Efficient Neural Architectures, Workload Mappings, and Hardware Layouts for Hyperspectral Imaging

Madeline Loui Anderson, Hyewon Jeong, Joanna Kondylis, Ferdi Kossmann

***Abstract*—Hyperspectral images (HSI) are becoming increasingly important in various sectors, including remote sensing, medical imaging, and material research. These images contain tens to hundreds of channels, making them high-dimensional and challenging to analyze. In this project, we investigate the energy-accuracy tradeoff of neural network architectures designed for HSI segmentation using 3D convolutions and analyze the effect of architectural configuration on MAC, the number of cycles, and parameters. We compare the results to the neural network using 2D convolutions and conclude that 3D convolution has a better energy-accuracy trade-off, achieving higher accuracy while using less energy. We then investigate hardware designs based on 1D and 2D PE (1×16, 2×8, 4×4) for performing 3D convolutions with variable filter sizes (3, 5, 7, 9). Our findings have important implications for the development of an efficient neural network for HSI segmentation in various applications, providing an understanding of the efficiency of 3D convolutions compared to 2D convolutions and which hardware design parameters allow for the efficient execution of 3D convolutions. Our findings have significant implications for developing more effective HSI segmentation in a variety of applications.**

I. Introduction

Hyperspectral images (HSI) capture information across the entire electromagnetic spectrum, resulting in 10s to 100s of spectral channels as opposed to 3 channels for RGB images. By capturing richer spectral information, HSIs can allow for more accurate predictions in tasks such as classification, object detection, and image segmentation. This has driven recent work in developing deep learning models for HSI analysis [1], [4], [12] and has led to wide adoption of HSIs in commercial, industrial, and military applications [7], including land-cover detection, agricultural development, environmental protection and urban planning.

HSI analysis comes with its own set of challenges. For HSI data, the number of channels is proportionally larger in comparison to the other spatial dimensions, which is not the case for ordinary RGB images. Therefore traditional analysis methods designed for RGB images may not be optimal for HSIs. An ideal hyperspectral classifier leverages both the spectral and spatial information in the images. Thus we investigate the use of 3D-convolution kernels for HSI segmentation and explore whether it is more suitable for the cubical HSI data structure as opposed to 2D-convolutions.

In order to gain a better understanding on how 3D convolutions can be used to build efficient HSI models, we evaluate them on three aspects: (1) we study the accuracy-energy tradeoff of different 2D and 3D convolutional neural network (CNN) architectures based on a state-of-the-art 3D CNN for HSI segmentation [3], (2) we explore the effect of using weight-stationary and out-stationary dataflows with varying filter sizes for a 3D CNN, and (3) we study how the energy consumption of 3D convolutions changes for different hardware architectures by adjusting the 2-dimensional PE layout. This work aims to extract insight on how to design and use 3D CNNs for HSI analysis in a hardware-constrained setting.

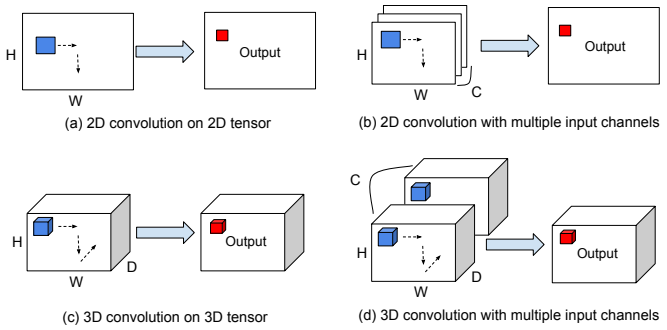

Fig. 1. The 2D and 3D convolution operations.

II. Background

3D convolutions apply a three-dimensional filter to a multi-channel image. The filter moves in three directions (x, y, z) to calculate the low level feature representations. The output shape is a 3 dimensional volume such as cube. 3D convolutions are often used for event detection in videos or 3D medical images (e.g. Magnetic Resonance Imaging and Computerized Tomography scans) [14]. 3D kernels are used to extract spatial features and spectral features from HSI data simultaneously [16]. Compared to the 2D kernel, the 3D kernel includes the depth as an additional dimension, allowing the kernel to slide over the spatial x, y, and spectral channel dimensions of the image. Figure 1 shows 3D convolution as compared to 2D convolution applied to 2D, 3D, and multi-channel images.

3D Convolution represented in Einsum notation $O_{n,m,p,q,f} = I_{n,c,up+r,uq+s,uf+t} \times F_{m,c,r,s,t}$, where n is the number of batch size for input/output feature maps (fmaps); c is the number of channels in the input fmaps and filters; r, s, t are the height, width, depth of the filter (weights), respectively; m is the number of channels in the output fmaps; p, q, f are the height, width, depth of the output fmap respectively; u is the stride of the convolution.

III. Related work

Several prior works have noted the importance of computational efficiency in HSI applications. The authors of [6] conduct an experimental analysis of performance, cost, latency,

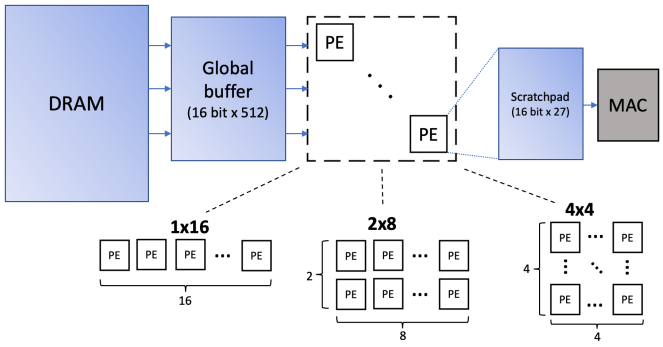

Fig. 2. Template for the architectures used in the experiments.

and energy consumption for different layers of a 3D CNN using the NVIDIA Jetson Tegra TX2. The authors of [11] investigate the energy consumption of further models on the ODROID-XU4, a heterogeneous computing device [11]. [8] proposes a hardware/software implementation for compressive sensing on a system-on-chip (SoC) field-programmable gate array (FPGA) and is shown to operate at competitive speeds with greatly reduced energy consumption. While prior works provide many meaningful insights, they are usually limited to a specific hardware architecture and may not be generalizable. Because of the recent growth of deep learning and the need for energy-efficient processing for onboard/remote sensing applications, our work explores the computational performance trade-offs of deep neural networks for HSI segmentation, providing more general insights for the optimal design of CNN models in the context of different hardware constraints. In particular, we provide an analysis on the energy-accuracy tradeoff for different 2D and 3D CNN workloads, effect of using different dataflows with varying filter size, and the energy efficiency of 3D convolutions using different PE layouts for HSI segmentation.

IV. Experimental Design

Efficiently running a Hyperspectral imaging workload involves the optimization of many moving parts. We aim to provide a reference for such tuning decision by evaluating the effect of tuning an HSI application at three levels: The neural network architecture, the mapping of the workload onto the hardware, and the hardware PE layout.

A. Experimental Setup

Dataset. We use the AVIRIS Indian Pines Data [2], which was gathered by the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) sensor over the Indian Pines test site in North-western Indiana in June 1992. The data has a spatial dimension of 145x145 with 220 spectral channels in the wavelength range 0.4-2.5 μm. It has a spectral resolution of 10 nm and a spatial resolution of 20 m by pixel. The ground truth contains 16 land cover classes with a total of 7,707 samples.

Model We consider CNNs for image segmentation on hyperspectral data. We use the 3D CNN from [5], which is one of the state-of-the-art 3D CNNs for HSI segmentation, as a basis of our workloads. The workload consists of six 3D convolutional layers and one fully connected layer. We also create a 2D CNN for segmentation for comparison with six 2D convolutional layers.

Hardware Architectures. All experiments use the same memory hierarchy and storage components. We use an off-chip DRAM buffer that is large enough to hold all tensors. We further use an on-chip global buffer of 16 bits $\times$ 512 which is shared among all PEs and a scratchpad of 16 bits $\times$ 27 which is contained on each PE. Figure 2 schematically depicts this hierarchy. To realistically model real-world hardware technologies, we model our hardware with the specifications provided in lab 3 [13] (i.e. same DRAM, same SRAM, same smart storage, same MAC).

The chip includes a PE array of 16 PEs, which may be arranged in the following layouts: 1×16, 2×8, 4×4. In our evaluation of the neural architecture and mappings, we assume a PE layout of 1×16. In our evaluation of the PE layouts, we compare the three different PE layouts.

Implementation Our experimental setup involves simulating all experiments using Timeloop [9] and Accelergy [15]. The PyTorch [10] framework is used for training the models and obtaining their accuracy, with our implementation based on the repository accompanying a 3D deep architecture [5] from the PyTorch implementation [1]. We further create helper functions that generate YAML files for Timeloop/Accelergy from a given PyTorch model and to evaluate different mappings and PE layouts.

B. Evaluation of Neural Network Architectures

We aim to investigate the resource-accuracy trade-off of different neural network architectures. We compare 2D CNNs against 3D CNNs by evaluating a large set of neural architectures for each of the CNN types. For both classes of neural networks, we investigate the accuracy achieved by different neural architectures compared to their energy consumption, latency, MAC count, and the number of trainable parameters. We evaluate the architectures on the Indian Pine dataset described in Subsection IV-A and on the 1×16 PE hardware architecture layout described in Figure 2. We always use an output stationary dataflow, as we find in Subsection V-B that such a dataflow leads to the lowest resource consumption for architectures that are commonly used in practice.

We split the entire Indian Pines dataset into an 80% training set and a 20% test set to ensure a proper evaluation of our model's performance and generalization capabilities. We used a batch size of 16, a learning rate of 0.01, Stochastic Gradient Descent optimizer with weight decay of 0.0005, and employ the cross-entropy loss for training the models.

C. Evaluation of Workload Mappings

The mapping of a workload to its underlying hardware can have a large impact on energy consumption, latency and utilization [9]. Compared to 2D convolutions, 3D convolutions further exacerbate two challenges of finding an optimal workload mapping. First, 3D convolutions entail a larger search space, with an untiled loop nest of 9 loops compared to only 7 loops for 2D convolutions. Most automated mapping tools

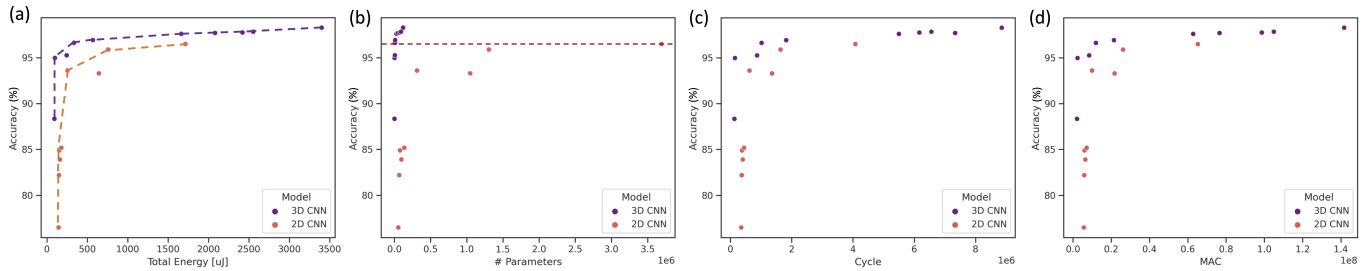

Fig. 3. Evaluation on the effect of (a) total energy, (b) parameters, (c) cycle, and (d) MAC on accuracy for 2D CNN and 3D CNN. (a) The dotted line indicates the Pareto front of 3D CNN (Purple) and 2D CNN (Orange), respectively. A red dotted line of (b) indicates the highest performance of 96.5 achieved by 2D CNN with 3.7 million parameters, which is comparable to 3D CNN achieving 96.63 with 8K parameters.

TABLE I

SUMMARY OF 3D CNN AND 2D CNN ARCHITECTURE AND ITS EFFECT IN ACCURACY, TOTAL ENERGY (μJ), MAC, THE NUMBER OF PARAMETERS, AND CYCLES. WE SUMMARIZE THE ARCHITECTURE CONFIGURATION OF EACH CONVOLUTIONAL LAYER (LAYER 1-6): (INPUT CHANNEL, OUTPUT CHANNEL, (FILTER WIDTH, FILTER HEIGHT, FILTER DEPTH)).

	Accuracy	Total Energy	MAC	Param	Cycle	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6
3D CNN	88.34%	92.83	2110912	1102	131932	1, 2, (3, 3, 3)	2, 2, (3, 1, 1)	2, 2, (3, 3, 3)	2, 2, (3, 1, 1)	2, 2, (3, 1, 1)	2, 2, (2, 1, 1)
	94.98%	98.85	2387712	2202	149232	1, 2, (3, 3, 3)	2, 2, (3, 1, 1)	2, 4, (3, 3, 3)	4, 4, (3, 1, 1)	4, 4, (3, 1, 1)	4, 4, (2, 1, 1)
	95.27%	244.5	8458800	6840	871600	1, 5, (3, 3, 3)	5, 5, (3, 1, 1)	5, 10, (3, 3, 3)	10, 10, (3, 1, 1)	10, 10, (3, 1, 1)	10, 10, (2, 1, 1)
	96.63%	334.74	11990400	8265	1018750	1, 5, (3, 3, 3)	5, 10, (3, 1, 1)	10, 10, (3, 3, 3)	10, 10, (3, 1, 1)	10, 10, (3, 1, 1)	10, 10, (2, 1, 1)
	96.93%	569.12	21399120	10425	1820500	1, 10, (3, 3, 3)	10, 10, (3, 1, 1)	10, 15, (3, 3, 3)	15, 15, (3, 1, 1)	15, 10, (3, 1, 1)	10, 10, (2, 1, 1)
	97.61%	1662.81	62802640	28375	5499790	1, 20, (3, 3, 3)	20, 20, (3, 1, 1)	20, 25, (3, 3, 3)	25, 25, (3, 1, 1)	25, 20, (3, 1, 1)	20, 20, (2, 1, 1)
	97.71%	2081.05	76564960	46120	7333940	1, 20, (3, 3, 3)	20, 20, (3, 1, 1)	20, 35, (3, 3, 3)	35, 35, (3, 1, 1)	35, 35, (3, 1, 1)	35, 35, (2, 1, 1)
	97.76%	2420.25	98673280	71140	6167080	1, 20, (3, 3, 3)	20, 20, (3, 1, 1)	20, 50, (3, 3, 3)	50, 50, (3, 1, 1)	50, 50, (3, 1, 1)	50, 50, (2, 1, 1)
	97.85%	2553.41	104952480	92340	6559530	1, 20, (3, 3, 3)	20, 20, (3, 1, 1)	20, 50, (3, 3, 3)	50, 50, (3, 1, 1)	50, 75, (3, 1, 1)	75, 75, (2, 1, 1)
	98.29%	3400.11	141696480	120840	8856030	1, 20, (3, 3, 3)	20, 20, (3, 1, 1)	20, 75, (3, 3, 3)	75, 75, (3, 1, 1)	75, 75, (3, 1, 1)	75, 75, (2, 1, 1)
2D CNN	76.48%	145.17	5666560	53860	354160	200, 20, (3, 3)	20, 20, (3, 1)	20, 35, (3, 3)	35, 35, (3, 1)	35, 35, (3, 1)	35, 35, (2, 1)
	82.19%	148.68	5920000	67000	370000	200, 20, (3, 3)	20, 20, (3, 1)	20, 50, (3, 3)	50, 50, (3, 1)	50, 50, (3, 1)	50, 50, (2, 1)
	84.89%	152.59	6086400	77400	380400	200, 20, (3, 3)	20, 20, (3, 1)	20, 50, (3, 3)	50, 50, (3, 1)	50, 75, (3, 1)	75, 75, (2, 1)
	83.90%	161.64	6470400.00	96900	404400	200, 20, (3, 3)	20, 20, (3, 1)	20, 75, (3, 3)	75, 75, (3, 1)	75, 75, (3, 1)	75, 75, (2, 1)
	85.18%	179.02	7180800.00	136800	448800	200, 20, (3, 3)	20, 20, (3, 1)	20, 100, (3, 3)	100, 100, (3, 1)	100, 100, (3, 1)	100, 100, (2, 1)
	93.61%	254.81	9977600.00	311600	623600	200, 20, (3, 3)	20, 20, (3, 1)	20, 100, (3, 3)	100, 100, (3, 1)	100, 200, (3, 1)	200, 400, (2, 1)
	93.30%	643.48	21760000.00	1048000	1360000	200, 20, (3, 3)	20, 20, (3, 1)	20, 100, (3, 3)	100, 100, (3, 1)	100, 500, (3, 1)	500, 800, (2, 1)
	95.90%	759.58	26176000.00	1306000	1636000	200, 20, (3, 3)	20, 20, (3, 1)	20, 200, (3, 3)	200, 200, (3, 1)	200, 500, (3, 1)	500, 800, (2, 1)
	96.50%	1715.26	65235200.00	3693200	4077200	200, 20, (3, 3)	20, 20, (3, 1)	20, 500, (3, 3)	500, 500, (3, 1)	500, 800, (3, 1)	800, 1000, (2, 1)

cannot handle search spaces of such sizes "out of the box" without further constraints (e.g. Timeloop throws an overflow error since the search space is larger than 2^{128}). Second, since the output is computed by sliding filters across the input, the filter size affects both output size and how often entries in input and output tensors are reused. Since 3D convolutions slide the filter across three dimensions, the impact of the filter size on data reuse and output size is larger than for 2D convolutions, that only slide across two dimensions.

To address the first issue, we explore different constraints that allow Timeloop to find efficient mappings. To ensure 100% utilization, we always constrain Timeloop to process different batch samples in parallel accross the PEs. We assume that users always want to make inference in batches, since each sample in the batch corresponds to classifying one pixel and the overarching task is to segment an entire image. We use this constraint because it gives the lowest pJ/Compute among many candidates and because parallelizing over the batch dimension is generalizable for many channel sizes and filter sizes.

We further compare the results when constraining Timeloop to output stationary dataflows versus weight stationary dataflows. For the output stationary mappings, we constrain Timeloop to iterate over all the output dimensions in DRAM and over the weight dimensions in the global buffer. For the weight stationary mapping, we constrain Timeloop to iterate over the weight dimensions in DRAM and the output dimensions in the global buffer.

To address the second issue, we evaluate the mappings for different filter sizes. We hereby always assume cube filters (e.g. $3 \times 3 \times 3$). We vary the filter sizes between $R, S, T = [3, 6, 9, 12]$ for the first layer of Hamida, i.e. with one input channel and 20 output channels.

D. Evaluation of PE Layouts

The arrangement of the 16 PEs affects their connectivity and hence the data forwarding between them. To find the optimal arrangement of the PEs, we evaluate the energy and latency of the following three PE layouts for different 3D convolutional layers: 1×16, 2×8, 4×4. We constrain our mapping to be output stationary since this generally enables more reuse. We also constrain the spatial dimensions to parallelize over the batch, N, and output channel dimension, M, to maximize utilization. These choices tend to produce more optimal mappings because batches can be run in parallel independently and each output channel's respective filter slides over the same inputs which allows for input activation reuse, which is applicable to every layer. Lastly, we chose to parallelize over M and N because most other dimensions for HSI segmentation tend to be smaller and require padding for full utilization. We use Timeloop [9] to explore the operational intensity for different layer sizes ($M = [1, 8, 16, 24, 32]$ which is similar to the sizes used in a 3D CNN for HSI segmentation [3]) and different filter sizes ($R, S, T = [3, 5, 7, 9]$) for the three PE layouts mentioned prior.

V. RESULTS

A. Evaluation of Neural Network Architectures

We experiment with the effect of accuracy on the change in the size of 2D and 3D CNN models (Table I). We compared the relationship between the test accuracy and total energy

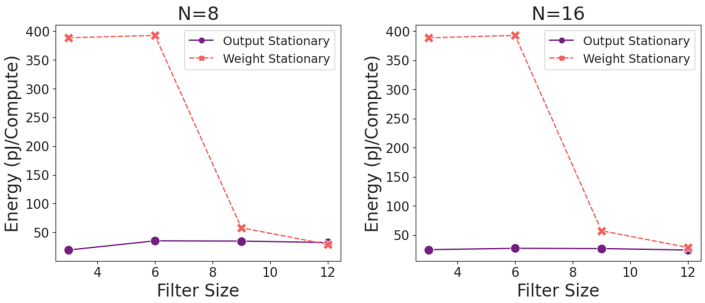

Fig. 4. Energy consumption of a weight stationary mapping compared to an output stationary mapping for different filter sizes.

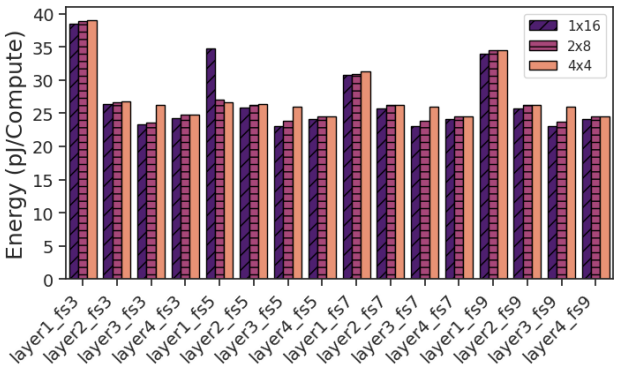

Fig. 5. Energy consumption of PE layouts compared for different layers. Input channel size to layer i is [1,8,16,24] respectively, output channel size is [8,16,24,32] respectively. Filter sizes are cube and denoted by the `fs` suffix.

consumption (μJ) in Figure 3 (a). Furthermore, the relationship between the test accuracy and size of the total parameters, MAC, and the cycle is summarized in Figure 3.

The Pareto frontal of both 2D CNN and 3D CNN is drawn as a line plot in Figure 3 (a). The 3D CNN achieved the Pareto optimal with lower energy and with higher performance, showing a better energy-accuracy trade-off for the HSI classification task.

We plot the accuracy compared to the number of parameters (Figure 3 (a)), cycle (Figure 3 (b)), and MAC (Figure 3 (c)). The results for cycles and MAC reveal a similar trend in the effect of cycle and MAC on accuracy as seen in the energy-accuracy trade-off graph (Figure 3 (a)). In Figure 3 (a), the accuracy indicated by the red dotted line was achieved in the 3D CNN model with a significantly lower number of parameters compared to the 2D CNN model. The 2D CNN model resulted in an accuracy of 96.5% with 3,693,200 parameters, a staggering 446 times the number of parameters of the 3D CNN model while achieving the same performance (96.63 with 8,265 parameters).

B. Evaluation of Workload Mappings

Figure 4 shows that for filter sizes commonly used in practice (e.g. $3 \times 3 \times 3$) the output stationary dataflow clearly dominates. Since we process a fixed $12 \times 12 \times 220$ image patch with no padding, there is an inverse relation between filter size and output size (i.e. larger filter sizes lead to smaller output sizes). For small filter sizes, the weight stationary data flow incurs large energies since the outputs do not fit into the global buffer and therefore a lot of reads and writes from/to DRAM occur. The weight stationary flow becomes more efficient for filter sizes larger than 6, where the output tensor starts to fit into the global buffer.

In certain scenarios, the weight stationary dataflow can outperform the output stationary one. This occurs when the output tensor is smaller than the weight tensor, and the output tensor fits into a lower-level buffer while the weight tensor doesn't. However, such scenarios will not occur for most practical problem settings and typically only occur when using large filter sizes. Figure 4 shows that with our default batch size of 16, the weight stationary dataflow couldn't beat the output stationary since the total size of the output tensor encompasses all of the 16 samples. However, when using a batch size of 8 instead (with 8 PEs), the total size of the output tensor is halfed and we can see that for a filter size of 12, the weight stationary mapping performed slightly better than the output stationary one ($12 * 12 * 12 > 8 * 1 * 1 * 208$).

The latencies are identical for the two mappings.

C. Evaluation of PE Layout

In the following subsection, we evaluate different PE layouts when varying filter sizes and the number of output channels of a 3D convolutional layer. As evidenced by Figure 5, the choice of PE layout does not significantly impact the operational efficiency in our setting, however the 1×16 layout tends to slightly outperform the others. The optimal mappings parallelize over the batch and output channels and are typically able to store weights in the scratchpad and global buffer to reduce DRAM accesses. Because of this, the energy usage per compute usually falls below 30 pJ/compute, which indicates that our output stationary mapping constraints are generalizable and optimal across different PE shapes, layer shapes, and filter sizes. We also found that the latency is the same across different PE layouts because the mapping is constrained by the number of operations. With our mapping design choices, each of the three PE layouts result in reasonable energy efficiency for 3D CNNs for HSI applications.

VI. Conclusion

In conclusion, our research has shown that using 3D convolutions for HSI segmentation can lead to a better energy-accuracy tradeoff compared to using only 2D convolutions. We have also demonstrated how we can map the 3D convolutional neural networks to simple 1D and 2D hardware architectures to perform the HSI segmentation. These findings have important implications for the development of more efficient models for HSI segmentation, as they suggest that optimizing both energy consumption and accuracy is possible through the use of 3D convolutions and hardware modifications. Future research could explore other types of neural network architectures or hardware designs to further optimize energy consumption and accuracy in hyperspectral image classification.

REFERENCES

[1] N. Audebert, B. Le Saux, and S. Lefèvre, "Deep learning for classification of hyperspectral data: A comparative review," *IEEE Geoscience and Remote Sensing Magazine*, vol. 7, no. 2, pp. 159–173, June 2019.

[2] M. F. Baumgardner, L. L. Biehl, and D. A. Landgrebe, "220 band aviris hyperspectral image data set: June 12, 1992 indian pine test site 3," Sep 2015. [Online]. Available: https://purr.purdue.edu/publications/1947/1

[3] A. Ben Hamida, A. Benoit, P. Lambert, and C. Ben Amar, "3-d deep learning approach for remote sensing image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 8, pp. 4420–4434, 2018.

[4] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, "Deep learning-based classification of hyperspectral data," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 6, pp. 2094–2107, 2014.

[5] A. B. Hamida, A. Benoit, P. Lambert, and C. B. Amar, "3-d deep learning approach for remote sensing image classification," *IEEE Transactions on geoscience and remote sensing*, vol. 56, no. 8, pp. 4420–4434, 2018.

[6] J. M. Haut, S. Bernabé, M. E. Paoletti, R. Fernandez-Beltran, A. Plaza, and J. Plaza, "Low–high-power consumption architectures for deep-learning models applied to hyperspectral image classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 16, no. 5, pp. 776–780, 2018.

[7] M. J. Khan, H. S. Khan, A. Yousaf, K. Khurshid, and A. Abbas, "Modern trends in hyperspectral image analysis: A review," *IEEE Access*, vol. 6, pp. 14 118–14 129, 2018.

[8] J. M. P. Nascimento, M. P. Véstias, and G. Martín, "Hyperspectral compressive sensing with a system-on-chip fpga," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, pp. 3701–3710, 2020.

[9] A. Parashar, P. Raina, Y. S. Shao, Y.-H. Chen, V. A. Ying, A. Mukkara, R. Venkatesan, B. Khailany, S. W. Keckler, and J. Emer, "Timeloop: A systematic approach to dnn accelerator evaluation," in *2019 IEEE international symposium on performance analysis of systems and software (ISPASS)*. IEEE, 2019, pp. 304–315.

[10] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[11] N. Pereira, J. Plaza, J. M. Haut, and A. Plaza, "On the evaluation of machine learning algorithms for hyperspectral image classification on a heterogeneous computing device," in *2021 16th Iberian Conference on Information Systems and Technologies (CISTI)*, 2021, pp. 1–6.

[12] A. Signoroni, M. Savardi, A. Baronio, and S. Benini, "Deep learning meets hyperspectral image analysis: A multidisciplinary review," *Journal of Imaging*, vol. 5, no. 5, 2019. [Online]. Available: https://www.mdpi.com/2313-433X/5/5/52

[13] V. Sze, J. Emer, J. Won, and M. Gilbert, "6.5930 hardware architecture for deep learning," Spring 2023. [Online]. Available: http://csg.csail.mit.edu/6.5930/info.html

[14] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," 2015.

[15] Y. N. Wu, J. S. Emer, and V. Sze, "Accelergy: An architecture-level energy estimation methodology for accelerator designs," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2019, pp. 1–8.

[16] Q. Yang, Y. Liu, T. Zhou, Y. Peng, and Y. Tang, "3d convolutional neural network for hyperspectral image classification using generative adversarial network," in *2020 13th International Conference on Intelligent Computation Technology and Automation (ICICTA)*, 2020, pp. 277–283.