DOI: 30.0485/ijearst.10.01.3500 https://ijearst.co.in/

ENGINEERING IN ADVANCED RESEARCH SCIENCE AND TECHNOLOGY

ISSN 2386-7876
Vol.02, Issue.02
August-2026
Pages: 1164-1180

Power and Density optimized Architecture for 3D Semantic Segmentation with Offset-Wise Weight Quantization using Radix8 booth encoding

[1]**Miriyala Ramalakshmi,** [2]**N. Veera Durga**
[1]M. Tech, Dept. of E.C.E, Pydah college of Engineering, Yanam Road, A.P
[2]Assistant Professor, Dept. of E.C.E, Pydah college of Engineering, Yanam Road, A.P

ABSTRACT: Three-dimensional (3D) semantic segmentation plays a vital role in modern applications such as autonomous driving, robotics, intelligent surveillance, and LiDAR-based scene understanding. However, the computational complexity and high energy consumption of 3D convolution operations present significant challenges for real-time deployment in resource-constrained systems. This work presents an enhanced energy-efficient 3D semantic segmentation processor based on offset-wise weight quantization, integrated with a Radix-8 Modified Booth Encoding multiplier to improve arithmetic performance and hardware efficiency. The architecture employs a neighbourhood generator and comparator array to efficiently process sparse voxel data while reducing unnecessary computations. The proposed Radix-8 Booth multiplier decreases the number of partial products during multiplication, resulting in lower power consumption, reduced hardware area, and improved processing speed. The combined optimization of quantized weights and efficient multiplication significantly enhances the overall performance of the segmentation processor. The design is implemented and evaluated using HDL-based simulation, and the results demonstrate improvements in computational efficiency, energy consumption, and throughput compared to conventional multiplier-based architectures. The proposed processor is well suited for real-time edge AI applications requiring fast and energy-efficient 3D scene understanding.

Keywords: 3D Semantic Segmentation, Energy-Efficient Architecture, Offset-Wise Weight Quantization, Hardware Accelerator, VLSI Design, Radix-8 Modified Booth Multiplier, Low-Power Computing, Voxel Processing, Deep Learning Hardware, ASIC/FPGA Implementation

INTRODUCTION: 3D semantic segmentation enables autonomous vehicles to understand and track the surroundings. The two primary sources of data for such tasks are the 2D images taken by cameras facing different directions away from the selfdriving vehicle and 3D point clouds collected by LiDAR sensors installed at the top of the vehicle. Different from ordered pixels as in image data, point clouds are sparse and unstructured. Since images are often collected alongside LiDAR data, how to leverage multiple modalities to further improve the 3D segmentation performance is an active area of research. While some 2D-3D fusion methods have been developed, the recent state-of-the-art methods for 3D segmentation are predominantly focused on 3D data. Inspired by the recent development in point cloud transformer models and vision foundation models, this project will be exploring the extraction of 3D point cloud features and 2D camera image features followed by effectively integrating 2D-3D features for outdoor scene 3D semantic segmentation. The fusion method developed to integrate the 2D and 3D features ultimately achieved an improvement on the primary mIoU metrics and convergence speed. In computer vision and graphics, segmenting 3D scenes is a dangerous and bleak issue. The purpose of 3D segmentation is to create computational algorithms that predict the fine-grained labels of objects in a 3D environment for a range of applications, such as medical image analysis, autonomous driving, industrial control, mobile, augmented and virtual reality and robotics [1,2]. The 3D segmentation can be divided into multi-sorts, instance and semantic segmentation. The goal of instance segmentation distinguishes between various instances of the same class labels in addition, semantic segmentation is to anticipate object class labels like table and chair e.g., when segmentation is able to discriminate between all individual objects, and this is referred to as instance segmentation. In contrast, semantic segmentation is used when segmentation can only discriminate between classes of objects. Fig. 1 is the explanation of the two segmentation techniques. Deep learning approaches have recently overtaken numerous study domains, due to their effectiveness in learning powerful features. Deep learning for 3D segmentation has also piqued the interest of the academic community throughout the last decade. However, many problems remain unresolved in 3D deep learning systems e.g., features from the RGB and depth channels are difficult to combine. The irregularity of point clouds makes exploiting local characteristics challenging, and transforming them to high-resolution voxels imposes a significant computing load [3,4]. This research provides a comprehensive summary of recent improvements in 3D segmentation using deep learning methods. It examines frequently used architectures, building components, convolution kernels and advantages and disadvantages in each scenario. Despite the fact that notable 3D segmentation surveys such as RGB-D semantic segmentation [5] and point clouds segmentation [6] have been published, these surveys do not cover all 3D data types and common application domains [2,3,5,6]. Furthermore, these surveys are mostly concerned with describing a broad review of deep learning using point clouds, rather than only 3D instance and semantic segmentation. Because of the significance of the two segmentation tasks, this research focuses solely on deep learning approaches, specifically for 3D instance and semantic segmentation. 3D SEMANTIC segmentation using point clouds is essential for perceiving the surrounding environment in a wide range of applications such as augmented reality and autonomous driving. Fig. 1(a) shows the voxel-based point cloud neural network (PNNs) [1], which is the most established approach for 3D semantic segmentation. This model first voxelizes irregular point clouds into a regular 3D grid and then applies sparse 3D convolutions to extract voxelwise features. Subsequently, these features are mapped back to the original points via devoxelization to produce per-point semantic labels. Due to their great capability of capturing geometric information, voxel-based PNNs are also adopted as 3D backbones in recent hybrid models [2], [3], [4]. However, conventional processors optimized for 2D convolution are not suitable for those voxel-based PNNs, highlighting the need for dedicated hardware for them. However, accelerating voxel-based PNNs faces three key challenges, as illustrated in Fig. 1(b). First, large-scale 3D point clouds demand enormous multiply-accumulate (MAC) operations. Existing PNN processors employ various design techniques such as dilated graph convolution [5] and max pooling prediction [6] to reduce computation, but these approaches are only applicable to point-based PNNs. Second, 3D layers in voxel-based PNNs rely on various types of operations beyond convolution. Submanifold sparse convolution (SBCONV) performs active voxel search (AVS)

DOI: 30.0485/ijearst.10.01.3500 https://ijearst.co.in/
to identify the active voxels within the receptive field prior to convolution. Furthermore, downsampling (DNSAMP) and upsampling (UPSAMP) require output coordinate generation in addition to AVS. Thus, it is critical to accelerate both AVS and output coordinate generation while keeping hardware overhead minimal. However, prior designs either fail to support output coordinate generation or rely on AVS schemes that are

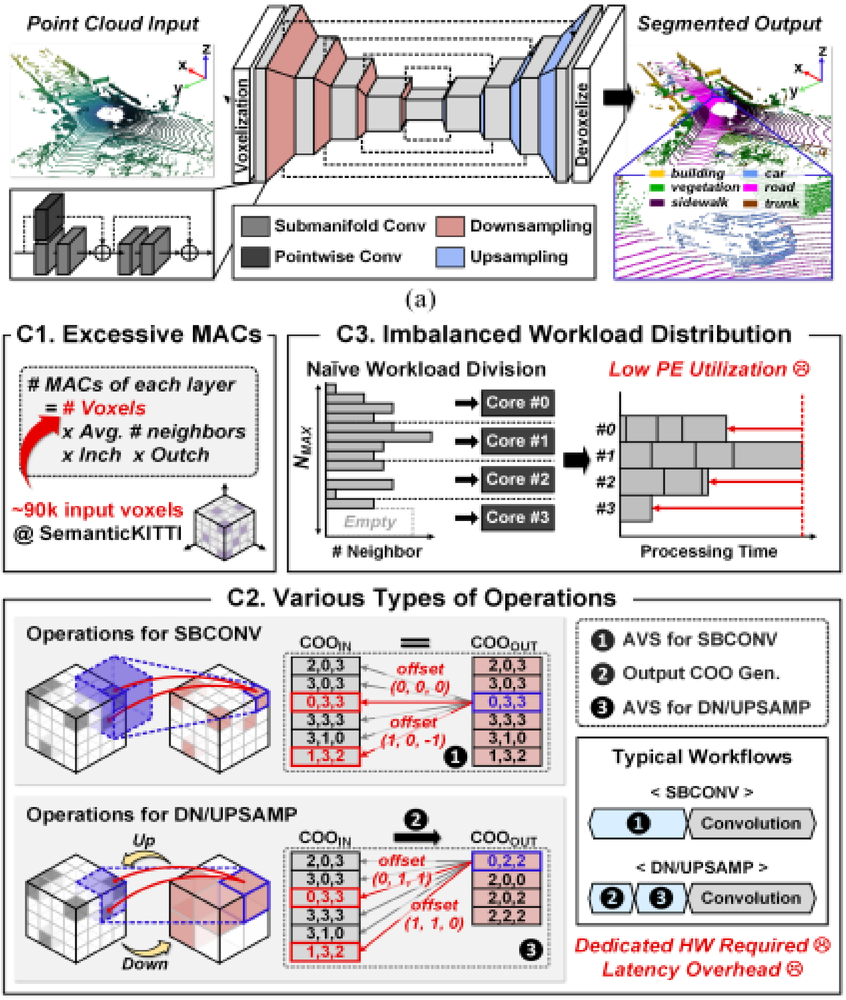


Fig. 1. (a) Overview of 3D semantic segmentation employing voxel-based PNNs and (b) challenges in processing voxel-based PNNs.

suboptimal in terms of latency and hardware overhead [7], [8], [9], [10]. Lastly, irregular voxel distributions cause unbalanced workloads across parallel computing resources, undermining hardware utilization. The hybrid scheduler in [8] could enable parallel processing of multiple blocks for layers with fewer channels, but the system performance is still limited by external memory I/O or the AVS module as they can process only one block at a time. In this work, we present a 3D semantic segmentation processor that achieves high energy efficiency by mitigating the design challenges above.

LITERATURE SURVEY: Feature representations of point clouds is key to model learning and can be grouped into three categories. Projection-based methods project 3D points to a 2D plane and perform 2D CNN, which is runtime efficient but results in 3D information loss. Voxel-based methods transform point clouds into voxel grids and apply

3D convolution or sparse convolutions, but face scalability issues as they are constrained by kernel size but need to achieve large receptive fields. Point-based methods operate directly on the points, such as PointNet [8], but the unstructured nature of point clouds impact their computational efficiency. Sparse Point-Voxel Convolution (SPVC) [10] uses both point-based and voxel-based methods but on two different streams of operations to, respectively, preserve geometrical information and enable large receptive fields, followed by fusing the information of both streams. Applying self-attention networks to 3D point cloud processing is natural as self-attention is invariant to permutation and input cardinality, while 3D point clouds are essentially sets of sparse unordered points in 3D space. Point Transformer (PT) [14] is composed of 3 types of blocks: Point Transformer Block that performs self-attention among points, Transition Down Block as an encoder, and Transition Up Block as a decoder, symmetric to the Transition Down Block as in a U-net structure. The PT decoder in the final layer produces a feature vector for each point and an MLP follows to map it to the final logits. Superpoint Transformer [9] partitions point clouds into a hierarchical superpoint structure and applies the self-attention mechanism to capture relationships among superpoints, achieving a model with comparatively fewer parameters but similar performance with other larger models. Both transformer approaches aim to partition and structure the unordered point clouds in ways to be better understood by the encoderdecoder style architecture. Along with the use of Transformer for point cloud segmentation, Vision foundation models use VisionTransformer (ViT) [4] to train on large-scale image datasets and produce high-quality visual features. One foundation model is Distillation with No Labels (DINO) [3], composed of self-supervised Vision Transformers with knowledge distillation, trained on ImageNet without labels. Given an image, DINO creates global and local views via data augmentation to encourage the model to recognize the same content across different contexts and scales. The student-teacher framework is core to DINO, where the student network processes multiple augmented views of an image and the teacher processes global views only. While both networks have the same structure, student parameters are updated by backpropagation, whereas teacher parameters are updated with an exponential moving average of student parameters. The teacher is observed to have better performance than the student throughout training, acting as a guidance to the student by providing higher quality target features for the student. Both networks produce probability distributions over the dimension of "prototype scores" and cross-entropy loss is computed to measure difference in output distributions. The combination of 3D point cloud and 2D images features to achieve 3D semantic segmentation are being explored more recently compared to single-modal data based segmentation. MSeg3D [5], which implements geometrybased and semantic-based fusion techniques, is one of the early models using both LiDAR and camera data to achieve comparable results with the state-of-the-art LiDAR only methods. BEVFusion [6] transforms camera and LiDAR features to the shared bird's-eye view (BEV) representation space and fuses the concatenated BEV features with fullyconvolutional BEV encoder. DINO in the room (DITR) [13], an approach presented in a very recent paper with code not yet published, takes advantage of 2D foundation model to extract image features, projects them to 3D, and finally injects them into a 3D point cloud segmentation model. DITR has two variants: an injection approach that uses 2D features from DINOv2 and a distillation approach that aligns 3D features with DINOv2 features during a pretraining phase, which can be used for cases when images are unavailable during inference. The 2D image features extracted by DINOv2 are reverse projected to 3D point

DOI: 30.0485/ijearst.10.01.3500 https://ijearst.co.in/
cloud and then max-pooled to different scales, such that they can be added to the skip connection between the pair of encoder-decoder blocks in a U-net style 3D point backbone. Point Transformer V3 (PTv3) [12] and DINOv2 [7] are the two backbone models used in this project. PTv3, different from its previous versions, utilizes serializations to structure points into formats that enable the attention mechanisms to better capture spatial and contextual information. Scalability and efficiency improvement is core to PTv3. To more efficiently define the spatial proximity of points, space-filling curves are used to serialize point clouds.

EXISTING METHOD:

3D semantic segmentation accelerator with an offset-wise weight quantization

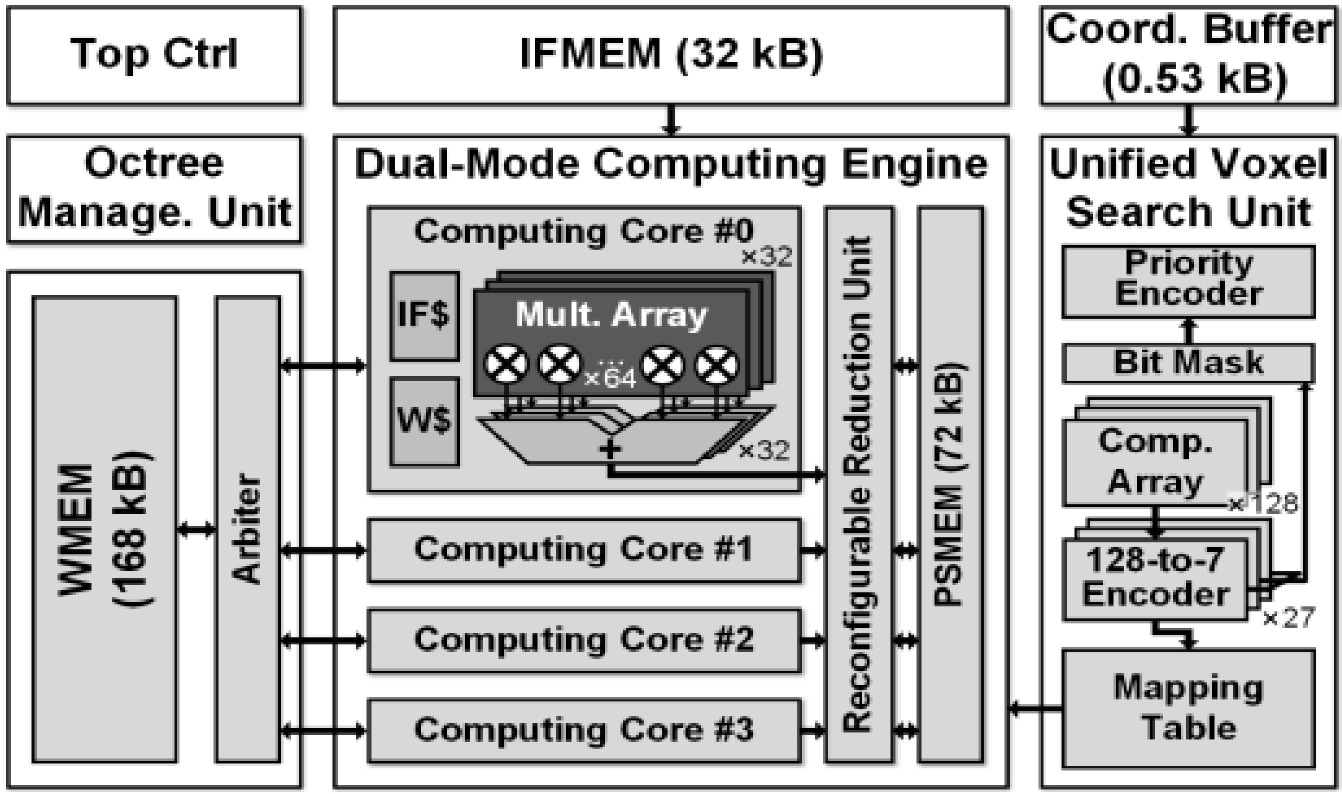


Fig. 2. Overall architecture of the proposed processor.

Overall Architecture Fig. 2 shows the overall architecture of the proposed processor for 3D semantic segmentation. It consists of a top controller, an octree management unit (OMU), DMCE, UVSU, memory banks, and a coordinate buffer. For a balanced blockwise processing, we adopt an octree structure [11] during off-chip preprocessing to group input voxels into blocks, instead of the fixed-size block-based grouping [8]. OMU manages block-level metadata, such as voxel counts and addresses. Based on this information, input feature maps (IFMAPs) and coordinates of the voxels belonging to each block (local voxels) are loaded from external memory. For SBCONV layers, the processor also loads the data of the voxels at the boundary of the neighboring blocks (foreign voxels) that may fall within a 3×3×3 kernel window centered on a local voxel, following [8]. DMCE comprises four computing cores and RRU. Each core supports bit-slice operations for mixed-precision weights exploiting 2b-8b multipliers.

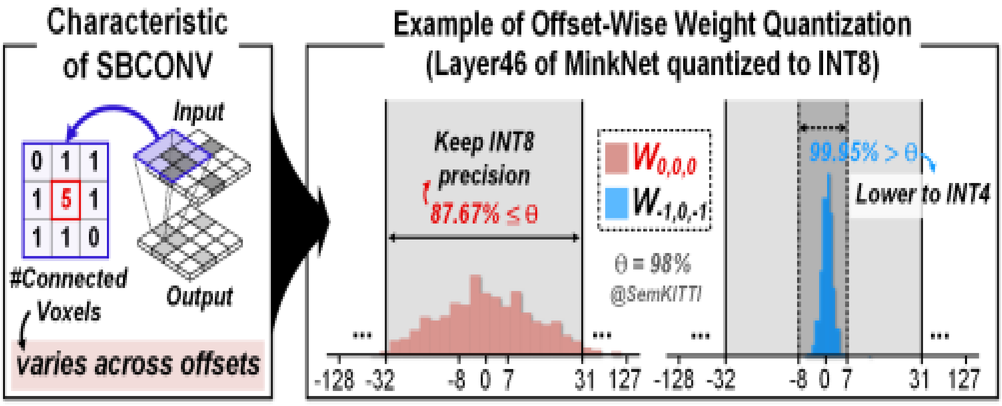


Fig. 3. Characteristics of SBCONV and the proposed offset-wise weight quantization scheme that leverages the weight distribution of each offset.

B. Offset-Wise Weight Quantization Fig. 3 illustrates the proposed offset-wise weight quantization scheme leveraging the unique characteristics of SBCONV. In SBCONV, convolution is performed only when a voxel exists at the center of the receptive field. Therefore, each offset of the cubic kernel is connected to a different number of input voxels, thus participating in computation with a varying number of output voxels. More specifically, the weights at the center of the kernel are connected to all voxels, whereas the number of voxels connected to weights at the other offsets depends on the spatial distribution of the voxels. When an offset is associated with a larger number of voxels, its weights are updated with more diverse gradients during training, which leads to a higher variance in the distribution of those weights. Therefore, the weights at the center exhibit the highest variance. To exploit this property, our offset-wise weight quantization scheme assigns different bit precisions (2/4/6/8-bit) to each offset based on the distribution of its weights, while sharing a scale factor across all offsets. The proposed scheme continues to reduce the precision in 2-bit steps until the proportion of non-overflowing weights drops below a predefined threshold. C. Unified Voxel Search Unit AVS can be viewed as a nested loop structure: (1) iterate over all local voxels (as output voxels), (2) iterate over 27 spatial offsets, and (3) iterate over all stored coordinates to determine if they are neighbors of the current output voxel. Fig. 4 displays UVSU, which parallelizes the inner two loops to search all neighbors of the current output voxel in a single cycle. To enable this, the neighborhood generators first produce a set of neighborhood coordinates using the base coordinate. Each comparator array determines whether each coordinate stored in the buffer is involved in the current convolution and, if so, identifies the corresponding offset. The comparison results are represented as a 27-bit bitmap. The bitmaps are fed into 27 encoders to extract the indices of all active voxels in parallel, and those indices are then stored in the mapping table. This approach reduces overall computation latency by 80.1% compared to the sequential search that should iterate over 27 offsets, with a 14.1% area overhead in UVSU due to the additional comparators, AND gates, and 128-to-7 encoders. While this approach still iterates over output coordinates, the latency of AVS is hidden by overlapping it with computation. Furthermore, UVSU supports efficient output coordinate generation for DNSAMP and UPSAMP layers. In conventional GPU implementations, output coordinate generation involves three sequential steps: round-down-to-nearest-even, sorting,

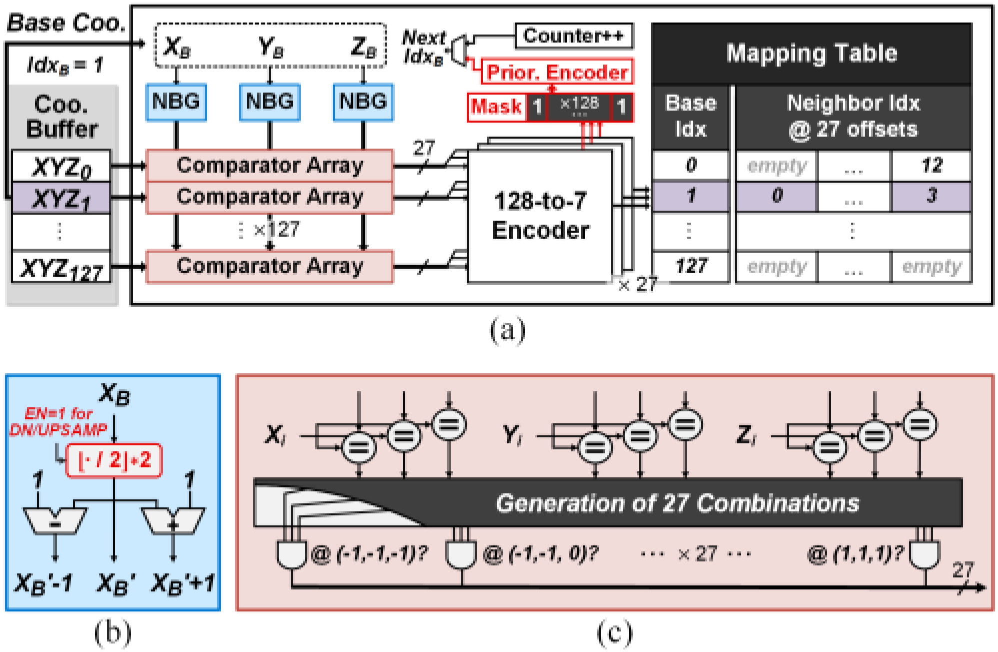


Fig. 4. Architecture of (a) unified voxel search unit, (b) neighborhood generator, and (c) comparator array. (Components highlighted in red are only enabled for DNSAMP and UPSAMP layers.)

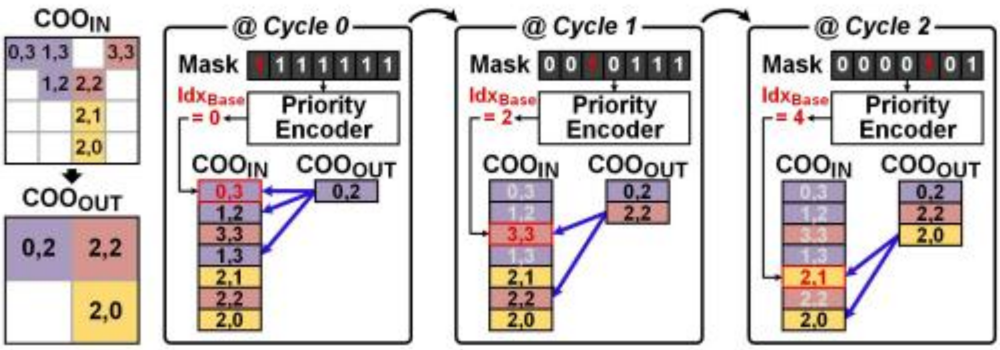


Fig. 5. Integrated output coordinate generation in UVSU.

and deduplication. Sorting and deduplication ensure that each output coordinate is unique, as multiple input coordinates may map to the same output coordinate. However, sequentially running them degrades throughput, and implementing dedicated hardware for each step would incur hardware overheads. Contrarily, UVSU integrates output coordinate generation with the base coordinate selection of AVS, introducing only a 0.44% area overhead. First, round-down-to-nearest-even is embedded in the neighborhood generators, and the rounded coordinate becomes the output coordinate. To avoid duplicates without sorting and deduplication, UVSU employs a bit-mask and a priority encoder, as depicted in Fig. 5. After AVS for the current base coordinate is completed, UVSU masks the searched input coordinates. Then, the priority encoder selects the first non-masked coordinate as the next base, preventing duplicates. As a result, UVSU reduces computation latency, power, and area by 12.7%, 20.5%, and 23.1%, respectively. D. Interleaving-Based Workload Balancing The proposed processor utilizes serial bit-sliced computation to fully exploit the benefit of offset-wise weight quantization in SBCONV layers. However, this serialization lowers bit-slice-level parallelism, resulting in poor utilization of computing cores. To address this issue, we introduce voxel-level parallelism; four cores asynchronously compute the outputs of their assigned local voxels in parallel. For high utilization, it is necessary to balance the workload across cores, and the workload per core can be expressed as:

$$WL_{core} \propto \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{N}_i} b_{i,j} = |\mathcal{V}| \cdot \left(\frac{1}{|\mathcal{V}|} \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{N}_i} b_{i,j} \right)$$

where V is the set of the assigned local voxels, Ni denotes the set of the neighbors for local voxel i, and bi, j denotes the bit-width required for the weight offset between voxel i and its neighbor j. As in Eq. (1), the workload per core can be decomposed into two factors: the number of allocated local voxels and the average workload per local voxel. The workload per local voxel depends on the spatial distribution of voxels, which appears random and is difficult to predict. However, we found that these variations tend to average out over multiple voxels. Therefore, we focus primarily on equalizing the number of local voxels allocated to each core. Fig. 6(a) shows two baseline workload distribution strategies along with the proposed interleaving-based allocation scheme. It also depicts the routing of mapping table entries to each core, where the neighbor information of local voxels is stored. Each core can identify its workloads by accessing the connected entries. An empty entry indicates no workload and is thus skipped. Since the voxel-level parallelism is first introduced in this work, there is no prior design to directly compare with. Therefore, we constructed our own baselines representing straightforward approaches. B1 is the simplest method, which partitions voxels into four equal-sized groups. This introduces significant imbalances in local voxel counts across cores, due to the inherent difficulty of the octree in fully loading voxels into fixed-size memory, as well as randomly intermixed foreign voxels. B2 further separates local and foreign voxels, then assigns the contiguous chunk of the local voxels to each core. While this achieves balanced voxel distribution, it requires a dynamic routing of mapping table entries, leading to a high routing complexity. To overcome these limitations, we propose an interleavingbased allocation scheme depicted in Fig. 6(a). As in B2, the local voxels are first separated from foreign voxels. Then, they are distributed to the cores in an interleaved manner instead of contiguous allocation. This guarantees that the number of local voxels assigned to each core differs by at most one. In addition, the table entries are hard-wired, significantly simplifying routing. Furthermore, the proposed strategy innately achieves a more balanced distribution in terms of the average workload per voxel compared to B2. This benefit arises since the voxels close to each other in memory are likely to be physically close in the 3D space, and physically close voxels often exhibit similar workloads. Therefore, interleaving ensures that these similar workloads are distributed across cores. Fig. 6(b) shows that our strategy reduces the computation latency by 47.3% and 9.8% compared to B1 and B2, respectively, on the SemanticKITTI dataset [12].

DOI: 30.0485/ijearst.10.01.3500 https://ijearst.co.in/

E. Dual-Mode Computing Engine AVS for all voxels must be completed before the four computing cores begin processing. To avoid AVS latency, we leverage the fact that all local voxels are always convolved with the center weight in SBCONV, which does not require AVS. Thus, AVS and the computation on center voxels can proceed in parallel. In addition, since both require the same number of cycles as the number of local voxels, this parallel execution fully hides AVS latency, as illustrated in Fig. 7(a).

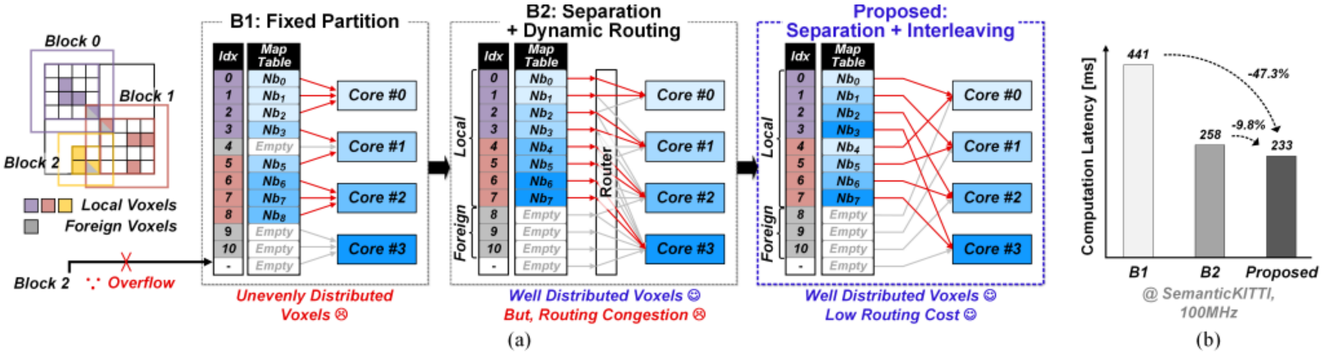


Fig. 6. (a) Overview of workload distribution schemes and (b) comparisons of their computation latency

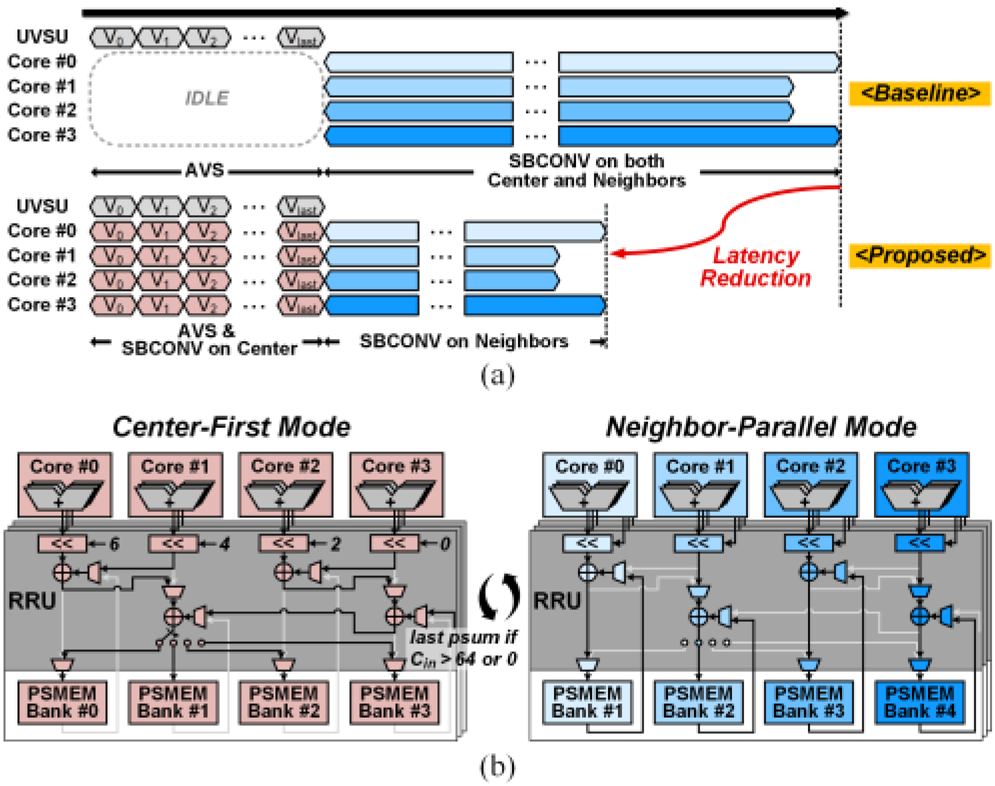


Fig. 7. (a) Overlapping between AVS and computation on center voxels and (b) two processing modes of DMCE.

To realize this, DMCE has dual-mode dataflow as shown in Fig. 7(b). In the center-first mode, all four cores collaboratively process center voxels regardless of their original core assignment. More specifically, when an IFMAP is broadcast to all four cores, each core uses its preloaded bit-sliced weights at the center to perform partial computations. Then, RRU aggregates the outputs from four cores and writes partial sums into memory. Once AVS and center voxel computations are complete, DMCE transitions to the neighbor-parallel mode. In this mode, each core independently processes the neighbor voxels associated with its assigned center voxels. In addition, RRU is reconfigured to operate as four independent accumulators. Overlapping between AVS and center voxel computations, enabled by dual-mode dataflow, results in an 18.0% reduction in overall computation latency.

PROPOSED METHOD:

3D semantic segmentation accelerator with an offset-wise weight quantization using Radix8 booth encoding multiplication

BOOTH MULTIPLIER:

Booth's Multiplication Algorithm is a Multiplication algorithm that multiplies two signed binary numbers in two's complement notation. The algorithm was invented by Andrew Donald Booth in 1950 while doing research on crystallography at Birkbeckcollege in Bloomsbury, London. Booth used desk calculators that were faster at shifting than adding and created the algorithm to increase their speed. Booth's algorithm is of interest in the study of computer architecture.

BOOTH ALGORITHM:

Booth's algorithm examines adjacent pairs of bits of the N-bits multiplier Y in signed two's complement representation, including in implicit bit below the least significant bit,y1=0.For each bit yi , for i running from 0 to N-1,the bits yi and yi-1 are considered. Where these two bits are equal, the product accumulator P is left unchanged. Where yi=0 and yi-1=1, the multiplicand times 2i is added to P; and where yi=1 and yi-1=0, the multiplicand times 2i is subtracted from P. The final value of P is the signed product. The representation of multiplicand and product are not specified; typically, these are both also in two's complement representation, like the multiplier, but any number system that support addition and subtraction will work as well. As stated here, the order of the steps is not determined. Typically, it proceeds from LSB to MSB, starting at i= 0; the multiplicand by 2i is then typically replaced by incremental shifting of the P accumulator to the right between steps; low bits can be shifted out, and subsequent additions and subractions can then be done just on the highest N bits of P[1].

RADIX-8 MODIFIED BOOTH ALGORITHM:

The Booth algorithm consists of repeatedly adding one of two predetermined values to a product P and then performing an arithmetic shift to the right on P.

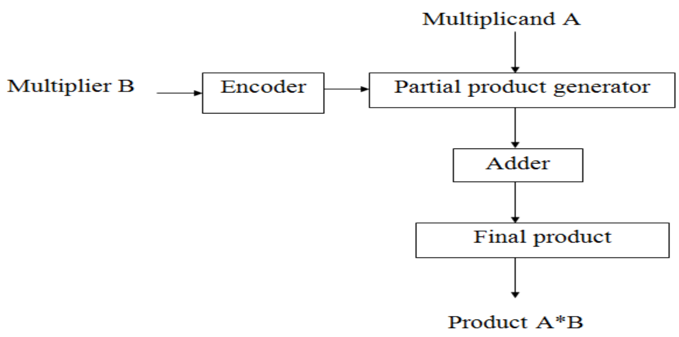


Fig. 8 Booth algorithm

The multiplier architecture consists of two architectures, i.e., Modified Booth. By the study of different multiplier architectures, we find that Modified Booth increases the speed because it reduces the partial products by half. Also, the delay in the multiplier can be reduced by using Wallace tree. The energy consumption of the Wallace Tree multiplier is also lower than the Booth and the array. The characteristics of the two multipliers can be combined to produce a high-speed and low-power multiplier.

The modified stand-alone multiplier consists of a modified recorder (MBR). MBR has two parts, i.e., Booth Encoder (BE) and Booth Selector (BS). The operation of BE is to decode the multiplier signal, and the output is used by BS to produce the partial product. Then, the partial products are added to the Wallace tree adders, similar to the carry-save-adder approach. The last transfer and sum output line are added by a carry look- ahead adder, the carry being stretched to the left by positioning.

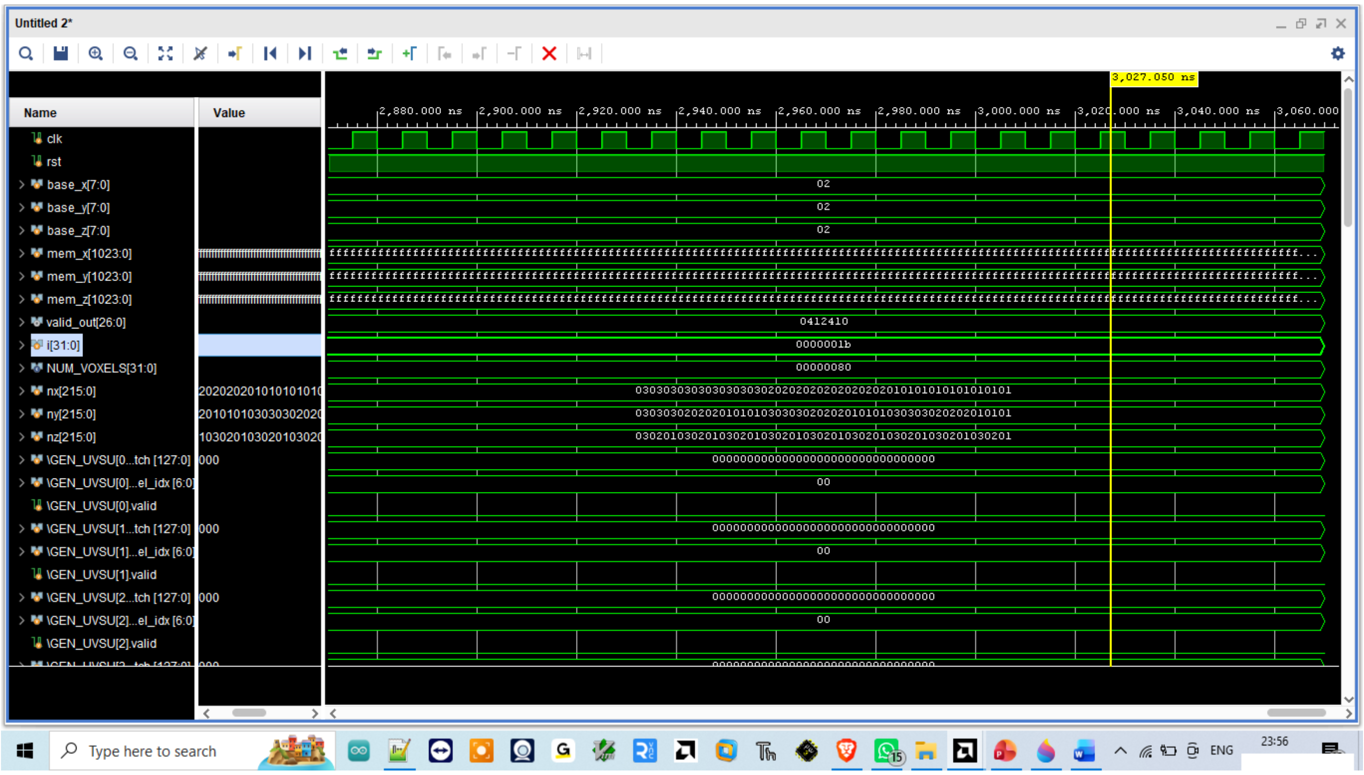
Table . a Quartet coded signed-digit table

Quartet value	Signed-digit value
0000	0
0001	+1
0010	+1
0011	+2
0100	+2
0101	+3
0110	+3
0111	+4
1000	-4
1001	-3
1010	-3
1011	-2
1100	-2
1101	-1
1110	-1
1111	0

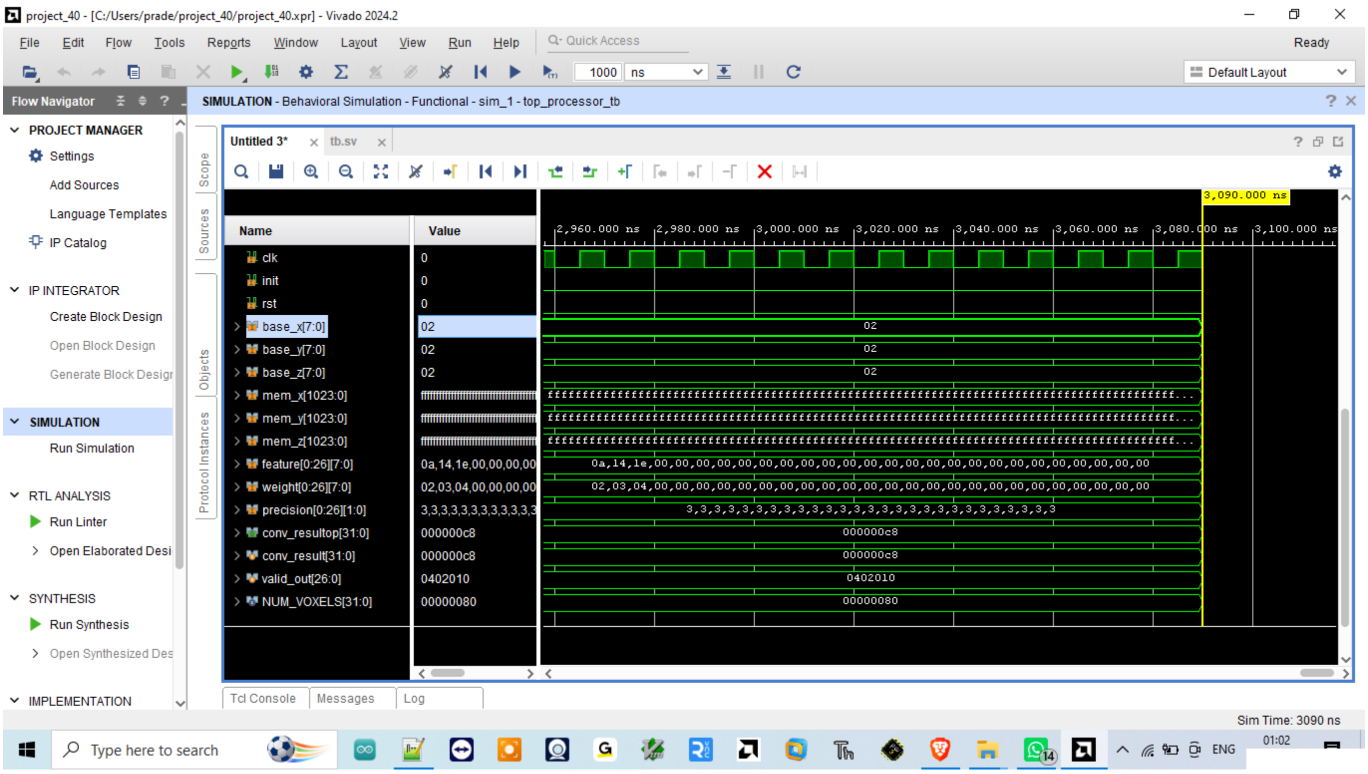
Here we have a multiplication multiplier, 3Y, which is not immediately available. To Generate it, we must run the previous addition operation: 2Y + Y = 3Y. But we are designing a multiplier for specific purposes and then the multiplier belongs to a set of previously known numbers stored in a memory chip. We have tried to take advantage of this fact, to relieve the radix-8 bottleneck, that is, 3Y generation. In this way, we try to obtain a better overall multiplication time or at least comparable to the time, we can obtain using a radix-4 architecture (with the added benefit of using fewer transistors). To generate 3Y with 21-bit words you just have to add 2Y + Y,ie add the number with the same number moved to a left position. A product formed by multiplying it with a multiplier digit when the multiplier has many digits. Partial products are calculated as intermediate steps in the calculation of larger products. The partial product generator is designed to produce the product multiplying by multiplying A by 0, 1, -1, 2, -2, -3, -4, 3, 4. Multiply by zero implies that the product is "0 ". Multiply by" 1 "means that the product remains the same as the multiplier. Multiply by "-1" means that the product is the complementary form of the number of two. Multiplying with "-2" is to move left one as this rest as per table.

Results:

EXISTING:



Proposed:



ADVANTAGES:

- **Low Power Consumption** – Offset-wise weight quantization reduces memory access and computation energy.
- **High Processing Efficiency** – Dedicated hardware accelerates 3D semantic segmentation tasks.
- **Reduced Memory Requirement** – Quantized weights occupy less storage compared to full-precision weights.
- **Fast Neighbor Search** – Parallel comparator arrays enable efficient neighborhood extraction in sparse voxel data.
- **Maintained Accuracy** – Different quantization levels for different offsets preserve segmentation performance while reducing hardware cost.
- **Real-Time Operation** – Suitable for latency-sensitive applications such as autonomous systems and robotics.

APPLICATIONS:

- Autonomous driving and intelligent transportation systems.
- Mobile robots and robotic navigation.
- LiDAR-based 3D scene understanding.
- Smart surveillance and security monitoring.
- Augmented Reality (AR) and Virtual Reality (VR).
- Industrial automation and warehouse robots.
- Drone-based environmental mapping and inspection.
- Smart city and intelligent infrastructure monitoring.

CONCLUSION: This project presents an enhanced version of the 3D semantic segmentation processor based on offset-wise weight quantization by incorporating a **Radix-8 Modified Booth Encoding multiplier** in the computation engine. The proposed modification reduces the number of partial products generated during multiplication, leading to lower computational complexity, reduced power consumption, and improved processing speed. The combination of efficient neighborhood processing, quantized weights, and optimized multiplication hardware enables faster execution of 3D semantic segmentation tasks while maintaining segmentation accuracy. The results demonstrate that the Radix-8 Booth multiplier significantly improves hardware efficiency, making the architecture suitable for real-time and energy-constrained edge AI applications such as autonomous vehicles, robotics, and intelligent sensing systems.

FUTURE SCOPE:

1. The proposed architecture can be extended with **adaptive or dynamic quantization techniques** to further reduce power and memory consumption.

2. Advanced multiplier architectures such as **approximate multipliers** or **hybrid Booth-Wallace multipliers** can be explored for additional performance improvements.
3. The design can be implemented on **advanced FPGA and ASIC technologies** to achieve higher throughput and lower energy consumption.
4. Support for larger and deeper 3D neural network models can be incorporated to improve segmentation accuracy for complex scenes.
5. Integration of **LiDAR, camera, and radar sensor fusion** can enhance environmental perception capabilities.
6. The processor can be optimized for deployment in **autonomous vehicles, drones, and mobile robots** requiring real-time 3D scene understanding.
7. Future work may investigate **on-chip learning and reconfigurable AI accelerators** to adapt to changing environments.
8. The architecture can be extended to other applications such as **medical 3D image segmentation, smart surveillance, industrial inspection, and digital twin systems**.

REFERENCES:

[1] AmirAli Abdolrashidi, Lisa Wang, Shivani Agrawal, Jonathan Malmaud, Oleg Rybakov, Chas Leichner, and Lukasz Lew. Pareto-optimal quantized resnet is mostly 4-bit. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 3091–3099, 2021. 8 [2] MohammadHossein AskariHemmat, Reyhane Askari Hemmat, Alex Hoffman, Ivan Lazarevich, Ehsan Saboori, Olivier Mastropietro, Sudhakar Sah, Yvon Savaria, and Jean-Pierre David. Qreg: On regularization effects of quantization. arXiv preprint arXiv:2206.12372, 2022. 8 [3] Ron Banner, Yury Nahshan, and Daniel Soudry. Post training 4-bit quantization of convolutional networks for rapiddeployment. Advances in Neural Information Processing Systems, 32, 2019. 3 [4] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In Proc. of European Conf. on Computer Vision (ECCV), pages 446–461. Springer, 2014. 13 [5] Rich Caruana. Multitask learning. Machine learning, 28: 41–75, 1997. 1, 2 [6] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, ˜ and Lucia Specia. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. pages 1–14, Vancouver, Canada, 2017. Association for Computational Linguistics. 13 [7] Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. 105(10):1865–1883, 2017. 13 [8] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. arXiv preprint arXiv:1805.06085, 2018. 3 [9] Soyun Choi, Youjia Zhang, and Sungeun Hong. Intra-inter modal attention blocks for rgb-d semantic segmentation. In Proc. of Int'l Conf. on Multimedia Retrieval (ICMR), pages 217–225, 2023. 2 [10] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In Proc. of Computer Vision and Pattern Recognition (CVPR), pages 3606–3613, 2014. 13 [11] Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical japanese literature. arXiv preprint arXiv:1812.01718, 2018. 13 [12] Adam Coates, Andrew Ng, and Honglak

Lee. An analysis of single-layer networks in unsupervised feature learning. In Proc. of Artificial Intelligence and Statistics (AISTATS), pages 215–223. JMLR Workshop and Conference Proceedings, 2011. 13 [13] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In Proc. of International Joint Conference on Neural Networks, pages 2921–2926. IEEE, 2017. 13 [14] MohammadReza Davari and Eugene Belilovsky. Model breadcrumbs: Scaling multi-task model merging with sparse masks. In Proc. of European Conf. on Computer Vision (ECCV), pages 270–287. Springer, 2024. 6, 14, 18 [15] Bill Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In Proc. of Int'l Workshop on Paraphrasing (IWP), 2005. 13 [16] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. 2019. 3 [17] Jun Fang, Ali Shafiee, Hamzah Abdel-Aziz, David Thorsley, Georgios Georgiadis, and Joseph H Hassoun. Post-training piecewise linear quantization for deep neural networks. In Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16, pages 69–86. Springer, 2020. 3 [18] Alexander Finkelstein, Uri Almog, and Mark Grobman. Fighting quantization bias with bias. arXiv preprint arXiv:1906.03193, 2019. 3 [19] Antonio Andrea Gargiulo, Donato Crisostomi, Maria Sofia Bucarelli, Simone Scardapane, Fabrizio Silvestri, and Emanuele Rodola. Task singular vectors: Reducing task interference in model merging. In Proceedings of the Computer Vision and Pattern Recognition Conference, pages 18695–18705, 2025. 2, 8, 14 [20] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. Proc. of Neural Information Processing Systems (NeurIPS), 31, 2018. 7, 15 [21] Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B Dolan. The third pascal recognizing textual entailment challenge. In Proc. of ACL-PASCAL Workshop on Textual Entailment and Paraphrasing, pages 1–9, 2007. 13 [22] Ian J Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, et al. Challenges in representation learning: A report on three machine learning contests. In Proc. of Int'l Conf. on Neural Information Processing (ICONIP), pages 117–124. Springer, 2013

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proc. of Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016. 5, 13 [24] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. 12(7): 2217–2226, 2019. 13