 2 and 3 for the HD performance listed in the last column. The bold row is the model where we set all layers to be 4 bits in the hardware although the average number of bits is less than 4. In HyDRATE, the quantized NN uses ResNet50 as a feature extractor per frame, and LSTM to extract temporal information.

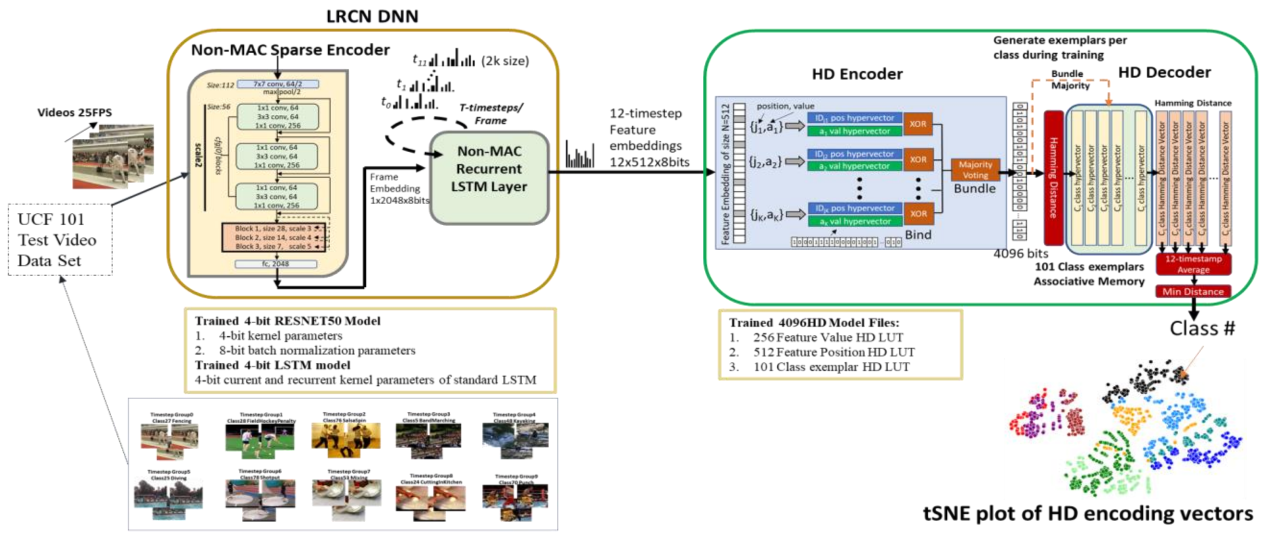
Table 1. Distillation NN training accuracy and bits

λ_{bwt}	Accuracy	Bits	HD Accuracy
	76	32 (baseline)	
0	62.1	3.3	66.7
8e-5	60.34	3.06	66.3
1e-4	60.34	3.03	66.03
2e-4	**56.56**	**2.79**	**63.9**
4e-4	51.0	2.68	58.44

IV. Training the HD Classifier

HyDRATE's HD backend is inspired by the VoiceHD architecture[1]. Different from VoiceHD, in which the features are encoded directly from the FFT of the time signal, HyDRATE takes the compressed feature vector (512x8 bits) per frame from the non-MAC NN, and the values and positions of each feature vector are encoded in HD. As shown in Fig. 3, the elements of the kth {index j, value a} pair in the feature vector are individually mapped into D-dimensional HD vectors $\{J_k, A_k\}$. The two HD vectors are then bit-wise combined to produce $T_k = J_k * A_k$, as per element representation, where * is the bitwise XOR function. Note only a small size (K+N) of distinct HD vectors (K=512 distinctive positions, N=256 HD distinctive values) are needed. Finally K HD vectors are *bundled* to generate an exemplar per image. More specifically, the *q*th image of the *i*th class can be encoded using a majority operation $Sq^i = \mathrm{MAJ}[T_1, T_2,\ldots,T_K]$. The class index of the sample is computed by the minimum of Hamming distances bweteen the encoded vector and the class exemplars. For UCF101 dataset, 101 HD exemplars are generated in the training from the training dataset. To train the HD model, we use majority voting to bundle all HD vectors of a class *i* from the training samples to form a *class* HD exemplar $C^i = \mathrm{MAJ}[S1^i\ldots,SQ^i]$, where Q is the number of training examples of that class. For video activity recognition, HyDRATE combines 12 frames per temporal window and averages the Hamming distance output to classify the activity. This improves the perfomance compared to a single image. Note that HD training is performed using feedforward bitwise logical operations and traditional backpropagation is not used.

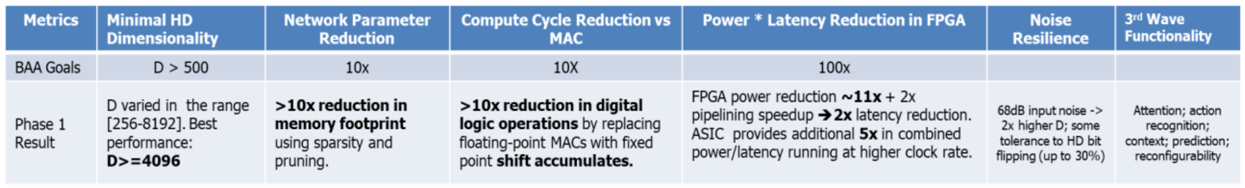
A key innovation in our proposed architecture lies in increasing the richness of operations performed with the HD representation. Previous approaches have used HD computations in a limited capacity, mainly as a way of performing a nearest-neighbor search largely defined by the non-HD input features.



LRCN DNN
Non-MAC Sparse Encoder
(2k size)
T-timesteps/ Frame
Non-MAC Recurrent LSTM Layer
Frame Embedding 1x2048x8bits
Videos 25FPS
UCF 101 Test Video Data Set
12-timestep Feature embeddings 12x512x8bits
Generate exemplars per class during training
HD Encoder
HD Decoder
Bundle Majority
position, value
Feature Embedding of size N=512
XOR
Majority Voting
Bundle
Bind
Hamming Distance
4096 bits
101 Class exemplars Associative Memory
12-timestamp Average
Min Distance
Class #
Trained 4-bit RESNET50 Model
1. 4-bit kernel parameters
2. 8-bit batch normalization parameters
Trained 4-bit LSTM model
4-bit current and recurrent kernel parameters of standard LSTM
Trained 4096HD Model Files:
1. 256 Feature Value HD LUT
2. 512 Feature Position HD LUT
3. 101 Class exemplar HD LUT
tSNE plot of HD encoding vectors

Fig. 3. Block diagram of HyDRATE from data input to activity recognition. 12 timestamp feature embeddings are generated from the quantized non-MAC feature extractor and are converted into HD vectors. They are compared with the class exemplars stored in the associative memory. The average output determines the activity class of the time. The tSNE plot shows clusters of each class in HD space from a 12-class subset. Note: LRCN stands for Long-term Recurrent Convolutional Network, implemented in this architecture as ResNet50 + LSTM.

Table 2. Summary of HyDRATE key performance metrics



Metrics	Minimal HD Dimensionality	Network Parameter Reduction	Compute Cycle Reduction vs MAC	Power * Latency Reduction in FPGA	Noise Resilience	3rd Wave Functionality
BAA Goals	D > 500	10x	10X	100x		
Phase 1 Result	D varied in the range [256-8192]. Best performance: **D>=4096**	**>10x reduction in memory footprint** using sparsity and pruning.	**>10x reduction in digital logic operations** by replacing floating-point MACs with fixed point **shift accumulates.**	FPGA power reduction **~11x** + 2x pipelining speedup ➔ **2x** latency reduction. ASIC provides additional **5x** in combined power/latency running at higher clock rate.	68dB input noise -> 2x higher D; some tolerance to HD bit flipping (up to 30%)	Attention; action recognition; context; prediction; reconfigurability

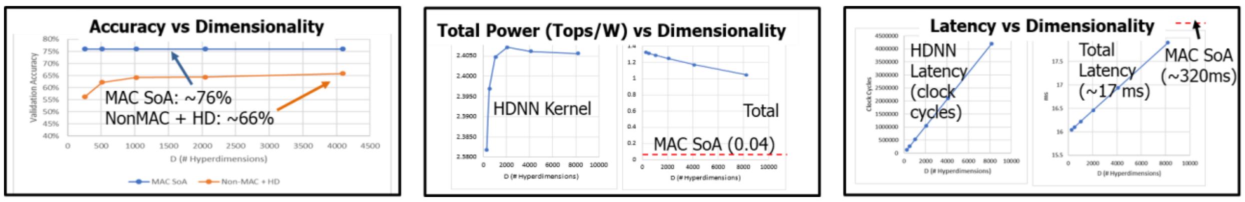


Fig. 4. Key performance results of HyDRATE architecture showing reduced power and latency with similar accuracy performance compared to a state-of-the-art MAC-based deep neural net (MAC SoA = ResNet50+LSTM)

V. FPGA Realization

In this section we describe the FPGA prototype implementation of the architecture to achieve 10x smaller memory footprint, 10x simpler operations and 100x lower latency/power compared to traditional deep neural networks (Table 2). The FPGA implementation uses only logic blocks and no DSP units, since the DSP units represent highly optimized multiply-accumulate (MAC) modules. This provides a more accurate representation of a targeted ASIC implementation. We use embedded memory where possible to emulate an optimized ASIC implementation that can be optimized using "process in memory" (PIM) type architectures.

The diagram in Fig. 5 shows the three main functions: ResNet50, LSTM, and HD; as well the pre-processor and post-processor as parallel modules with interfaces to the AMBA system bus. The AMBA system bus provides DMA channels to the external memory and data and control access to the modules by the embedded ARM processors. The Pre-processor takes video or test data from the video port or from an SD card and reformats the data for the ResNet50 module. The post-processor provides data and imagery to the display port and SD card for demonstration and testing.

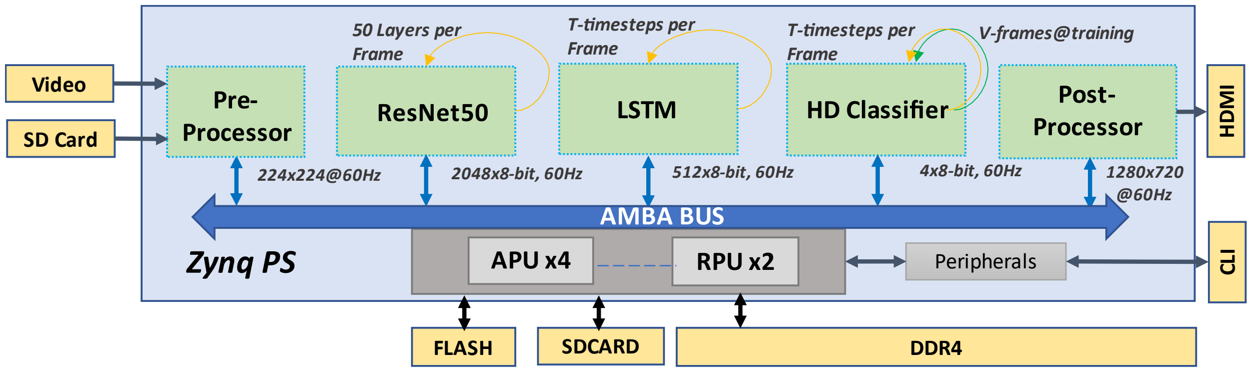


Fig. 5. Implementation in FPGA

Both ResNet50 and LSTM functions are implemented using a configurable Neural Network Processing Engine (NNPE) that includes S configurable vector modules based on shift-and-accumulate (SACC) architecture achieving 10x simpler operations than multiply-accumulate (MAC) functions. Fig. 6 shows the diagram of the NNPE which also includes a configurable data input buffer, a parameter buffer for each SACC module, and a configurable data output buffer.

Each vector SACC module has N parallel sign, shift, and add functions, and an accumulator to perform M×N vector operations. The input buffer and parameter buffer are organized to provide M×N wide data representing $k \times k$ convolutions for Cin channels, where $M \times N \geq k \times k \times Cin$. All SACC modules operate on the same input data, generating S output channels. This operation is repeated Cout/S times, where the data is stored in the output buffer with data organized to provide the data for the next layer after swapping the input and the output buffer. For most of the layer operations, the data can remain in the local data buffers. Only the first input data and the final output data, and some of the intermediate data are read and/or written to the external memory, saving significant amount of power. These data buffers with the SACC modules could be implemented as a PIM (Processor in Memory) architecture in a future ASIC.

The vector SACC module can be programmed/configured for any of the ResNet50 layers, or for many of the common CNN type layers. The output functions contain options to compute a Batch Normalization function, scaling of the data, a RELU operation, and non-linear operators such as sigmoid and tanh functions that are required by LSTM-like functions. For the FPGA implementation, the NNPE is separately configured for ResNet50 and a parallel operating LSTM function. This could also be implemented as a software configurable module in an ASIC for a wide set of applications.

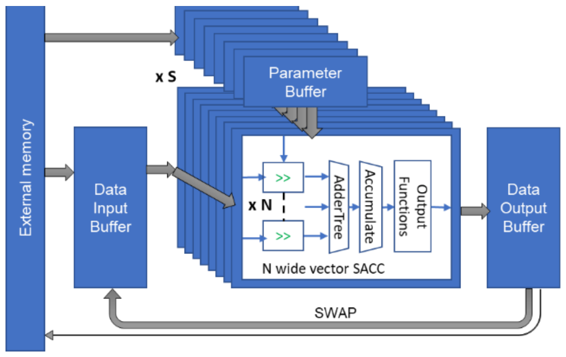


Fig. 6. NNPE architecture used in ResNet50 and LSTM modules

For every frame input, the ResNet50 module generates 2048 feature embeddings output. For every frame processed by ResNet50, the LSTM module uses the 2048 feature embeddings and executes a sliding window of T-timesteps (T=12 in Fig. 3) and generates 512 feature embeddings with position and value.

The HD Classifier module comprises an HD encoder, an HD Decoder and a new HD Exemplar block as shown in Fig. 3. Trained position encoding HD vectors, value encoding HD vectors and UCF101 class exemplars are loaded in the LUT memories in HD Classifier. A future ASIC implementation would be able to take advantage of an associative memory for the decoder. The HD Classifier module is parametrized based on HD vector length.

Using the 512 feature embeddings generated by LSTM every timestep, the HD Encoder generates a frame HD vector. Every timestep, minimum Hamming distance between the frame vector and all trained class exemplars are computed in the HD decoder. At the last timestep (end of the LSTM timestep sliding window), the Hamming distances of all timesteps per class are averaged. Resulting class with minimum averaged Hamming distance is the best matched class.

During in-situ training of a new class, using the feature embeddings from LSTM for T-timesteps/frame and V-input frames, frame HD vectors generated by HD encoder. Bit-wise bind/bundle operations of frame vectors are performed over T-timesteps/frame for V-frames. At the end of the training, bit-wise majority is performed to create a new class exemplar, that is appended to the class exemplar LUT memory. The HD Decoder is not active during in-situ training.

Power, complexity, memory, and bandwidth are significantly reduced using SACC, with 4-bit coefficients and 8-bit data input compared to a 32-bit float MAC implementation on the FPGA, and even more so in an ASIC implementation. In our FPGA test, a 32-bit float MAC required 7.9x more power, executes 4x slower, and uses specialized DSP modules in the FPGA. Reducing the data and weights bit widths saves ~6x memory bandwidth and enables us to save all intermediate data in FPGA memory instead of reading/writing from/to DRAM. We estimated that the ResNet50 implementation for the same configuration is 5.8x higher in power and 4x slower in rate and 4x longer in latency.

In our configuration for the larger FPGA, we use an 8x 256-vector SACC processing at 15 msec per 224x224 frame, with a clock rate of 200 MHz The LSTM is configured as a 128 vector SACC followed by a 32 vector SACC and requires 11.8 msec for 12 timesteps/ResNet50 frame, of which 11 can operate in parallel with processing of one frame by the ResNet50. The HD module is implemented as a 4096-vector operating requiring 8.1msec for 12 timesteps from LSTM, of which 11 can operate in parallel with processing of one frame by the ResNet50. For this configuration we estimate the FPGA power at 5W with 16.7 msec latency. This is 24.2x better power×latency than the MAC implementation and is estimated to be > 100x better when implemented in an ASIC.

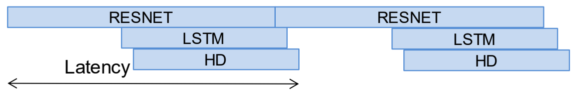


Fig. 7. Latency diagram showing the parallel operation of LSTM and HD, except for the most recent frame

In other optimizations**,** the latency can be reduced by almost 2x by doubling the parallel NNPE for the ResNet50. The HD encoder and HD reconfiguration computations could be shared since they perform similar arithmetic operations and do not run in parallel. This would reduce the size of the HD classifier. Since the rest of the processing (ResNet50 and LSTM) take relatively longer time, the HD encoder's bit-wise computation can be performed in sequence. This would increase the HD processing latency but would not affect the overall system latency significantly.

VI. Reconfigurability and Real-Time Demo

Software Framework is shown in Fig. 8. The two real-time processors (RPU0 and RPU1) that run Xilinx FreeRTOS are enabled in the Zynq Ultrascale+ MPSOC to control the accelerators, manage memory, and execute the application. The framework has a hardware abstraction layer (HAL) that interacts with the Operating System. HAL comprises device drivers that handle interrupts, threads, semaphores, and converts the API calls into register accesses to hardware devices, etc. A multi-threaded service flow architecture is used to manage and control the hardware devices in parallel.

Software framework was implemented in a mix of C++ and python. A MicroPython script is used as a command line interface (CLI) to run the tests. Python scripts are used to define tests and configure the hardware system.

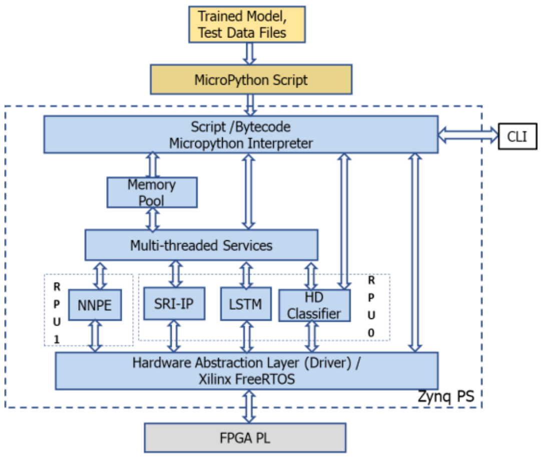

Trained Model, Test Data Files
MicroPython Script
Script /Bytecode Micropython Interpreter
CLI
Memory Pool
Multi-threaded Services
RPU1
NNPE
SRI-IP
LSTM
HD Classifier
RPU0
Hardware Abstraction Layer (Driver) / Xilinx FreeRTOS
Zynq PS
FPGA PL

Fig. 8. Software Architecture with RPU control

Real-time demonstration plan: We have been implementing the application on two different size Xilinx FPGAs. Fig. 9Fig. 9 shows a XU15EG and the larger XU19EG modules we used with different configurations and performance.

Fig. 9. FPGA Demonstration platform

Retraining baseline LRCN network model (originally trained with UCF101 dataset) with videos of activities captured by the demo camera is done to perform the real-time demonstration of video activity classification using the FPGA demonstration platform.

The trained LRCN model and HD Classifier LUTs, MicroPython scripts, HyDRATE software application and the FPGA bit file are saved in the SD Card as shown in Fig. 5. The FPGA platform is interfaced to a color camera, display and serial CLI. The CLI is used by user to initiate in-situ training, run scripts and get system performance information.

Inference of video activity classification is performed by focusing the center of the camera at the activity. Sequence of images captured by the camera are pre-processed to take the 224x224 center ROI of the image. 224x224 images are then processed by the LRCN network and HD classifier, as explained in Section V, and the classified activity result is displayed by the Post-processor on the HDMI display. A high-level inference data flow is shown in Fig. 10.

In-situ training of a new activity data flow is shown in Fig. 10. The user can initiate real-time training of a new activity with a new class label via the CLI interface. Generation of new class exemplar using the incoming frame embeddings from LRCN network is explained in HD Classifier Module of Section V.

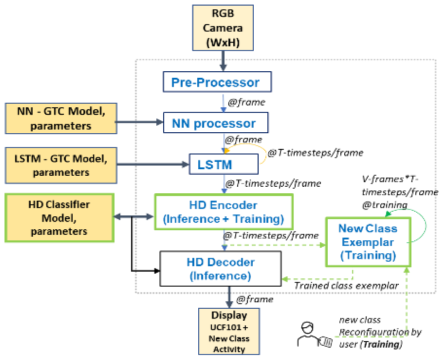

RGB Camera (WxH)
Pre-Processor
@frame
NN - GTC Model, parameters
NN processor
@frame
@T-timesteps/frame
LSTM - GTC Model, parameters
LSTM
@T-timesteps/frame
V-frames*T-timesteps/frame @training
HD Classifier Model, parameters
HD Encoder (Inference + Training)
New Class Exemplar (Training)
@T-timesteps/frame
HD Decoder (Inference)
Trained class exemplar
@frame
Display UCF101 + New Class Activity
new class Reconfiguration by user (Training)

Fig. 10. Real-time Training and Inference Flow

Accuracy, end-to-end latency and power estimate from simulation and hardware emulation are summarized in Table 2 and Fig. 4. Our estimate shows >10x reduction in memory footprint and >10x reduction in non-MAC operations compared to MAC operations in FPGA in addition to 2x latency reduction. In ASIC implementation of the same circuits, >100x of power×latency improvement is expected considering higher clock speed.

In software simulation, performance (power, latency, accuracy) variance in terms of HD dimensionality, signal noise, quantized bit and levels are studied and analyzed. For example, HyDRATE can tolerate 68dB of noise increase by doubling the HD dimensions. It can also tolerate 30% of bit flips from HD encoding, which shows the excellent resiliency of distributed signals in HD computing.

Final metrics will be measured on the real-time demonstration platform. To measure **accuracy** of classification, the application will run on a video set with known activity as ground truth. It will record the classes on the target platform for that video set and will be compared against ground truth to report accuracy. **Latency** measurement will be indicated using camera-to-display timing on the HDMI display. **System power** will be measured using a COTS DC power meter connected to the power supply while the application is actively running on the board. Static (idle) power required to power the hardware platform would be excluded from the measurement.

VII. SUMMARY

SRI's HyDRATE architecture is a non-MAC feature extractor combined with a HD classifier. HyDRATE has been optimized for video classification, and its performance is on par with MAC-based DNNs such as ResNet50 + LSTM. The authors believe this is the first non-MAC HD architecture to perform 3D (video) analytics at video rates. Its non-MAC, logic-based FPGA implementation addresses the low-power needs of edge-friendly deployment while also providing edge-friendly retraining of the HD decoder.

VIII. ACKNOWLEDGMENT

We offer special thanks to Dr. Young-Kai Chen, program manager for DARPA's Artificial Intelligence Exploration (AIE) HyDDENN program, for giving SRI the opportunity to explore and present new concepts in hyperdimensional computing.

IX. REFERENCES

[1] M. Imani et al., "VoiceHD: Hyperdimensional Computing for Efficient Speech Recognition", ICRC, 2017.

[2] Parajuli, S., Raghavan, A. and Chai, S. "Generalized Ternary Connect: End-to-End Learning and Compression of Multiplication-Free Deep Neural Networks." arXiv preprint arXiv:1811.04985 (2018).