A Survey on Five-Dimensional Convolutional Neural Networks (5D-CNN)

Shilpa Biradar[1] , Dr.Savita Patil[2], Dr.Anuradha Savadi[3]

1.Student, *Dept of Electronics and Communication Engineering, Sharnbasva University, Kalaburagi, Karnataka, India.*

2.Professor, Dept of Electronics and Communication Engineering, Sharnbasva University, Kalaburagi Karnataka, India.

3.Associate Professor, Dept of Electronics and Communication Engineering, Sharnbasva University, Kalaburagi Karnataka, India.

Abstract

The rapid growth of deep learning applications involving spatio-temporal, volumetric, and multi-modal data has driven the evolution of convolutional neural networks (CNNs) beyond traditional two-dimensional (2D) and three-dimensional (3D) models. Five-dimensional convolutional neural networks (5D-CNNs), which operate on tensors comprising time, channel, depth, height, and width dimensions, offer superior representational capability for complex data such as multi-view video, hyperspectral imaging, and medical diagnostics. However, the increased dimensionality introduces significant computational and hardware challenges, particularly for real-time and embedded systems. This paper presents a comprehensive survey of CNN dimensional evolution and introduces an efficient RTL-based hardware implementation of a 5D-CNN architecture. The proposed design leverages FSM-based control flattened memory organization, and MAC reuse to achieve scalability without exponential growth in hardware resources. Functional verification and cycle-level analysis demonstrate the feasibility of deploying 5D-CNNs in hardware-constrained environments.

Keywords - 5D CNN; High-dimensional convolution; RTL implementation; FSM-based architecture; Hardware acceleration.

I. INTRODUCTION

Convolutional Neural Networks (CNNs) have become a cornerstone of modern machine learning, particularly in image and signal processing applications. Early CNN architectures were limited to two-dimensional (2D) data, primarily addressing spatial correlations in images. Subsequently, three-dimensional (3D) CNNs extended this paradigm to include volumetric or temporal information, enabling applications such as video analysis and medical imaging.

Recent advancements in sensing and data acquisition technologies have led to the emergence of highly complex datasets that incorporate multiple dimensions simultaneously, including time, channels, depth, height, and width. To effectively model such data, five-dimensional convolutional neural networks (5D-CNNs) have been proposed. These models offer enhanced feature extraction capabilities by jointly exploiting correlations across all dimensions. Despite their potential, practical implementation of 5D-CNNs remains challenging due to their immense computational complexity and memory requirements.

Most existing research on 5D-CNNs focuses on software-based implementations using GPUs or high-performance computing platforms. Hardware realizations, particularly at the RTL level, are scarce. This paper addresses this gap by surveying existing CNN dimensional extensions and presenting a complete RTL implementation of a 5D-CNN accelerator optimized for area efficiency and scalability.

FUSION METHODS

- **CNN and its Categories**

The concept of Convolutional Neural Networks (CNNs) was discussed in "[1]". CNNs are a class of deep learning models designed to automatically learn hierarchical feature representations from structured data such as images and videos. By employing convolutional layers with shared weights and local receptive fields, CNNs effectively reduce redundancy input data while improving learning efficiency and classification accuracy. CNNs are capable of extracting meaningful spatial and contextual information, making them robust to noise, translation, and local variations in the input. Due to these properties, CNNs have become the backbone of modern computer vision systems and form the foundation for advanced architectures such as 2D, 3D, and high-dimensional CNN models.

- **Two-Dimensional Convolutional Neural Networks (2D CNNs)**

Two-dimensional CNNs were discussed in "[2]" as the foundational architecture for image-based applications. In 2D CNNs, convolution operations are performed along the height and width dimensions of an input image using learnable 2D kernels. This enables effective extraction of local spatial features such as edges, corners, textures, and object shapes. Pooling layers are commonly employed to reduce spatial resolution and computational complexity while retaining salient features. 2D CNNs are widely used in applications including object detection, face recognition, handwriting recognition, and medical image analysis. Well-known architectures such as LeNet, AlexNet, VGGNet, and ResNet are based on the 2D CNN paradigm. Despite their effectiveness for static image analysis, 2D CNNs are limited in their ability to model temporal dependencies or volumetric structures, making them less suitable for video processing and 3D data analysis.

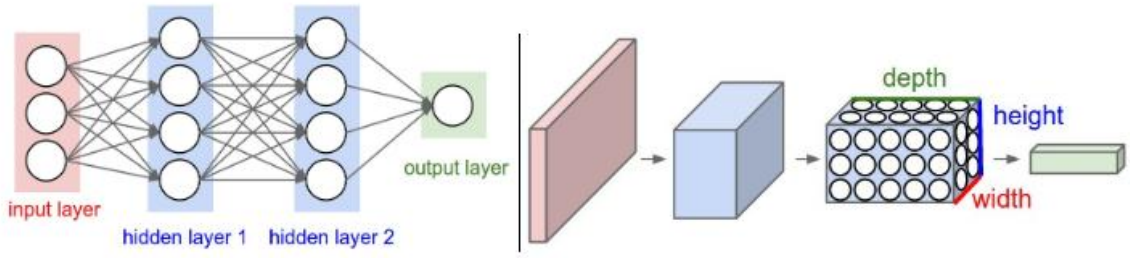

Figure 1: **Left**: A regular 3-layer Neural Network. **Right**: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels).

- **Three-Dimensional Convolutional Neural Networks (3D CNNs)**

The extension of CNNs into three dimensions was presented in "[3]". Three-dimensional CNNs incorporate an additional depth or temporal dimension into the convolution of operation, enabling kernels to operate across height, width, and time (or volume) simultaneously. This method allows the network to capture spatio-temporal features, such as motion patterns in video sequences or structural continuity in volumetric medical data. 3D CNNs are particularly effective for applications including video classification, action recognition, gesture analysis, and volumetric medical imaging such as MRI and CT scans. By learning features jointly across spatial and temporal dimensions, 3D CNNs outperform 2D CNNs in tasks involving dynamic or volumetric data. However, the expanded dimensionality significantly increases computational complexity, memory requirements, and training time, necessitating optimization techniques and hardware acceleration for practical deployment.

- **Four-Dimensional Convolutional Neural Networks (4D CNNs)**

  Four-dimensional CNNs were explored in "[4]" to process dynamic volumetric data evolving over time. These networks extend conventional 3D CNNs by introducing an explicit temporal dimension, enabling convolution operations across three spatial dimensions and one time dimension simultaneously. The core method involves applying 4D convolution kernels that capture both spatial structure and temporal continuity within volumetric sequences. By jointly modeling spatial and temporal correlations, 4D CNNs provide improved representation of motion dynamics and temporal consistency, making them suitable for applications such as dynamic medical imaging (e.g., cardiac MRI), time-resolved scientific simulations, and video-based volumetric analysis. However, the additional dimension significantly increases memory footprint, computational complexity, and data movement, necessitating optimization techniques such as kernel factorization, dimensional reduction, and hardware acceleration for practical deployment.

- **Five-Dimensional Convolutional Neural Networks (5D CNNs)**

  Five-dimensional CNNs were introduced in "[5]" to jointly process time, channel, depth, height, and width dimensions within a unified learning framework. Unlike lower-dimensional CNNs that treat certain dimensions independently, 5D CNNs apply convolution kernels across all five dimensions simultaneously, enabling the network to learn rich spatio-temporal and cross-modal correlations. This method is particularly effective for complex data such as hyperspectral video streams, multi-sensor surveillance inputs, and advanced medical imaging involving multiple modalities over time. By capturing dependencies across spatial structure, temporal evolution, and modality-specific channels, 5D CNNs achieve superior representational capability and improved accuracy compared to 2D, 3D, and 4D counterparts. However, the inclusion of additional dimensions results in exponential growth in computational complexity, memory usage, and data movement, posing significant challenges for real-time and embedded hardware implementation. To address these challenges, optimization strategies such as kernel decomposition, dimensional factorization, quantization, and hardware-aware accelerator design are essential.

- **Residual Convolutional Neural Networks (ResNet)**

  Residual learning was proposed in "[6]" to overcome the vanishing and exploding gradient problems commonly encountered in very deep CNNs. The core method introduces skip (identity) connections that allow the input of a layer to be directly added to its output. Instead of learning an explicit mapping, the network learns a residual function, which simplifies optimization and improves convergence. This architecture enables effective training of extremely deep networks such as ResNet-50, ResNet-101, and ResNet-152, achieving superior accuracy and stability compared to conventional CNNs. Residual connections also facilitate faster convergence and improved generalization, making ResNet a widely adopted backbone for image classification, object detection, and feature extraction tasks.

- **Dense Convolutional Neural Networks (DenseNet)**

  DenseNet architectures were discussed in "[7]". In DenseNet, each layer receives feature maps from all preceding layers, forming dense connectivity. This method encourages feature reuse, strengthens gradient flow, and mitigates the vanishing gradient problem. Instead of relearning redundant features, later layers build upon previously extracted representations, leading to improved parameter efficiency.

  DenseNet significantly reduces the number of parameters while maintaining or improving accuracy, making it suitable for medical imaging and fine-grained classification tasks. However, dense connections increase memory access overhead, which must be carefully managed in hardware implementations.

- **Inception-Based CNN Architectures**

Inception networks were proposed in "[8]" to capture multi-scale spatial features within a single network layer. The core method involves performing parallel convolution operations using multiple kernel sizes such as 1×1, 3×3, and 5×5, followed by concatenation of the resulting feature maps. This design enables the network to simultaneously learn both fine-grained and coarse-level features. Inception architectures efficiently balance representational power and computational cost, making them suitable for large-scale image recognition tasks. The inclusion of 1×1 convolution layers serve as a dimensionality reduction technique, significantly lowering the number of parameters and computational complexity while preserving discriminative information.

- **Depthwise Separable Convolutional Neural Networks**

Depthwise separable CNNs were introduced in "[9]" to significantly reduce computational complexity. This method decomposes standard convolution into depthwise convolution, which operates independently on each channel, followed by pointwise convolution, which combines channel-wise outputs. This approach reduces multiplications and parameters by nearly an order of magnitude compared to standard convolution. Depthwise separable CNNs form the backbone of many efficient architectures used in embedded and mobile systems.

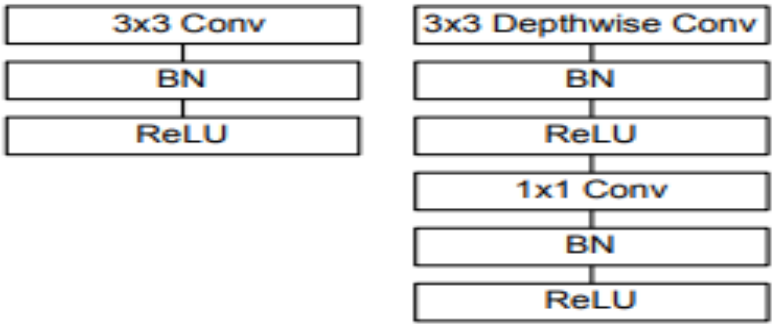

Figure 2. **Left**: Standard convolutional layer with batchnorm and ReLU. **Right**: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU.

- **Mobile-Oriented CNN Architectures**

Mobile CNN architectures were presented in "[10]" with the objective of achieving low power consumption and reduced memory usage for deployment on resource-constrained devices. These networks employ lightweight convolution operations, such as depthwise separable convolutions, along with architectural innovations including inverted residual blocks and linear bottlenecks to minimize computational overhead while preserving representational capability. Mobile-oriented CNNs are widely adopted in applications such as smartphones, drones, wearable devices, and Internet of Things (IoT) systems, where real-time inference is required under strict power and latency constraints. Although these architectures achieve high efficiency, careful design trade-offs are necessary to balance accuracy and computational complexity, particularly for more demanding visual recognition tasks.

- **Capsule Neural Networks**

Capsule networks were proposed in "[11]" to explicitly preserve spatial hierarchies and part–whole relationships that are often lost in conventional CNN architectures. Unlike scalar neurons in CNNs, capsules consist of vector-based neurons that encode both the existence (probability) and pose information (such as orientation, scale, and position) of visual features. A key component of capsule networks is the dynamic routing mechanism, which determines how outputs from lower-level capsules are selectively routed to higher-level capsules based on agreement. This routing strategy enables improved modeling of spatial relationships and enhances robustness to variations in viewpoint and

orientation. However, capsule networks involve high computational and memory complexity, which limits their scalability and practical deployment in real-time and resource-constrained systems.

- **Graph-Based Convolutional Neural Networks**

Graph Convolutional Neural Networks (Graph CNNs) were introduced in "[12]" to extend convolution operations to non-Euclidean domains, where data is represented as graphs rather than regular grids. Unlike conventional CNNs that rely on fixed spatial neighborhoods, Graph CNNs perform convolution using graph adjacency relationships, either through spectral methods based on graph Laplacians or spatial message-passing frameworks. By aggregating information from neighboring nodes, Graph CNNs effectively capture complex relational structures and dependencies present in graph-structured data. These networks have demonstrated strong performance in applications such as social network analysis, molecular and chemical structure modeling, recommendation systems, and sensor network data processing. However, irregular topology and variable neighborhood sizes introduce challenges in terms of scalability, computational efficiency, and hardware implementation.

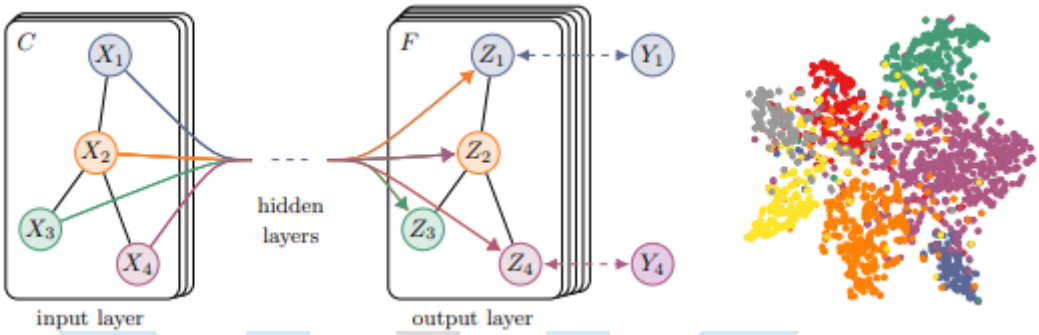

Figure 3: **Left**: Graph Convolutional Network. **Right**: Hidden layer activations

- **Recurrent Convolutional Neural Networks**

Recurrent CNNs were discussed in "[13]" as hybrid architectures that integrate convolutional layers with recurrent units such as Long Short-Term Memory (LSTM) or Gated Recurrent Units (GRU). In these models, CNN layers are responsible for extracting spatial features from individual frames or inputs, while recurrent layers model temporal dependencies and sequential correlations across time. This architectural combination enables effective learning of both spatial and temporal information, making recurrent CNNs particularly suitable for applications such as video analysis, action and gesture recognition, and time-series prediction. Although recurrent CNNs offer improved temporal modeling compared to pure CNN architectures, their sequential processing nature increases computational complexity and latency, which can be challenging for real-time and hardware-constrained implementations.

- **Attention-Based Convolutional Neural Networks**

Attention mechanisms in CNNs were presented in "[14]" to enhance feature learning by selectively emphasizing informative regions of the input. Attention modules dynamically assign adaptive weights to spatial locations or feature channels, enabling the network to focus on the most relevant features while suppressing less important information. Common attention strategies include spatial attention, channel attention, and self-attention mechanisms. By guiding the feature extraction process, attention-based CNNs improve both model interpretability and recognition performance. These architectures have demonstrated significant gains in applications such as object detection, semantic segmentation, medical image analysis, and fine-grained visual recognition. However, the inclusion of attention modules introduces additional computational overhead, which must be carefully managed for real-time and hardware-constrained systems.

- **Multi-Modal Convolutional Neural Networks**

Multi-modal CNNs were explored in "[15]" to effectively integrate information from heterogeneous sensing modalities such as RGB cameras, infrared sensors, depth sensors, radar, and LiDAR. The primary objective of multi-modal learning is to exploit complementary characteristics of different modalities, where one modality compensates for the limitations of another.

Three major fusion strategies are commonly employed. Early fusion combines raw input data from multiple modalities before feature extraction, enabling joint low-level representation learning. Intermediate or feature-level fusion merges modality-specific feature maps extracted by independent CNN branches, allowing each modality to learn specialized representations before integration. Late fusion, also known as decision-level fusion, combines classification or detection of outputs from separate networks, improving robustness under noisy or missing sensor conditions.

Multi-modal CNNs demonstrate improved performance in challenging environments such as low illumination, occlusion, and adverse weather conditions. These architectures are widely applied in surveillance, autonomous navigation, medical diagnosis, and defense systems. However, challenges remain in terms of synchronization, data alignment, and increased computational complexity, especially for real-time and embedded deployments.

- **Quantized Convolutional Neural Networks**

Quantized Convolutional Neural Networks were discussed in "[16]" to address the challenges of deploying CNNs on resource-constrained hardware platforms. The key method in quantized CNNs involves reducing the numerical precision of weights, activations, and sometimes gradients from floating-point representations to fixed-point or even binary formats. Common quantization techniques include post-training quantization, quantization-aware training, and binary or ternary quantization.

By lowering bit-widths, quantized CNNs significantly reduce memory footprint, computational complexity, and power consumption. These properties make them suitable for FPGA, ASIC, and edge-AI accelerators. However, aggressive quantization may lead to accuracy degradation, particularly for deeper and high-dimensional CNN models, requiring careful design of trade-offs.

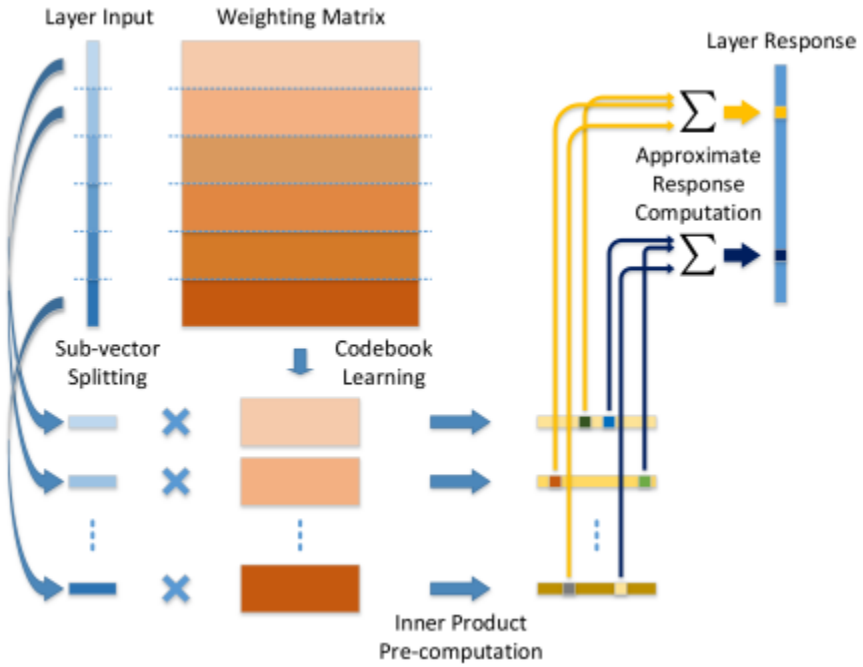

Figure 4. The parameter quantization and test-phase computation process of the fully-connected layer.

- **Sparse Convolutional Neural Networks**

  Sparse CNNs were proposed in "[17]" to exploit inherent sparsity in neural network weights and activations. The underlying method focuses on identifying and eliminating redundant or zero-valued computations during inference. Sparsity can be introduced through network pruning, L1 regularization, or structured sparsity constraints during training. Sparse CNNs reduce computation, memory bandwidth, and energy consumption while maintaining comparable accuracy. These networks are especially beneficial for large-scale CNN models deployed on hardware accelerators. However, irregular sparsity patterns can complicate hardware implementation, necessitating specialized sparse computation engines.

- **Hardware-Oriented Convolutional Neural Network Architectures**

  Hardware-oriented CNN architectures were discussed in "[18]" with an emphasis on designing CNN models that align with hardware constraints. The core methods include loop tiling, data reuse, parallel processing, and pipelining to minimize memory access and latency. These architectures often rely on finite state machines (FSMs) for control and optimized dataflow strategies such as weight-stationary, output-stationary, or row-stationary execution. Such CNN designs are crucial for real-time systems where power, area, and latency are critical. Hardware-oriented CNNs enable efficient deployment on FPGAs and ASICs, but require co-design of algorithms and hardware, increasing design complexity.

- **CNN Accelerators for Embedded Systems**

  CNN accelerators for embedded systems were presented in "[19]". These accelerators are specialized hardware blocks designed to speed up convolution, pooling, and activation operations. Common accelerator architectures employ parallel multiply-accumulate (MAC) units, on-chip memory buffering, and DMA-based data movement to achieve high throughput. Embedded CNN accelerators aim to balance performance, power consumption, and silicon area. They are widely used in applications such as autonomous vehicles, surveillance systems, and IoT devices. However, accelerator flexibility is often limited, making it challenging to support diverse CNN models without reconfiguration.

- **High-Dimensional CNNs for Real-Time Applications**

  High-dimensional CNNs, including 4D and 5D CNNs, were discussed in "[20]" for real-time and complex data processing applications. These networks process multi-dimensional inputs such as spatio-temporal, spectral, and multi-modal data. The key method involves extending convolution operations across additional dimensions while preserving correlations between them. High-dimensional CNNs achieve superior representational power and accuracy but pose significant challenges in terms of computation and memory requirements. To enable real-time performance, optimization techniques such as dimension reduction, approximate computing, and hardware acceleration are essential.

Conclusion

This paper presented a comprehensive survey of convolutional neural network methods, ranging from traditional 2D CNNs to advanced five-dimensional CNN architectures. Various CNN models and extensions were discussed, highlighting their advantages, limitations, and application domains. While high-dimensional CNNs offer superior feature representation, they introduce significant computational and hardware challenges. Therefore, the selection of an appropriate CNN architecture depends heavily on the target application and system constraints. Efficient hardware-oriented designs are essential for enabling real-time deployment of high-dimensional CNNs in embedded and resource-constrained environments.

Reference

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1097–1105.

[3] S. Ji, W. Xu, M. Yang, and K. Yu, "3D convolutional neural networks for human action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, Jan. 2013.

[4] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3D convolutional networks," in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, 2015, pp. 4489–4497.

[5] J. Qiu, Q. Wang, and X. Yang, "High-dimensional convolutional neural networks," *IEEE Access*, vol. 7, pp. 180226–180237, 2019.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[7] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4700–4708.

[8] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[9] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1–9.

[10] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510–4520.

[11] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 3856–3866.

[12] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2017.

[13] J. Donahue *et al.*, "Long-term recurrent convolutional networks for visual recognition and description," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 2625–2634.

[14] X. Wang *et al.*, "Non-local neural networks," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7794–7803.

[15] A. Valada, C. Ronninger, and W. Burgard, "Deep multimodal fusion for robust perception," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2016, pp. 1–8.

[16] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "BinaryNet: Training deep neural networks with weights and activations constrained to +1 or −1," in *Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 3123–3131.

[17] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2016.

[18] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.

[19] Y. Chen *et al.*, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, 2016, pp. 262–263.

[20] C. Zhang, Y. Li, and J. Zhang, "Optimizing convolutional neural networks for high-dimensional data," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 11, pp. 3931–3942, Nov. 2020.