electronics

MDPI

Review

GPU-Based Embedded Intelligence Architectures and Applications

Li Minn Ang [1,*] and Kah Phooi Seng [2]

[1] School of Science and Engineering, University of the Sunshine Coast, Petrie, QLD 4502, Australia
[2] School of Engineering and Information Technology, University of New South Wales, UNSW Adfa, Canberra, ACT 2612, Australia; k.seng@unsw.edu.au
* Correspondence: lang@usc.edu.au

Abstract: This paper present contributions to the state-of-the art for graphics processing unit (GPU-based) embedded intelligence (EI) research for architectures and applications. This paper gives a comprehensive review and representative studies of the emerging and current paradigms for GPU-based EI with the focus on the architecture, technologies and applications: (1) First, the overview and classifications of GPU-based EI research are presented to give the full spectrum in this area that also serves as a concise summary of the scope of the paper; (2) Second, various architecture technologies for GPU-based deep learning techniques and applications are discussed in detail; and (3) Third, various architecture technologies for machine learning techniques and applications are discussed. This paper aims to give useful insights for the research area and motivate researchers towards the development of GPU-based EI for practical deployment and applications.

Keywords: embedded intelligence; GPU; multi-GPU; parallel architecture; machine learning

Citation: Ang, L.M.; Seng, K.P. GPU-Based Embedded Intelligence Architectures and Applications. *Electronics* **2021**, *10*, 952. https://doi.org/10.3390/electronics10080952

Academic Editor: Juan M. Corchado

Received: 10 December 2020
Accepted: 2 February 2021
Published: 16 April 2021

1. Introduction

Embedded Intelligence (EI) in products or systems gives it the capability to reflect on its own operational performance. EI requires the use of sensors, communications and computational processing that are embedded into the product or system to meet specific operational goals. In recent years, machine learning [1], deep learning [2] and artificial intelligence (AI) have seen wide adoption across different platforms and impose new requirements on existing computing systems and architectures. They can exist solely as software, but in most cases, they require the use of hardware components to build standalone intelligent machines. This is where the relation between "intelligence" and embedded systems becomes important. There are different platforms for the deployment of machine learning, deep learning and AI: (1) Graphics Processing Unit (GPU); (2) Field Programmable Gate Array (FPGA); (3) Central Processing Unit (CPU); (4) Application Specific Integrated Circuit (ASIC); and (5) Field Programmable System on Chip (FPSoC).

Over the past few years, advancements in GPU architecture have provided significant increases in computational power. GPU has hundreds of smaller cores and is designed for the types of computations used to render lightning-fast graphics. GPUs which were initially designed as the drivers for video game screen-to-controller responsiveness have impacted the entire IT industry. As accelerators have been adopted, GPUs have been extended for a wide range of other applications and are also deployed in high-performance computing systems to provide powerful computational capability. GPUs have been utilized to function as hardware accelerators [3] in speeding up training and inference of machine learning, deep learning and AI. The density of their cores and power efficiency enable them to meet real-time requirements and the intensive computation challenges of machine learning, deep learning and AI. Machine learning algorithms have seen wide adoption across different hardware platforms including GPU, for energy-efficiency, small form-factor and affordable devices and applications. Modern smartphones include hardware for

accelerating machine learning algorithms, and software frameworks have been optimized for embedded platforms, and hardware accelerators are being commoditized such as the edge Tensor Processing Unit (TPU) [4].

Similarly, GPUs are also widely used to serve the purpose of accelerating deep learning which has been demonstrated to be an effective tool for many applications. Emerging deep learning cloud services provided by AI service providers further accelerate the usage of deep learning in many business-critical processes. Deep learning platforms of large companies provide customized hardware such as servers, storage and networking communications to support the processing of high computational workloads. Different from the above, deep learning in centralized cloud environments incurs longer latency, energy usage and financial overheads. Research and development platforms have their own specific features and prefer focusing on cost-effective GPUs for the development of limited-scale computational clusters to handle diverse deep learning workloads. One trend can be observed in recently established competitions such as "low-power image recognition challenge" (LPIRC) [5] where there is a balanced emphasis on performance accuracy, computational throughput and power consumption. Other evidence is the increasing trends from major vendors and startups for providing low-power hardware accelerators for deep learning.

Some current development trends for GPU-based EI are towards: (1) Using lower precision arithmetic (e.g., from 32 bit to 16 bit representations); (2) Exploiting operations on sparse matrices (e.g., convolutional neural networks (CNNs) often have many zero weights which can be exploited with a mechanism termed ZeroSkip [6]). The ZeroSkip mechanism is discussed further in Section 3.6; and (3) Binary neural networks (BNNs) which are customized DNNs that use binary representations for weights and activations values. BNNs are discussed further in Section 3.1. Therefore, embedded intelligence with deep learning approaches for small and power efficient devices is attractive in several domains. Some examples of GPU-based EI for real-world domains and applications with social impacts can be found in predictive systems for disaster early warning and forecasting management (large-scale water supply systems management [7], simulation and forecasting for floods [8] and fires [9]) and usage in the electronics industry (circuit solver for electronic systems with large numbers of components [10]). Further examples of GPU-based EI applications and domains will be discussed in Sections 3 and 4.

Currently, a comprehensive survey or review for embedded intelligence research and development on GPU is not available. Most reviews on machine learning, deep learning or AI do not include or focus on hardware or embedded intelligence. This paper presents a comprehensive review and several representative studies of the emerging and current paradigms for embedded intelligence research and development on GPUs with the focus on the enabling technologies, applications and challenges. The overview and classifications of embedded intelligence research and development on GPUs are shown in Table 1. The research works are classified into the various categories: (1) First, the overview and classifications of GPU-based EI research are presented to show the full spectrum in this area which also serves as a concise summary of the scope of the paper; (2) Second, various architecture technologies for deep learning techniques and applications are discussed in detail; and (3) Third, various architecture technologies for machine learning techniques and applications are discussed. This paper aims to give useful insights for the research area and motivate researchers towards the development of GPU-based EI for practical deployment and applications. The remainder of the paper is as follows. Section 2 presents the overview and the different classification areas of EI research on GPU. This is followed by Sections 3 and 4 which give discussions on the GPU-based deep learning and machine learning techniques and applications. Section 5 concludes the paper.

Table 1. Classification Descriptors for Embedded Intelligence Research on GPU.

Classification Descriptor	References
GPU-based Deep Learning Technologies for EI	
Architecture framework and strategy	[11–25]
Scheduling and communication	[26–29]
Image processing and computer vision	[30–40]
Medical or health	[41–44]
Modeling or prediction	[45–51]
Convolution or performance analysis	[6,52–54]
VLSI placement	[55]
GPU-based Machine Learning Technologies for EI	
Architecture platform	[56–65]
Applications	[66–77]

GPU: Graphic Process Unit; EI: Embedded Intelligence.

2. Overview and Classifications of EI Research on GPU Architecture

Table 1 gives an overview and classification of EI research on GPU-based architecture technologies and applications. The table shows the full spectrum in this research area, and also serves as a concise summary of the scope of the paper. There are two main classification descriptors which are reviewed and discussed in this paper for EI: (1) GPU-based deep learning technologies for EI; and (2) GPU-based machine learning technologies for EI. The classification descriptor for (1) is further divided into seven sub-descriptors (architecture framework and strategy, scheduling and communication, image processing and computer vision, medical or health, modeling or prediction, convolution or performance analysis and VLSI placement). The classification descriptor for (2) is further divided into two sub-descriptors (architecture platform and applications). The right column in the table shows the relevant works and references which correspond to the respective sub-descriptors. This facilitates the rapid searching of the reviewed works for readers.

3. Deep Learning on GPU Architecture

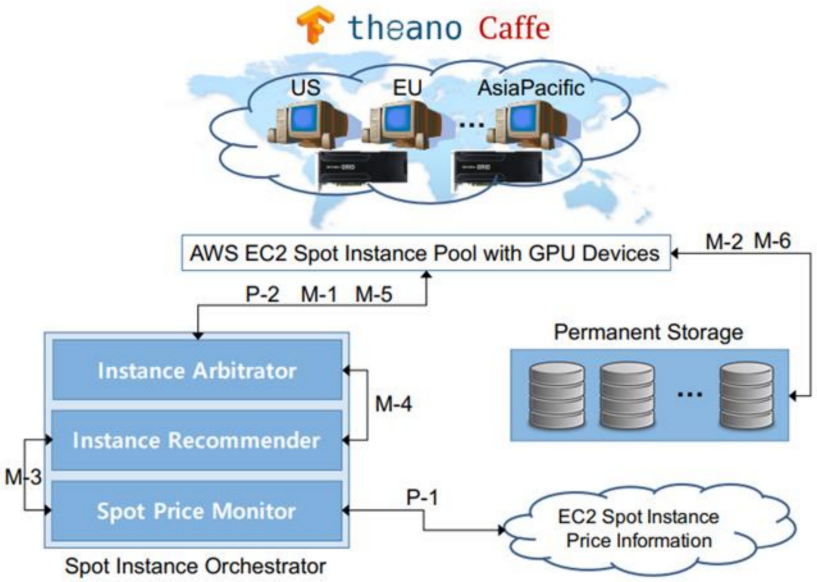
Deep learning approaches and techniques have been proposed and deployed to address many real-world problems such as bioinformatics, manufacturing, robotics, computer vision and natural language processing. Some well-known deep learning models are convolutional neural networks (CNNs) (e.g., AlexNet, GoogleNet, etc). Commercial cloud services offered by large technology companies have expanded the adoption of deep learning in various business critical processes. Several other deep learning frameworks have also been proposed such as TensorFlow (from Google), CNTK (from Microsoft) and Caffe 2 (from Facebook) to facilitate the training required for large datasets on GPU-enabled computational clusters.

3.1. Architecture Framework and Strategy

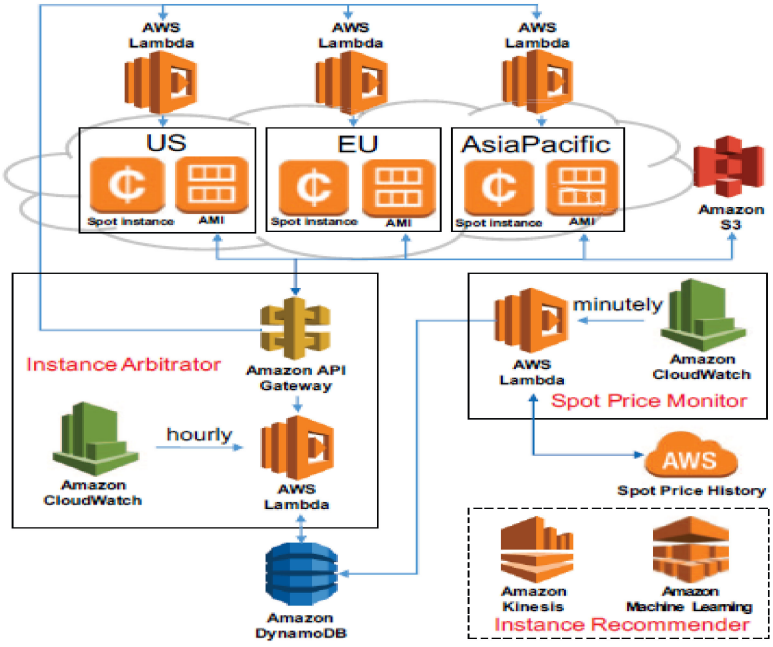
Ultra-deep neural networks (UDNNs) have been proposed to produce high-quality models. However, the training process for the UDNN is resource intensive and time-consuming. This limits the training efficiency of UDNN on modern GPUs due to the limited DRAM capacity. The authors in [11] proposed a new architecture called AccUDNN which is an accelerator to optimize the limited GPU memory resources to speed up the UDNN training. The architecture of AccUDNN consists of several interconnected modules: (1) The information collector gathers the features and attributes to build the performance model; (2) The performance model builder analyses the run-time characteristics and behavior in terms of computational performance, memory utilization and communication requirements; (3) The constraint unit develops the conditions that do not lead to performance degradation; and (4) The hyperparameter tuner computes the optimal minibatch

size to meet the efficiency constraints. Their experimental results showed that the proposed architecture resulted in a reduction of memory requirements for training the ResNet-152 from 24 GB to 8 GB.

The authors in [12] proposed the DeepSpotCloud architecture to execute deep learning tasks with cost efficiency and fault-tolerance. Figure 1 shows a system architecture of DeepSpotCloud. There are several important modules in this architecture: (1) The Spot Instance Orchestrator which executes the tasks for spot price monitoring, recommendation and arbitration; (2) The Spot Price Monitor which accesses the current GPU spot price; and (3) The Instance Arbitrator which monitors the running spots to identify the interrupted tasks. Their experimental results showed significant cost gain with a marginal increase in the task running time.



(**a**)



(**b**)

Figure 1. (**a**) DeepSpotCloud architecture and (**b**) the implementation of DeepSpotCloud [12]. AWS: Amazon Web Services. (Reprinted with permission from ref. [12]. Copyright 2021 IEEE).

The authors in [13] proposed a scalable architecture for large-scale DNN training in a distributed environment. The framework consists of a four-tier technology stack: (1) Hardware infrastructure consisting of Nvidia CUDA GPU cluster and nodes; (2) Nvidia CUDA drivers; (3) Middleware for HDFS storage and cluster resource management (Apache Flink); and (4) The application framework which enables the user to manage the storage of huge amounts of data (GB). The framework also includes the learning pipeline for scaling across distributed environments, a neural network package and deep architecture builder, GPU executor, linear algebra library and data format parsers.

A significant challenge for training DNNs such as CNNs is the high amount of computational resources and memory bandwidth. The design optimization requires the accurate modelling of how the GPU performance increases with the available computational and memory resources. The authors in [14] proposed DeLTA which is an analytical GPU model for CNNs for various parameters (arithmetic performance, memory hierarchy traffic, data reuse) to optimize computation throughput and memory bandwidth. Their work deployed two NVIDIA Pascal GPUs (P100 and TITAN Xp) and a Volta GPU (V100) and was validated on four CNN architectures (AlexNet, VGG, GoogLeNet, ResNet). Their experimental results showed that the DeLTA architecture could be used for the resource space exploration and the identification of tradeoffs using various scaling parameters for GPU designs to meet different design requirements.

The authors in [15] proposed GRAMARCH, a heterogeneous 3D NoC-enabled GPU and ReRAM (Resistive random-access memory) architecture that exploits the advantages of ReRAM and GPUs for 3D Network-on-Chip. Figure 2 shows the proposed GRAMARCH architecture. The GRAMARCH architecture consists of two layers: (1) The bottom layer contains the GPU and Last Level Cache (LLC) titles, and (2) The top layer contains the ReRAM for storage and computation, and includes the eDRAM buffers, in-situ multiple-accumulate units (IMA), output registers, shift-and-add, sigmoid and max-pool units. Their experimental results showed performance improvement of up to 53 times compared to conventional GPUs for image segmentation. A further paper by the authors in [16] proposed an M3D-enabled architecture termed AccuReD, that combined ReRAM arrays together with GPU cores for training CNNs with high performance and accuracy. Their experimental results showed that the proposed architecture could accelerate the CNN training process by up to twelve times compared to conventional GPU platforms.

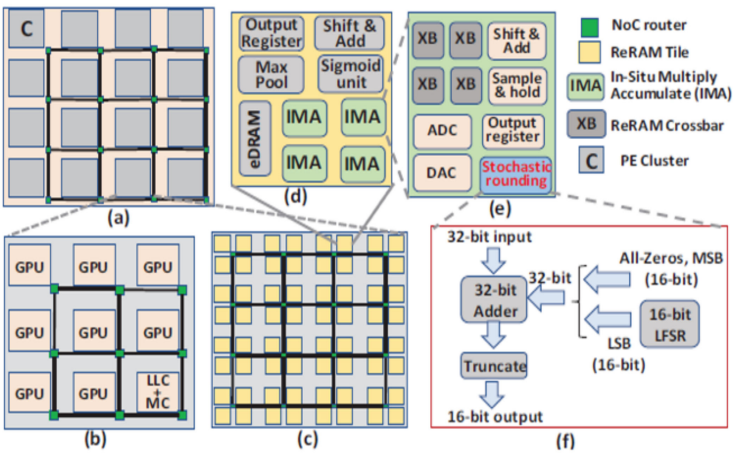


Figure 2. Proposed GRAMARCH architecture [15]. (Reprinted with permission from ref. [15]. Copyright 2021 IEEE). (**a**) Top view of proposed architecture, (**b**) Lower logic layer, (**c**) Upper ReRAM layer, (**d**) ReRAM tile, (**e**) In-situ Multiply Accumulate (IMA) and (**f**) Stochastic rounding implementation.

The authors in [17] proposed a performance model for an asynchronous SGD deep learning system called SPRINT on GPU supercomputers. There are two stages in the training strategy: (1) Computation of cost derivatives, and (2) Addition of derivatives and current weights for weights updating. In their proposed architecture, the GPU threads

process computes the gradient of randomly-picked samples and accumulate them to the host memory, and the update threads process executes MPI all-reduce to update the weights. Their experimental results used 192 GPUs and gave an average error of 5% to 19% for various DCNN architectures to perform image classification tasks on GPU supercomputers.

The authors in [18] proposed the vDNN++ (virtualized DNN) architecture. The proposed VDNN++ had three main solutions: (1) improved asynchronous transfer; (2) usage of heuristics to reduce the memory fragmentation; and (3) compression to reduce the memory footprint. Their experimental results showed that the performance of the proposed approach (vDNN++) gave an improvement of up to 60% compared with a previous approach (vDNN).

The authors in [19] proposed an approach for training RNNs (recurrent neural networks) on multiple GPUs. Their approach partitions the training data using map-reduce and batch-bucketing optimization and is distributed among the GPU processes. Their experimental results validated the approach and compared the approach using a different number of buckets for a number of performance parameters (clock cycle time, number of epochs and loss value).

The authors in [20] proposed a pipeline-hybrid parallelism training approach for training DNNs termed Pipe-Torch. In their approach, they formulated DNN training with pipeline-hybrid parallelism, and designed an algorithm to map and model the GPU cluster with a heterogeneous network environment. Their experimental results showed that the Pipe-Torch architecture and approach gave a 1.4 times performance speedup improvement.

The authors in [21] proposed a combination design of parameter-server and all-reduce schemes. Their proposed design deploys the asynchronous parameter-server aggregation for node communications among worker nodes and the synchronous all-reduce method for aggregation among intranode GPUs. Their experimental results showed that the approach led to computational performance improvement and increased the resource utilization in the GPU cluster.

The authors in [22] proposed an architecture to pipeline the training of multiple deep learning models on a GPU cluster. The dataset is partitioned to be handled by a specific GPU computation node. In their approach, each computation node uses preloaded data for training a new model or refining a partially trained model from a previous node. The authors also proposed a circle-based pipeline where data are distributed at each computation node. Their experimental results showed that the proposed pipeline architecture and framework could reduce the overall training time by several hours compared to the baseline method.

Another significant challenge is for the deployment of DNNs which have high computational complexity on mobile devices with the power and performance constraints. A solution is to use binary neural networks (BNNs) which are customized DNNs that use binary representations for weights and activations values. The authors in [23] proposed PhoneBit, a GPU-accelerated BNN inference engine which can be deployed on Android-based mobile devices. Their work was validated using BNN variants for DNNs (AlexNet, YOLOv2 Tiny and VGG16). Their experimental results showed that the PhoneBit gave significant performance speedup and energy efficiency advantages when compared with other DNN approaches and deployments on mobile devices.

The authors in [24] proposed a BNN hardware accelerator design. Figure 3 shows their proposed BNN accelerator design. The architecture consists of many PEs (processing elements) which can be increased or decreased. The architecture can be scaled by adding more PEs. The proposed accelerator approach supports the hardware operations to perform the computations for the BNNs. Figure 4 illustrates the sequences of operations that a processing element takes to process a BNN layer. Their architecture was implemented on Aria 10 FPGA and their experimental results showed that the FPGA BNN implementation gave higher performance efficiency over the CPU (Xeon server) and GPU implementations (Titan X and TX1).

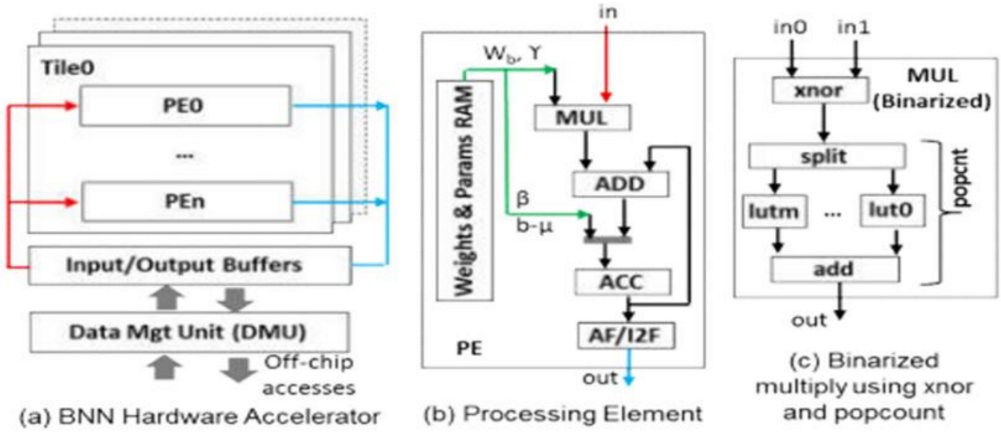


Figure 3. BNN hardware accelerator architecture [24]. (Reprinted with permission from ref. [24]. Copyright 2021 IEEE).

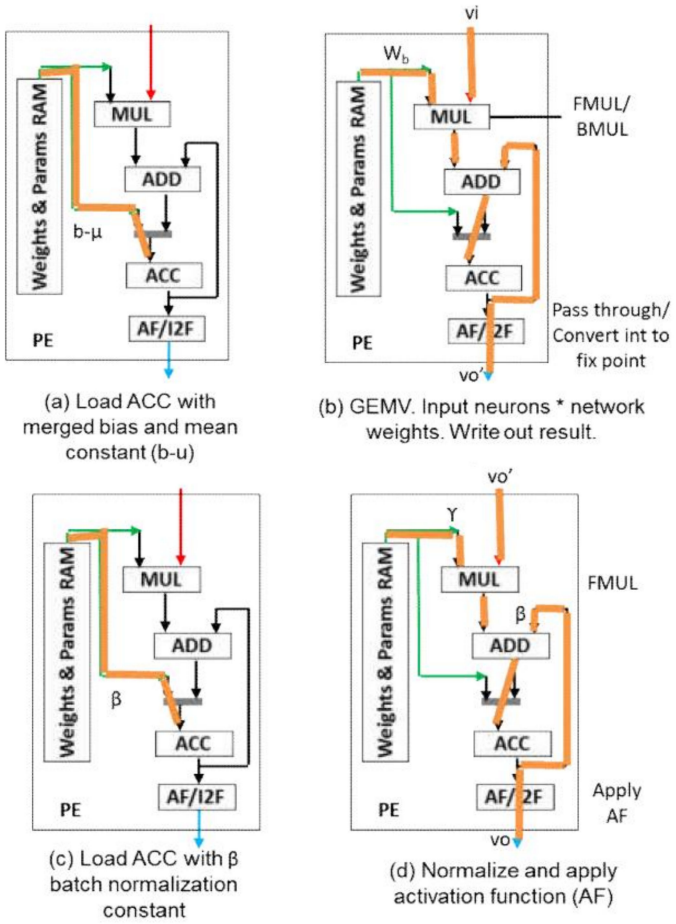


Figure 4. Sequences of operations that a processing element takes to process a BNN layer [24]. (Reprinted with permission from ref. [24]. Copyright 2021 IEEE).

The authors in [25] proposed a power-efficient CNN implementation for FPGAs and GPUs to accelerate the DNN computations. Their GPU-FPGA approach implemented a DNN architecture (CNN). It contained six layers (three convolutional layers, two max pooling layers and one fully-connected softmax activation layer). The DNN architecture implemented the fully-connected layer on the FPGA and used the GPU for the other components. Their experimental work used a TX2 GPU (from Nvidia) and a proposed Artix-7 FPGA (from Xilinx), and showed increased computation speed and decreased power consumption.

3.2. Scheduling and Communication

The authors in [26] proposed an approach termed GENIE for dynamic quality of service (QoS) scheduling for a GPU cluster framework. The GENIE scheduling framework. contained two sections: (1) Offline profiling and (2) Online scheduling. The offline profiling section implements the prediction model based on the characterization of workloads, whereas the online profiling section implements the scheduling strategy based on the QoS specifications. Their experimental results were conducted on a GPU cluster and showed an improvement of 67% in comparison with other baseline schedulers.

The authors in [27] studied the communication requirements for DNN training on GPU and identified hardware approaches to reduce the bandwidth. Their proposed architecture focused on utilizing distributed GPU software tools (e.g., CUDA and MPI). The authors investigated the communication overheads involved in training DNNs for both weak and strong scaling of training and showed that overlapping could significantly reduce them. The authors evaluated the performance of these methods on microbenchmarks and end-to-end training using the opensource LBANN deep learning toolkit, and showed that their approach could improve on existing methods, particularly for larger-scale DNN implementations.

The authors in [28] modelled broadcast communication schemes and analyzed the performance of various approaches on GPU clusters. Their work investigated several different broadcast algorithms (ring, K-nomial and IB-MCAST) on GPU clusters. The IB-MCAST header is stored in the host (in this case CPU) memory and the GPU data are collected in a single step via an IB gather operation with the GDR feature enabled. Their experimental results gave a 68% latency reduction in comparison with other state-of-the-art methods. The authors in [29] proposed an approach to improve the performance communications between GPU nodes by reducing the bottlenecks and I/O communications in CUDA aware MPI runtimes. In this approach, minibatches are further divided into several sub-minibatches and copied to GPU buffers. Their work was validated on the Cray CS-Storm cluster and the GPU cluster contained 12 nodes containing 8 NVIDIA K-80 GPUs. The results showed that their designs gave performance improvements of 23%, 21% and 15% for the CIFAR-10, MNIST and ImageNet datasets, respectively.

3.3. Image Processing and Computer Vision

The authors in [30] proposed a GPU-based framework for training 3D CNNs using the voxelization of polygonal models. Their approach uses geometric transformations and vertex displacement computations for data augmentation in the GPU. The authors framework was tested with the ModelNet10 and ModelNet40 datasets from Princeton. In their approach, the voxelization was carried out on-the-fly compared with the standard approach of carrying out the voxelization separately. Their experimental work which was implemented using C++ with a single-thread implementation showed that their approach gave higher performance in comparison with the standard method which performs the voxelization of models separately.

The authors in [31] proposed an efficient GPU-based image target detection approach using CNNs. Their proposed approach focused on accelerating the high-level detection and scheduling task and less on accelerating the low-level convolution computations. Figure 5 shows their proposed approach which consists of three modules: (1) Sliding-window module—to preprocess the image; (2) parameter-generating module—to perform the control and generates a combination instruction which is used as input into the CNN module; and (3) CNN feedforward execution module.

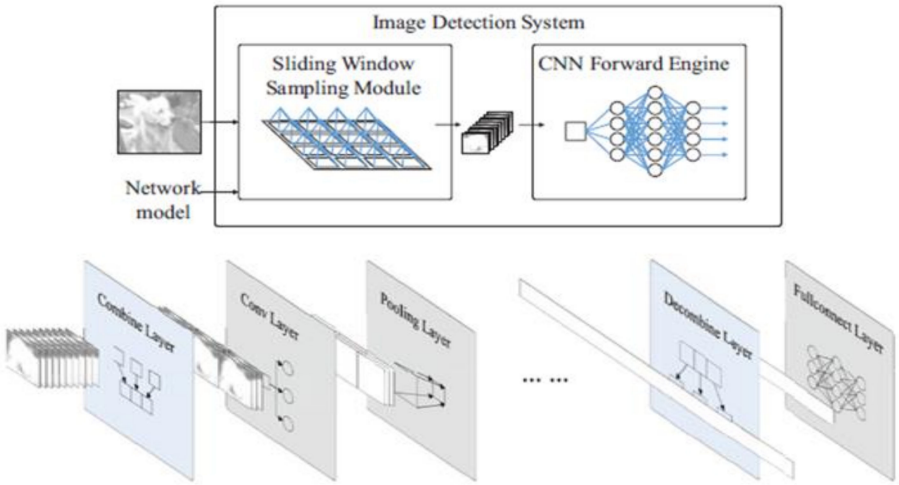


Figure 5. GPU-based approach for image target detection [31]. (Reprinted with permission from ref. [31]. Copyright 2021 IEEE).

The authors in [32] presented a GPU-based real-time stereo estimation architecture termed StereoBit using the BNN (binary neural network) for computation of the disparity map. Figure 6 shows their proposed approach. Their experiments used a StereoBit network architecture with four layers. Their experimental results used a TITAN XP GPU and showed that the framework gave a performance improvement for the stereo computations (60 fps) and reduced the memory usage for storing parameters.

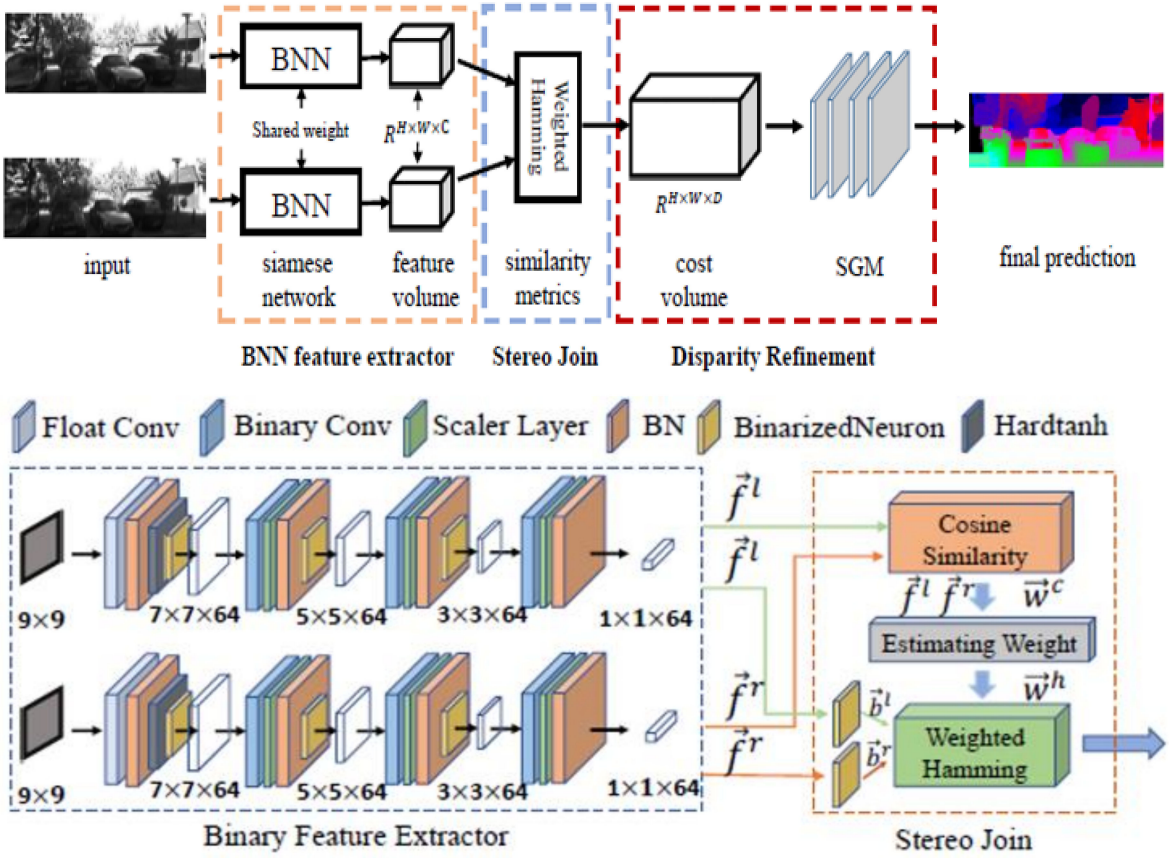


Figure 6. GPU-based approach for real-time stereo estimation [32]. (Reprinted with permission from ref. [32]. Copyright 2021 IEEE).

The authors in [33] proposed a hardware/software GPU-based design for an embedded image recognition system. Their proposed GPU-based embedded image recognition approach used the NVIDIA GPU to accelerate the computations for an immune CNN to perform the image recognition. Their experimental results showed that the proposed hardware/software system design gave improved recognition performance and computational speed compared with the traditional approach.

The authors in [34] proposed a real-time face recognition architecture using DNN on an embedded GPU system. Their face detection system involved tracking faces and using a CNN to perform the recognition task. The main challenge of the design is the computational requirements of the CNN when implemented into a hardware platform with limited computational resources. Their experimental work used a Jetson TX2 which contains a high-performance processor in a power-efficient design. The authors in [35] also worked on the image/face recognition. They performed video analysis to identify the face attributes and to analyze the gender and age features of the customers in business. The face recognition task was realized using a convolution neural network (CNN) on a hardware platform utilizing an embedded GPU.

The authors in [36] developed a GPU-based emotion recognition robotic eye for a service robot. Human emotional states such happy, sad and relaxed were recognized by a trained deep learning architecture (ConvNet). The robot eye performs two tasks in the robotic system. Human heads and faces are processed by pretrained haar cascade classifiers, followed by the recognition of human emotional states by the trained ConvNet. The authors in [78] proposed a bird species classification system using deep learning (CNN) on a GPU parallel computing platform. Their approach used the Caltech-UCSD Birds 200 data set for training the CNN. Their experimental work was performed using TensorFlow and showed that the DCNN computing platform and implementation could predict 88% of bird species accurately.

The authors in [37] proposed an embedded GPU cluster computing approach for the inference of CNN. Their work involved the use of pre-trained CNN models developed by the U.S. Air Force Research Laboratory. Their experimental platform consisted of six Jetson TX2 development boards which were operated in parallel. The TX2 development board cluster was used to perform evaluations for a range of image sizes, tile sizes, tile overlaps and network configurations. Their experimental results showed that the parallel GPU cluster computing framework gave a performance increase of 4.3 times. The authors in [38] applied CNN for the classification of adjective noun pairs on a GPU cluster. Their approach used the ResNet50 CNN with 50 layers in their experiments that was used to map a $224 \times 224 \times 3$ image to a 1200-dimensional vector for representing a probability distribution over the various classes.

The authors in [39] proposed an efficient gradient-based neural architecture search approach. Their approach represents the search space as a DAG (directed acyclic graph) which contains multiple sub-graphs. Each sub-graph represents a particular neural architecture. The authors used a differentiable sampler over the DAG to avoid the need for traversing all the sub-graph possibilities. Their experimental work showed that the approach was computationally efficient and reduced the search cost of the standard approach by about 104 times. In addition, the CNN and RNN models which were discovered indicated a competitive performance compared to state-of-the-art models. The authors in [40] evaluated the reliability of the YOLO object detection framework. In their approach, they used GPUs designed with various architectures (Pascal, Kepler, Maxwell) for the detection of objects. The authors also proposed an approach termed the Algorithm-Based Fault-Tolerance (ABFT) technique for the matrix multiplication kernels of DNNs. Their experimental work used the Caltech and Visual Object Classes datasets and gave 50–60% detections and corrections with induced corruptions.

3.4. Medical or Health

Magnetic resonance imaging (MRI) is popular in modern medicine. To obtain satisfying resolution within a limited scanning time, some fast MRI image reconstruction algorithms based on parallel imaging such as GRAPPA are used. RAKI is an improved GRAPPA-like method that reconstructs the under-sampled k-space with estimated point-spread functions (kernel) in the *k*-space. In RAKI, kernels are trained by a multi-layer CNN. However, RAKI reconstruction depends on multiple CNNs and the implementation of RAKI reconstruction is a time-consuming process. The authors in [41] proposed accelerated strategies for RAKI implementation aided by GPU parallel programming and TensorFlow, a popular deep learning framework. In their approach, they limited the iteration number of solving optimization problems in the CNN training, while ensuring a satisfying result. They also parallelized the training tasks by CPU multiprocessing to maximize the performance by fully utilizing GPU resources to achieve further acceleration. Their experimental work used two Intel E5–2643 CPUs and a NVIDIA Tesla K80 GPU and showed that their approach gave more than 60 times performance improvement compared with the conventional sequential implementation.

Computed tomography (CT) imaging is used for several applications from diagnosis to health care. The authors in [42] proposed a fast reconstruction technique termed Deep Learning Model Based Iterative Reconstruction (DL-MBIR). Their proposed DL-MBIR approach was trained using a 16-layer residual CNN to produce reconstructions that approximated MBIR images. Their experimental work was implemented using multiple GPUs (Titan X) and TensorFlow, and showed that the 2.5D reconstruction method achieved similar quality to the 3D reconstruction at a lower computational cost. Cervical cancer is a common cancer in women. The authors in [43] proposed an approach to classify the cervix type using DNN and GPU. Their experimental results used TensorFlow and Keras models and techniques and a CUDA parallel computing platform. The CUDA architecture provided a large API for applications.

Birth asphyxia (perinatal asphyxia) is a medical condition characterized by abnormal breathing patterns in newborn children. The authors in [44] developed a CNN approach to enable asphyxia to be determined at its early stages. Their approach observed the patterns in a children's cries after training the CNN on samples obtained from affected children. Their experimental work was implemented on NVIDIA DIGITS and gave a classification accuracy of 94%.

3.5. Modeling, Prediction and Memory

To achieve exascale computing, energy-efficiency is an essential parameter for high-performance computing systems. The authors in [45] proposed an approach to model GPU parameters (computational performance, energy and power consumption). Their work aimed to predict the changes in the computational time and energy consumption for different scaling and frequencies of various GPU domains. Their approach used the LSTM recurrent neural model which includes the input of the PTX assembly code of a GPU kernel. Their experimental work was performed on four GPU architectures (Tesla T4, Titan V, Titan Xp and GTX Titan X) and achieved energy savings of 8.0% for Tesla T4, 6.0% for Titan V, 29.0% for Titan Xp and 11.5% for GTX Titan X.

The limited and small GPU memory poses challenges to fit the full model and data and limits the size of the training model. To improve the efficiency of training a large deep neural network on a GPU, the authors in [46] proposed a data pinning algorithm that reduces GPU memory pressure. They also analyzed access patterns of deep learning computation, formally formulated the GPU data pinning problem, and provided a data pinning algorithm that reduces the data movement overheads between the CPU memory and GPU memory. Their technique reduced the GPU memory usage during the backward phase by performing gradient computation and weight update simultaneously. Their experimental results used the VGG-19 deep neural network implementation model and showed improvements over other approaches (e.g., GeePS [47]).

The authors in [48] proposed an approach for overhead reduction between a host PC and the GPU method based on data swapping for reducing the memory usage. The proposed method introduced the virtual layer concept on a GPU. Their experimental results showed that their approach could reduce memory usage by about 73% and the training time by about 33% compared with the traditional method. The authors in [49] proposed a memory management approach for efficient GPU memory utilization. Their approach reduced the I/O contention among many GPUs by performing an adjustment of the number of GPUs that can perform swapping operations on a specific layer. The authors also proposed an intelligent pre-fetching algorithm that operated from the CPU memory to the GPU. Their experimental results showed that their proposed approach could achieve high throughput processing while sustaining a large batch size.

The authors in [50] proposed the design and technique for an out-of-core cuDNN library which enables CNN computations which exceed the size of the GPU memory capacity. The software stack using out-of-core (OOC) cuDNN includes extensions of some CUDA functions for memory management. Their approach was validated by computing a CNN requiring 60 GB memory on a GPU with a capacity of 16 GB physical memory. The authors in [51] proposed a memory-efficient architecture for GPU termed GPU-only Unified ConvMM. In their approach, the matrix multiplication convolutional layer is accelerated using the parallel computational resources of the GPU. Furthermore, a unified memory architecture is used to optimize the flow of this GPU-only ConvMM layer. The unified memory (UM) architecture which contains the shared memory for the CPU and GPU. Their experimental results showed that the performance of the proposed algorithm could be improved using the GPU implementation because the GPU implementation minimized the data transfers between host and device and could provide improvements in performance.

3.6. Convolution and Performance Analysis

Convolution operations form a large part of the computational requirements for CNNs. The authors in [6] targeted to enhance the performance of the state-of-the-art convolution algorithm (termed Winograd convolution) on the GPU. To address the issue of CNNs often having many zero weights, they proposed a hardware mechanism (termed ZeroSkip) to skip the multiplication operations that will give zero results regardless of input data. Their approach used a customized hardware module in the GPU which would dynamically check for zero elements and manipulate the program counter (PC) accordingly. Their experimental results used a Titan X and showed that the ZeroSkip method gave a 51.8% higher performance than the baseline Winograd convolution.

The use of distributed deep learning (DDL) presents some challenges in terms of portability and scalability issues. The authors in [52] provided a detailed performance analysis of distributed TensorFlow using the Horovod framework for scalability under various runtime parameters. The authors implemented DDL approaches for three architectures (AlexNet, GoogleNet and ResNet50). Their experimental work used the Nvidia K40, K80 and P100 GPUs. Their testing used synthetic image data with various parameters (e.g., batch size and number of GPUs). Their results showed that the Horovod framework gave linear performance (images/s) for scalability up to 256 GPUs.

The authors in [53] proposed a fault-tolerant approach with soft errors as faults in the network and introduced triple modular redundancy to detect and rectify the faults. Their work showed that the presence of faults in the network would cause a significant decrease in the accuracy of the network. CNNs and other neural networks use activation functions to introduce nonlinearity into models for the computation of complex functions. The authors in [54] investigated four activation functions (sigmoid, hyperbolic tangent, rectified linear unit (ReLU) and exponential linear unit (ELU)) by implementing them on CNN using a GPU. Their experiments used the Nvidia GPU 940MX for training and testing of the CNN model. Their work showed that the ReLU activation function had better performance than the sigmoid and hyperbolic tangent activation functions. Another observed result was that the ELU had better performance than ReLU.

3.7. VLSI Placement

Analytical placement techniques represent the current state-of-the-art for VLSI placement with the capability to solve a nonlinear optimization problem. The authors in [55] proposed a GPU-accelerated placement approach termed DREAMPlace. The DREAMPlace algorithm implementation used deep learning toolkits which consists of three stacks (low-level operators, gradient derivation and optimization engines). Their experimental results showed that their DREAMPlace approach had a performance improvement of thirty times speedup in global placement compared to other state-of-the-art methods.

4. Machine Learning in GPU Architecture

Machine learning (ML) algorithms have seen wide adoption across different domains and hardware platforms in recent years. This section describes different types of machine learning techniques for embedded intelligence on GPUs.

4.1. Architecture/Platform/Framework and Strategy

The authors in [56] proposed a parallel approach termed the H-ELM (hierarchical extreme learning machine) algorithm based on GPU and Flink which is an in-memory cluster computing platform. Figure 7 shows the architecture and workflow of H-ELM. Flink uses Java interfaces to communicate with the GPU. The GFlink architecture is shown in Figure 8.

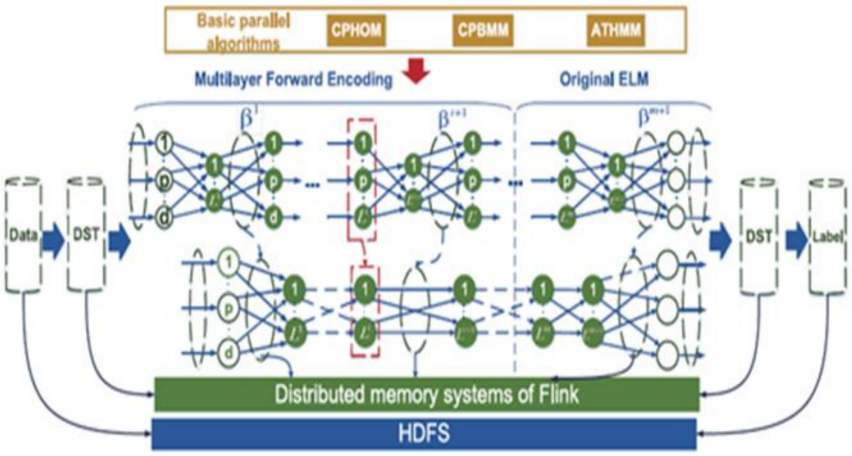


Figure 7. H-ELM GPU architecture and workflow [56]. (Reprinted with permission from ref. [56]. Copyright 2021 IEEE).

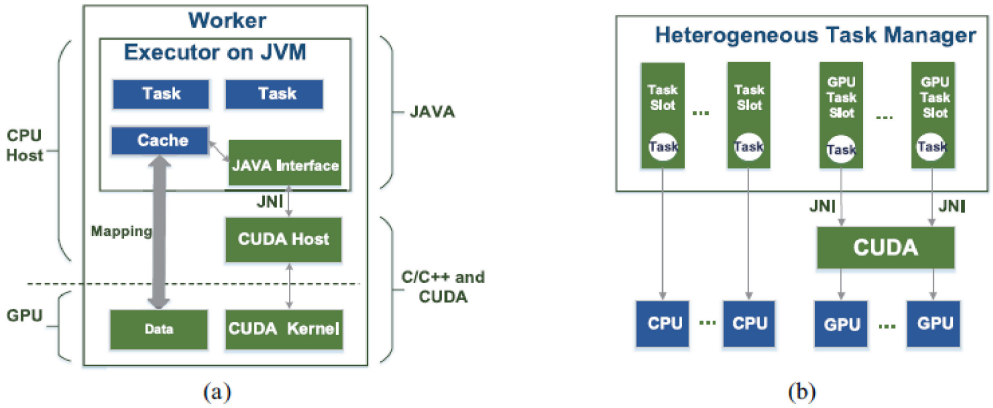


Figure 8. GFlink: (**a**) Architecture of a work node, (**b**) Heterogeneous task management [56]. (Reprinted with permission from ref. [56]. Copyright 2021 IEEE).

The authors in [57] proposed a GPU architecture which is integrated with the Spark Big data framework. The HeteroSpark clusters can be visualized as Spark clusters with GPUs connected with Spark worker nodes. The HeteroSpark architecture approach integrated the GPU accelerator into the Spark framework to increase data parallelism and algorithm acceleration. Their experimental work validated the HeteroSpark architecture using popular machine learning applications, and showed that the HeteroSpark architecture could achieve a performance improvement of 18 times.

The authors in [58] proposed an efficient GPU-based MapReduce framework to accelerate SVM learning. In their framework, GPUs were deployed to perform the parallel numerical calculations and MapReduce was used to perform parallel task scheduling and processing. In their approach, the search task for the SVM is performed in parallel using the MapReduce computational model. The experimental results showed that their proposed GPU-based MapReduce framework had significant performance improvements for SVM learning.

The authors in [59] proposed fast and low precision learning for GPU-accelerated spiking neural network (SNN) simulator architecture termed ParallelSpikeSim. The ParallelSpikeSim simulator uses unsupervised learning and stochasticity and enables fast and accurate learning with low precision operations. Their experimental results showed up to two to three times speedup in learning performance compared to deterministic SNN architectures for simple and complex datasets.

The authors in [60] proposed an event-based and time-driven SNN simulator for a hybrid CPU–GPU platform. The authors performed a comparative study for the different simulation methods (event-driven and time-driven methods) with various simulation techniques and computational platforms (single-core CPU, multi-core CPU and GPU). Their experimental work implemented the EDLUT (event-driven neural simulator based on lookup tables) in CPU/GPU clusters, and their results showed improvements in the spike propagation and queue management time. The authors in [61] proposed a neural accelerated architecture for GPUs termed NGPU which enables the scalable integration of neural accelerators for large numbers of GPU cores. The proposed architecture has three features for improving performance: (1) elimination of fetch/decode during the neural execution; (2) reduction of memory/register file accesses by storage of partial results and parameters in dedicated small buffers within the SIMD lanes; and (3) implementation of the sigmoid using a lookup table. Their experimental results showed that NGPU had a 2.4 times average performance improvement speedup and a 2.8 times average energy reduction for a diverse set of benchmarks.

The authors in [62] proposed a novel machine learning approach to find the optimal choice of GPU memory requirements for CUDA applications. The workflow of the proposed approach has two phases: (1) Offline learning; and (2) Online inference. The Offline learning phase used the NSight CUDA Profiler to collect a set of profiling metrics. Their work investigated various classifiers including the random forest, random tree and LogitBoost to identify the classifier which would give the highest accuracy. Their experimental results showed that the proposed approach was able to predict accurately the optimal memory requirements for discrete memory or unified memory space.

The authors in [63] proposed a generic sparsity pattern termed Regularized Multi Block (RMB) sparsity pattern, an efficient storage format (CRMB), and a fast GPU algorithm for processing the RMBMM (SDMM with the multiplicand having the RMB sparsity pattern). Figure 9 shows the CRMB storage format for storing an RMB sparse matric. Their work showed that the RMB sparsity pattern enabled efficient implementations for parallel algorithms and a reduction in storage for a sparse matrix.

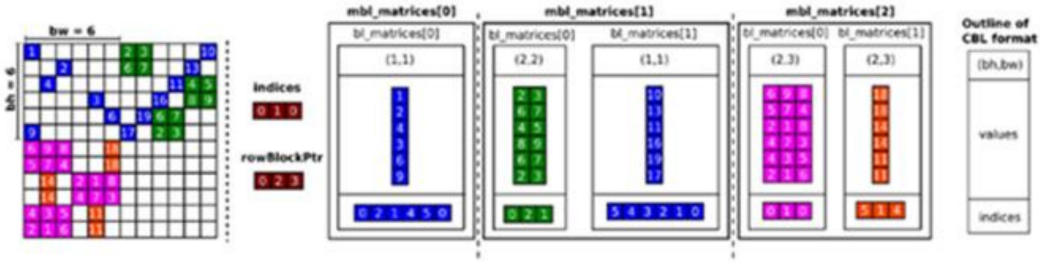


Figure 9. Regularized Multi Block (RMB) sparsity pattern for GPU memory [63]. (Reprinted with permission from ref. [63]. Copyright 2021 IEEE).

The authors in [64] proposed an approach to allocate computations on GPU kernels to optimize the problem parameters (neural structure and training set size) and GPU parameters. Figure 10 shows the neural model and its mapping into a GPU. They termed this representation BNL (basic neural layer) with the objective to optimize the computational speed for a matrix of samples. Their experimental results showed that performance increases of 100 to 250 times could be achieved by optimizing the number of threads and the GPU global memory.

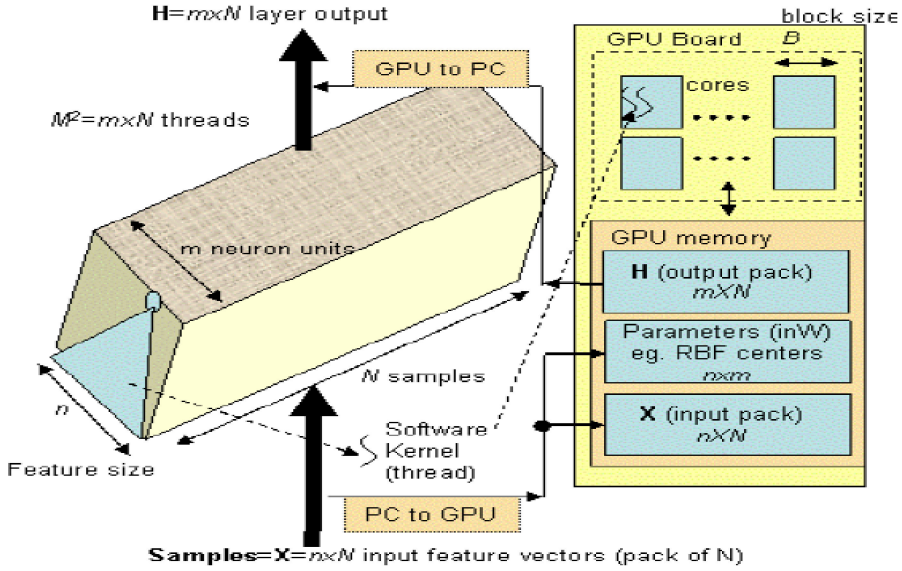


Figure 10. BNL neural model and mapping to GPU [64]. (Reprinted with permission from ref. [64]. Copyright 2021 IEEE).

The authors in [65] proposed the implementation of a multichannel HNN (MHNN) on a GPU platform. Their work used three levels of parallelism (thread, block and stream) and was designed according to the MHNN-based unmixing procedure. Their experimental results showed that the GPU implementation gave improvements of several hundred times compared to the sequential CPU implementation.

4.2. Applications

The authors in [66] proposed an approach to optimize the energy consumption of a GPU by using dynamic voltage and frequency scaling (DVFS). Their approach implemented the DVFS energy management model in a GPU. Their experimental results used three GPU platforms (Tesla, Fermi and Kepler) and showed improvements for performance and power. The authors in [67] proposed another GPU and memory coordinated energy saving approach using the ELM termed EDVFS. The EDVFS energy saving approach performs the voltage and frequency adjustments based on the extracted runtime characteristics. Their experimental results showed that the EDVFS approach gave a maximum energy savings

and average energy savings of 10.63% and 2.68%, respectively, when compared to the traditional DVFS.

The authors in [68] proposed a GPU approach for PLV (phase locking value) biomarkers. Their experimental results showed that their approach gave a 21.3 times improvement from a search space of over 1.2 million and a significant reduction in complexity for on-device processing. The authors in [69] proposed an approach for an efficient GPU-based implementation of multivariate empirical mode decomposition (MEMD) for massive neural data processing. The authors exploited the computational power of GPUs for parallelizing the MEMD algorithm using CUDA. The parallel MEMD implementation used the recursive sifting process which is performed until the stopping criterion. Their experimental results showed an improved performance of 6 to 16 times compared to traditional PC-based implementations.

The authors in [70] explored the usage of GPUs for simulating large-scale neuronal networks based on the AdEx (Adaptive Exponential) neuron-model. Their approach used the AdEx with STDP synapses model. Their implementation used the Thrust GPU library to bypass the expensive host-device memory operations. Their experimental results showed that the optimized GPU implementation gave a performance improvement of fifty times when compared to a reference multicore implementation. Furthermore, their work showed that a dual-GPU configuration could push the acceleration to a speedup of ninety times for networks of 20,000 neurons.

The authors in [71] proposed a GPU Simulator of MLMVN (multilayer neural network with multi-valued neurons). The MLMVN requires fewer neurons and iterations for its learning algorithm to optimize a network learning error. Their experimental work showed that utilizing the massive parallelism of a GPU could give a thirty times performance improvement for the MLMVN learning process. The authors in [72] proposed a novel approach for ECG recognition over GPU platforms using the probabilistic neural network (PNN). They also studied various features (discrete wavelet transform (DWT), autoregressive (AR), and wave Statistic) for the PNN. Their experimental results showed improvements in computational time using a parallel PNN (P-PNN) using CUDA. The algorithm performance of P-PNN in terms of the recognition rate was also improved compared to other learning models such as SVM and ELM.

The authors in [73] proposed an approach for fast soma cell detection in knife-edge scanning microscopy (KESM) for high-throughput imagery using GPU-accelerated machine learning. Their proposed approach requires few training data and performs real-time cell detection at a rate that exceeded the data rate of traditional KESM. The authors in [74] proposed a parallel implementation of chaos neural networks for an embedded GPU using OpenCL (Open Computing Language). Their experimental results showed that the parallel CNN approach implemented on an embedded GPU gave a pseudo-random number generator that was 49% faster than the AES in counter mode.

The authors in [75] proposed an approach to identify abnormal behaviors for anomaly-based intrusion detection system (IDS). The training and operation of the neural architecture on GPU are performed in three stages: (1) preparation of initial data; (2) transfer of data to the CUDA device; and (3) kernel routine and transfer of the result to the host device. Their experimental results showed that the GPU implementation gave a thirty times performance improvement compared to a CPU implementation.

The authors in [76] proposed an approach for robot trajectory generation and the computation of collision-free trajectories for robot swarms using GPU. The performance of the proposed framework was evaluated on two case studies: (1) a swarm of 200 quadcopters traversing a maze and (2) a fleet of 100 bicycle robots changing their formation. Their experimental results showed that the proposed method required just seconds to compute the feasible and collision-free trajectories for the entire swarm. The authors in [77] proposed the GPU WiSARD Vessel Tracker GWVT which tracked maritime vessels in dynamic environments (varying maneuverability, appearance and shape). The GWVT used the WiSARD weightless neural network and was implemented on a GPU. The parallel processing capa-

bilities of the GPU enabled fast execution of the tracking algorithm. Their experimental results showed that the GWVT gave improved performance over a CPU tracker.

5. Conclusions

This paper has given a comprehensive survey of the research area of embedded intelligence (EI) for GPU-based architectures and hardware implementations. The review has covered both the newer deep learning approaches and the traditional machine learning approaches. The area of GPU memory scheduling and communication has also been included. GPU-based EI applications for several areas including image processing and computer vision, medical applications, modeling or prediction, convolution or performance analysis and VLSI placement have also been discussed to demonstrate the wide potential of these EI technologies for real-world deployments. The paper aims to be useful and to motivate researchers towards performing increasing research in this emerging and important technology area.

Author Contributions: Conceptualization, L.M.A. and K.P.S.; writing—original draft preparation, K.P.S.; writing—review and editing, L.M.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Jordan, M.I.; Mitchell, T.M. Machine learning: Trends, perspectives, and prospects. *Science* **2015**, *349*, 255–260. [CrossRef] [PubMed]
2. Deng, L.; Yu, D. Deep learning: Methods and applications. *Found. Trends Signal Process.* **2014**, *7*, 197–387. [CrossRef]
3. Aluru, S.; Jammula, N. A Review of Hardware Acceleration for Computational Genomics. *IEEE Des. Test* **2014**, *31*, 19–30. [CrossRef]
4. Belletti, F.; King, D.; Yang, K.; Nelet, R.; Shafi, Y.; Shen, Y.-F.; Anderson, J. Tensor Processing Units for Financial Monte Carlo. In Proceedings of the 2020 SIAM Conference on Parallel Processing for Scientific Computing, Seattle, WA, USA, 12–15 February 2020; pp. 12–23.
5. Gauen, K.; Rangan, R.; Mohan, A.; Lu, Y.H.; Liu, W.; Berg, A.C. Low-power image recognition challenge. In Proceedings of the 2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC), Chiba, Japan, 16–19 January 2017; pp. 99–104.
6. Park, H.; Kim, D.; Ahn, J.; Yoo, S. Zero and data reuse-aware fast convolution for deep neural networks on GPU. In *Proceedings of the Eleventh IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, Pittsburgh, PA, USA, 2–7 October 2016*; Association for Computing Machinery: New York, NY, USA, 2016; pp. 1–10.
7. Meng, W.; Cheng, Y.G.; Wu, J.; Yang, Z.; Zhu, Y.; Shang, S. GPU Acceleration of Hydraulic Transient Simulations of Large-Scale Water Supply Systems. *Appl. Sci.* **2018**, *9*, 91. [CrossRef]
8. Liu, Q.; Qin, Y.; Li, G. Fast Simulation of Large-Scale Floods Based on GPU Parallel Computing. *Water* **2018**, *10*, 589. [CrossRef]
9. Černý, D.; Dobeš, J. GPU Accelerated Nonlinear Electronic Circuits Solver for Transient Simulation of Systems with Large Number of Components. *Electronics* **2020**, *9*, 1819. [CrossRef]
10. Kim, S.; Cho, J.; Park, D. Accelerated DEVS Simulation Using Collaborative Computation on Multi-Cores and GPUs for Fire-Spreading IoT Sensing Applications. *Appl. Sci.* **2018**, *8*, 1466. [CrossRef]
11. Guo, J.; Liu, W.; Wang, W.; Yao, C.; Han, J.; Li, R.; Lu, Y.; Hu, S. AccUDNN: A GPU Memory Efficient Accelerator for Training Ultra-Deep Neural Networks. In Proceedings of the 2019 IEEE 37th International Conference on Computer Design (ICCD), Abu Dhabi, United Arab Emirates, 17–20 November 2019; pp. 65–72.
12. Lee, K.; Son, M. DeepSpotCloud: Leveraging Cross-Region GPU Spot Instances for Deep Learning. In Proceedings of the 2017 IEEE 10th International Conference on Cloud Computing (CLOUD), Honolulu, HI, USA, 25–30 June 2017; pp. 98–105.
13. Del Monte, B.; Prodan, R. A scalable GPU-enabled framework for training deep neural networks. In Proceedings of the 2016 2nd International Conference on Green High Performance Computing (ICGHPC), Nagercoil, India, 26–27 February 2016; pp. 1–8.
14. Lym, S.; Lee, D.; O'Connor, M.; Chatterjee, N.; Erez, M. DeLTA: GPU Performance Model for Deep Learning Applications with In-Depth Memory System Traffic Analysis. In Proceedings of the 2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Madison, WI, USA, 24–26 March 2019; pp. 293–303.
15. Joardar, B.K.; Nitthilan, K.J.; Janardhan, R.D.; Li, H.; Pande, P.P.; Chakrabarty, K. GRAMARCH: A GPU-ReRAM based Heterogeneous Architecture for Neural Image Segmentation. In Proceedings of the 2020 Design, Automation & Test in Europe Conference & Exhi-bition (DATE), Grenoble, France, 9–13 March 2020; pp. 228–233.
16. Joardar, B.K.; Doppa, J.R.; Pande, P.P.; Li, H.; Chakrabarty, K. AccuReD: High Accuracy Training of CNNs on ReRAM/GPU Heterogeneous 3D Architecture. *IEEE Trans. Comput. Des. Integr. Circuits Syst.* **2020**. [CrossRef]

17. Oyama, Y.; Nomura, A.; Sato, I.; Nishimura, H.; Tamatsu, Y.; Matsuoka, S. Predicting statistics of asynchronous SGD parameters for a large-scale distributed deep learning system on GPU supercomputers. In Proceedings of the 2016 IEEE International Conference on Big Data (Big Data), Washington, DC, USA, 5–8 December 2016; pp. 66–75.
18. Shriram, S.B.; Garg, A.; Kulkarni, P. Dynamic Memory Management for GPU-Based Training of Deep Neural Networks. In Proceedings of the 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS), Rio de Janeiro, Brazil, 20–24 May 2019; pp. 200–209.
19. Khomenko, V.; Shyshkov, O.; Radyvonenko, O.; Bokhan, K. Accelerating recurrent neural network training using sequence bucketing and multi-GPU data parallelization. In Proceedings of the 2016 IEEE First International Conference on Data Stream Mining & Processing (DSMP), Lviv, Ukraine, 23–27 August 2016; pp. 100–103.
20. Zhan, J.; Zhang, J. Pipe-Torch: Pipeline-Based Distributed Deep Learning in a GPU Cluster with Heterogeneous Net-working. In Proceedings of the 2019 Seventh Internetional Conference on Advanced Cloud and Big Data (CBD), Suzhou, China, 21–22 September 2019; pp. 55–60.
21. Kim, Y.; Choi, H.; Lee, J.; Kim, J.-S.; Jei, H.; Roh, H. Efficient Large-Scale Deep Learning Framework for Heterogeneous Multi-GPU Cluster. In Proceedings of the 2019 IEEE 4th International Workshops on Foundations and Applications of Self* Systems (FAS*W), Umeå, Sweden, 16–20 June 2019; pp. 176–181. [CrossRef]
22. Chen, C.-F.R.; Lee, G.G.C.; Xia, Y.; Lin, W.S.; Suzumura, T.; Lin, C.-Y. Efficient Multi-training Framework of Image Deep Learning on GPU Cluster. In Proceedings of the 2015 IEEE International Symposium on Multimedia (ISM), Miami, FL, USA, 14–16 December 2015; pp. 489–494. [CrossRef]
23. Chen, G.; He, S.; Meng, H.; Huang, K. PhoneBit: Efficient GPU-Accelerated Binary Neural Network Inference Engine for Mobile Phones. In Proceedings of the 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 9–13 March 2020; pp. 786–791.
24. Nurvitadhi, E.; Sheffield, D.; Sim, J.; Mishra, A.; Venkatesh, G.; Marr, D. Accelerating binarized neural networks: Comparison of FPGA, CPU, GPU, and ASIC. In Proceedings of the 2016 International Conference on Field-Programmable Technology (FPT), Xi'an, China, 7–9 December 2016; pp. 77–84.
25. Tu, Y.; Sadiq, S.; Tao, Y.; Shyu, M.L.; Chen, S.C. A Power Efficient Neural Network Implementation on Heterogeneous FPGA and GPU Devices. In Proceedings of the 2019 IEEE 20th International Conference Information Reuse and Integration for Data Science (IRI), Los Angeles, CA, USA, 30 July–1 August 2019; pp. 193–199.
26. Chen, Z.; Quan, W.; Wen, M.; Fang, J.; Yu, J.; Zhang, C.; Luo, L. Deep Learning Research and Development Platform: Characterizing and Scheduling with QoS Guarantees on GPU Clusters. *IEEE Trans. Parallel Distrib. Syst.* **2020**, *31*, 34–50. [CrossRef]
27. Dryden, N.; Maruyama, N.; Moon, T.; Benson, T.; Yoo, A.; Snir, M.; Van Essen, B. Aluminum: An Asynchronous, GPU-Aware Communication Library Optimized for Large-Scale Training of Deep Neural Networks on HPC Systems. In Proceedings of the 2018 IEEE/ACM Machine Learning in HPC Environments (MLHPC), Dallas, TX, USA, 11–16 November 2018; pp. 1–13.
28. Chu, C.-H.; Lu, X.; Awan, A.A.; Subramoni, H.; Hashmi, J.; Elton, B.; Panda, D.K. Efficient and Scalable Multi-Source Streaming Broadcast on GPU Clusters for Deep Learning. In Proceedings of the 2017 46th International Conference on Parallel Processing (ICPP), Bristol, UK, 14–17 August 2017; pp. 161–170.
29. Banerjee, D.S.; Hamidouche, K.; Panda, D.K. Re-Designing CNTK Deep Learning Framework on Modern GPU Enabled Clusters. In Proceedings of the 2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Luxembourg City, Luxembourg, 12–15 December 2016; pp. 144–151.
30. Ogayar-Anguita, C.J.; Rueda-Ruiz, A.J.; Segura-Sánchez, R.J.; Díaz-Medina, M.; García-Fernández, Á.L. A GPU-Based Framework for Generating Implicit Datasets of Voxelized Polygonal Models for the Training of 3D Convolutional Neural Networks. *IEEE Access* **2020**, *8*, 12675–12687. [CrossRef]
31. Li, S.; Dou, Y.; Lv, Q.; Wang, Q.; Niu, X.; Yang, K. Optimized GPU acceleration algorithm of convolutional neural net-works for target detection. In Proceedings of the 2016 IEEE 18th International Conference High Performance Computing and Communications, Sydney, NSW, Australia, 12–14 December 2016; pp. 224–230.
32. Chen, G.; Meng, H.; Liang, Y.; Huang, K. GPU-Accelerated Real-Time Stereo Estimation with Binary Neural Network. *IEEE Trans. Parallel Distrib. Syst.* **2020**, *31*, 2896–2907. [CrossRef]
33. Gong, T.; Fan, T.; Guo, J.; Cai, Z. Gpu-based parallel optimization and embedded system application of immune convolutional neural network. In Proceedings of the 2015 International Workshop Artificial Immune Systems (AIS), Taormina, Italy, 17–18 July 2015; pp. 1–8.
34. Saypadith, S.; Aramvith, S. Real-Time Multiple Face Recognition using Deep Learning on Embedded GPU System. In Proceedings of the 2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), Honolulu, HI, USA, 12–15 November 2018; pp. 1318–1324.
35. Xu, J.; Wang, B.; Li, J.; Hu, C.; Pan, J. Deep learning application based on embedded GPU. In Proceedings of the 2017 First International Conference on Electronics Instrumentation & Information Systems (EIIS), Harbin, China, 3–5 June 2017; pp. 1–4.
36. Appuhamy, E.J.G.S.; Madhusanka, B. Development of a GPU-Based Human Emotion Recognition Robot Eye for Service Robot by Using Convolutional Neural Network. In Proceedings of the 2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS), Singapore, 6–8 June 2018; pp. 433–438.

37. Kain, E.; Wildenstein, D.; Pineda, A.C. Embedded GPU Cluster Computing Framework for Inference of Convolutional Neural Networks. In Proceedings of the 2019 IEEE High Performance Extreme Computing Conference (HPEC), Waltham, MA, USA, 24–26 September 2019; pp. 1–7.
38. Campos, V.; Sastre, F.; Yagues, M.; Torres, J.; Giró-I-Nieto, X. Scaling a Convolutional Neural Network for Classification of Adjective Noun Pairs with TensorFlow on GPU Clusters. In Proceedings of the 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), Madrid, Spain, 14–17 May 2017; pp. 677–682.
39. Dong, X.; Yang, Y. Searching for a Robust Neural Architecture in Four GPU Hours. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 1761–1770.
40. dos Santos, F.F.; Draghetti, L.; Weigel, L.; Carro, L.; Navaux, P.; Rech, P. Evaluation and mitigation of soft-errors in neu-ral network-based object detection in three gpu architectures. In Proceedings of the 2017 47th Annual IEEE/IFIP International Conference Dependable Systems and Networks Workshops (DSN-W), Denver, CO, USA, 26–29 June 2017; pp. 169–176.
41. Zhang, C.; Weingartner, S.; Moeller, S.; Ugurbil, K.; Akcakaya, M. Fast GPU Implementation of a Scan-Specific Deep Learning Reconstruction for Accelerated Magnetic Resonance Imaging. In Proceedings of the 2018 IEEE International Conference on Electro/Information Technology (EIT), Rochester, MI, USA, 3–5 May 2018; Volume 2018, pp. 399–403.
42. 38. Ziabari, A.; Ye, D.H.; Srivastava, S.; Sauer, K.D.; Thibault, J.-B.; Bouman, C.A. 2.5 D deep learning for CT image reconstruction using a multi-GPU implementation. In Proceedings of the 2018 52nd Asilomar Conference Signals, Systems, and Computers, Pacific Grove, CA, USA, 28–31 October 2018; pp. 2044–2049.
43. Bijoy, M.B.; Shilimkar, V.; Jayaraj, P.B. Detecting Cervix Type Using Deep learning and GPU. In Proceedings of the 2018 IEEE Region 10 Humanitarian Technology Conference (R10-HTC), Malambe, Sri Lanka, 6–8 December 2018; pp. 1–6.
44. Moharir, M.; Sachin, M.U.; Nagaraj, R.; Samiksha, M.; Rao, S. Identification of asphyxia in newborns using gpu for deep learning. In Proceedings of the 2017 2nd International Conference for Convergence in Technology (I2CT), Mumbai, India, 7–9 April 2017; pp. 236–239.
45. Guerreiro, J.; Ilić, A.; Roma, N.; Tomás, P. GPU Static Modeling Using PTX and Deep Structured Learning. *IEEE Access* **2019**, *7*, 159150–159161. [CrossRef]
46. Jhu, C.-F.; Liu, P.; Wu, J.-J. Data Pinning and Back Propagation Memory Optimization for Deep Learning on GPU. In Proceedings of the 2018 Sixth International Symposium on Computing and Networking (CANDAR), Takayama, Japan, 23–27 November 2018; pp. 19–28. [CrossRef]
47. Cui, H.; Zhang, H.; Ganger, G.R.; Gibbons, P.B.; Xing, E.P. Geeps: Scalable deep learning on distributed gpus with a gpu specialized parameter server. In Proceedings of the Eleventh European Conference on Computer Systems, London, UK, 18–21 April 2016; pp. 4:1–4:16.
48. Fukushi, M.; Kanbara, Y. A GPU Implementation Method of Deep Neural Networks Based on Data Swapping. In Proceedings of the 2019 IEEE International Conference on Consumer Electronics—Taiwan (ICCE-TW), Yilan, Taiwan, 20–22 May 2019; pp. 1–2.
49. Kim, Y.; Lee, J.; Kim, J.-S.; Jei, H.; Roh, H. Efficient Multi-GPU Memory Management for Deep Learning Acceleration. In Proceedings of the 2018 IEEE 3rd International Workshops on Foundations and Applications of Self* Systems (FAS*W), Trento, Italy, 3–7 September 2018; pp. 37–43. [CrossRef]
50. Ito, Y.; Matsumiya, R.; Endo, T. ooc_cuDNN: Accommodating convolutional neural networks over GPU memory capacity. In Proceedings of the 2017 IEEE International Conference on Big Data (Big Data), Boston, MA, USA, 11–14 December 2017; pp. 183–192.
51. Rizvi, S.T.H.; Cabodi, G.; Francini, G. GPU-only unified ConvMM layer for neural classifiers. In Proceedings of the 2017 4th International Conference on Control, Decision and Information Technologies (CoDIT), Barcelona, Spain, 5–7 April 2017; pp. 0539–0543.
52. Malik, A.; Lu, M.; Wang, N.; Lin, Y.; Yoo, S. Detailed Performance Analysis of Distributed Tensorflow on a GPU Cluster using Deep Learning Algorithms. In Proceedings of the 2018 New York Scientific Data Summit (NYSDS), Upton, NY, USA, 6–8 August 2018; pp. 1–8. [CrossRef]
53. Thanasekhar, B.; Gomathy, N.; Shwetha, B.; Sumithra, A. Fault and Delay Tolerance in Deep Learning Framework under GPU. In Proceedings of the 2019 11th International Conference on Advanced Computing (ICoAC), Chennai, India, 18–20 December 2019; pp. 139–146.
54. Raniah, Z.; Shaziya, H. GPU-based empirical evaluation of activation functions in convolutional neural networks. In Proceedings of the 2018 2nd International Conference Inventive Systems and Control (ICISC), Coimbatore, India, 19–20 January 2018; pp. 769–773.
55. Lin, Y.; Jiang, Z.; Gu, J.; Li, W.; Dhar, S.; Ren, H.; Khailany, B.; Pan, D.Z. DREAMPlace: Deep Learning Toolkit-Enabled GPU Acceleration for Modern VLSI Placement. *IEEE Trans. Comput. Des. Integr. Circuits Syst.* **2020**. [CrossRef]
56. Chen, C.; Li, K.; Ouyang, A.; Tang, Z.; Li, K. GPU-Accelerated Parallel Hierarchical Extreme Learning Machine on Flink for Big Data. *IEEE Trans. Syst. Man, Cybern. Syst.* **2017**, *47*, 2740–2753. [CrossRef]
57. Li, P.; Luo, Y.; Zhang, N.; Cao, Y. Heterospark: A heterogeneous cpu/gpu spark platform for machine learning algorithms. In Proceedings of the 2015 IEEE International Conference on Networking, Architecture and Storage (NAS), Boston, MA, USA, 6–7 August 2015; pp. 347–348.
58. Sun, T.; Wang, H.; Shen, Y.; Wu, J. Accelerating support vector machine learning with GPU-based mapreduce. In Proceedings of the 2015 IEEE International Conference Systems, Man, and Cybernetics, Kowloon, China, 9–12 October 2015; pp. 876–881.

59. She, X.; Long, Y.; Mukhopadhyay, S. Fast and Low-Precision Learning in GPU-Accelerated Spiking Neural Network. In Proceedings of the 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), Florence, Italy, 25–29 March 2019; pp. 450–455.
60. Naveros, F.; Luque, N.R.; Garrido, J.A.; Carrillo, R.R.; Anguita, M.; Ros, E. A Spiking Neural Simulator Integrating Event-Driven and Time-Driven Computation Schemes Using Parallel CPU-GPU Co-Processing: A Case Study. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *26*, 1567–1574. [CrossRef]
61. Yazdanbakhsh, A.; Park, J.; Sharma, H.; Lotfi-Kamran, P.; Esmaeilzadeh, H. Neural acceleration for gpu throughput proces-sors. In Proceedings of the 48th International Symposium Microarchitecture, Columbus, OH, USA, 12–16 October 2019; pp. 482–493.
62. Xu, H.; Emani, M.; Lin, P.-H.; Hu, L.; Liao, C. Machine Learning Guided Optimal Use of GPU Unified Memory. In Proceedings of the 2019 IEEE/ACM Workshop on Memory Centric High Performance Computing (MCHPC), Denver, CO, USA, 18 November 2019; pp. 64–70.
63. Vooturi, D.T.; Kothapalli, K. Efficient Sparse Neural Networks Using Regularized Multi Block Sparsity Pattern on a GPU. In Proceedings of the 2019 IEEE 26th International Conference on High Performance Computing, Data, and Analytics (HiPC), Hyderabad, India, 17–20 December 2019; pp. 215–224.
64. Dogaru, R.; Dogaru, I. Optimization of GPU and CPU acceleration for neural networks layers implemented in python. In Proceedings of the 2017 5th International Symposium on Electrical and Electronics Engineering (ISEEE), Galati, Romania, 20–22 October 2017; pp. 1–6.
65. Mei, S.; He, M.; Shen, Z. Optimizing Hopfield Neural Network for Spectral Mixture Unmixing on GPU Platform. *IEEE Geosci. Remote Sens. Lett.* **2013**, *11*, 818–822. [CrossRef]
66. Huang, Y.; Guo, B.; Shen, Y. GPU Energy Consumption Optimization with a Global-Based Neural Network Method. *IEEE Access* **2019**, *7*, 64303–64314. [CrossRef]
67. Li, J.; Guo, B.; Shen, Y.; Li, D.; Wang, J.; Huang, Y.; Li, Q. GPU-memory coordinated energy saving approach based on extreme learning machine. In Proceedings of the 2015 IEEE 17th International Conference on High Performance Computing and Communications, New York, NY, USA, 24–26 August 2015; pp. 827–830.
68. O'Leary, G.; Taras, I.; Stuart, D.M.; Koerner, J.; Groppe, D.M.; Valiante, T.A.; Genov, R. GPU—Accelerated Parameter Selection for Neural Connectivity Analysis Devices. In Proceedings of the 2018 IEEE Biomedical Circuits and Systems Conference (BioCAS), Cleveland, OH, USA, 17–19 October 2018; pp. 1–4.
69. Mujahid, T.; Rahman, A.U.; Khan, M.M. GPU-accelerated multivariate empirical mode decomposition for massive neural data processing. *IEEE Access* **2017**, *5*, 8691–8701. [CrossRef]
70. Neofytou, A.; Chatzikonstantis, G.; Magkanaris, I.; Smaragdos, G.; Strydis, C.; Soudris, D. GPU Implementation of Neural-Network Simulations Based on Adaptive-Exponential Models. In Proceedings of the 2019 IEEE 19th International Conference on Bioinformatics and Bioengineering (BIBE), Athens, Greece, 28–30 October 2019; pp. 339–343.
71. Hacker, C.; Aizenberg, I.; Wilson, J. GPU simulator of multilayer neural network based on multi-valued neurons. In Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, Canada, 24–29 July 2016; pp. 4125–4132.
72. Phaudphut, C.; So-In, C.; Phusomsai, W. A parallel probabilistic neural network ECG recognition architecture over GPU platforms. In Proceedings of the 2016 13th International Joint Conference on Computer Science and Software Engineering (JCSSE), Khon Kaen, Thailand, 13–15 July 2016; pp. 1–7.
73. Mayerich, D.; Kwon, J.; Panchal, A.; Keyser, J.; Choe, Y. Fast cell detection in high-throughput imagery using GPU-accelerated machine learning. In Proceedings of the 2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro, Chicago, IL, USA, 30 March–2 April 2011; pp. 719–723. [CrossRef]
74. Liu, Z.; Murakami, T.; Kawamura, S.; Yoshida, H. Parallel Implementation of Chaos Neural Networks for an Embedded GPU. In Proceedings of the 2019 IEEE 10th International Conference on Awareness Science and Technology (iCAST), Morioka, Japan, 23–25 October 2019; pp. 1–6.
75. Van, N.T.T.; Thinh, T.N. Accelerating Anomaly-Based IDS Using Neural Network on GPU. In Proceedings of the 2015 International Conference on Advanced Computing and Applications (ACOMP), Ho Chi Minh City, Vietnam, 23–25 November 2015; pp. 67–74.
76. Hamer, M.; Widmer, L.; D'andrea, R. Fast generation of collision-free trajectories for robot swarms using GPU acceleration. *IEEE Access* **2018**, *7*, 6679–6690. [CrossRef]
77. Moreira, R.D.S.; Ebecken, N.F.F.; Affiliation, N.F.F.E. GWVT: A GPU maritime vessel tracker based on the wisard weightless neural network. *Nav. Eng. J.* **2017**, *129*, 109–116. [CrossRef]
78. Gavali, P.; Banu, J. Bird Species Identification using Deep Learning on GPU platform. In Proceedings of the 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), Vellore, India, 24–25 February 2020; pp. 1–6.