6 june–11 june Reading Paper and Summary of Paper

Deep neural networks have been successful in many areas, but deploying them to end-user applications remains challenging. Edge devices, such as mobile phones, wearables, IoT, embedded and autonomous systems, and intelligent sensors, have limited memory, computing resources, and power-handling capability. To reduce cloud transmission cost, optimization techniques have been developed to handle DL deployment efficiently on the edge.

Machine learning algorithms extract and learn characteristic features from data for automatic decision-making. Deep learning or Deep neural network is one of the subfields of ML. The three main layers of an NN are the input layer, hidden layers, and output layer. Training is the refinement of the output through an iterative learning process that includes both forward and backward propagation. Weights and biases are the parameters learned and updated through the training process. Backpropagation is an efficient way to update the weights and biases.

The deployment of DL models for inference can be of two main types: 1-cloud-based processing 2- Inference on edge Cloud-based processing requires data transmission from the acquisition site to the cloud, while edge devices have limited memory, computation, and power budget.

Edge inference has many advantages over cloud-based computing, such as faster and real-time use cases, better security, bandwidth efficiency, scalability, and reliability. AI applications are migrating from the cloud to embedded edge devices, such as virtual assistants, smartphones, wearable devices, remote monitoring of patients, thermal screening, and disease prediction etc. Edge devices consider the best tradeoffs between performance, power, energy efficiency, latency, cost, and size.

Edge computing is a new way of doing computing where processing happens closer to where the data is created, making it easier to use DL in real life situations. There are surveys available on topics related to DL, such as theories, designs, algorithms, and uses. Discussions on compression performance evaluation are limited, software frameworks are absent, and there is a lack of reviewing hardware acceleration platforms. Edge training is slower compared to training on the cloud. Edge AI- Edge AI combines edge computing and AI to leverage data generated at the edge for various applications. It involves four key elements: edge caching, edge training, edge inference, and edge offloading. Edge caching involves collecting and storing data from edge devices to reduce computational complexity and inference time. Edge training enables devices to learn patterns from the cached data, either independently or collaboratively with cloud support. Edge inference performs real-time computations on trained ML models, and edge offloading distributes computing tasks to connected centers or cloud servers when edge hardware capacity is insufficient. ML algorithms such as decision trees, SVMs, K-nearest neighbors (KNNs) and linear regression (LR) are more appropriate than DL in many real world applications. ProtoNN is an implementation of KNN on edge devices demonstrating real-time inference and minimal storage.

Deep Learning - Deep learning models are like the brain's neurons and can learn things automatically. To choose a model, we consider how complicated it is, how well it performs, how easy it is to maintain, and how many resources it needs. There are two main types of machine learning algorithms: supervised and unsupervised learning. Supervised learning uses a loss function to measure how different our predictions are from the actual correct answers. Neural networks are used in both supervised and unsupervised learning, and semi-supervised learning combines labeled and unlabeled data. Reinforcement learning is another technique where the model learns by trying different things and getting feedback on what works and what doesn't. There are different types of deep learning architectures, such as feed-forward neural networks and recurrent neural networks. To improve accuracy, we often make the deep learning models more complex by adding more layers and using lighter computations. Compression techniques can help make the models smaller and more efficient by reducing their complexity and memory usage.

Edge Compatible Deep Learning Model Design in Two Ways- 1- Neural Architecture Search- NAS helps us find optimal DNN architectures and components that work well on resource constrained edge hardware devices, allowing us to strike a balance between compact computations, reduced parameters, and acceptable accuracy. Neural Architecture Search is an algorithm used to find the best DNN architectures and components for edge hardware devices. It has three main parts: the search space, the search strategy, and performance estimation. The search space is defined by setting constraints on the possible network architectures, operations, layers, and repeated patterns. The search strategy considers candidate architectures based on the performance of previously explored solutions that worked well. Performance estimation is a way to evaluate the performance of different architectures without training them from scratch.

2- Compact Network Design-

In optimized convolution for convolutional neural networks, techniques like input size reduction and pointwise convolution are used to decrease the number of computations. MobileNet and MobileNet-v2 employ these optimizations, along with additional techniques like residual connections and intermediate data encoding. SqueezeNet reduces computations by decreasing the number of input filters, using 1x1 filters instead of 3x3 filters, and downsampling with global average pooling. ShuffleNet reduces operations by applying convolutions in parallel to different parts of the inputs. For recurrent neural networks, improvements are made at the unit and network levels. LSTM and GRU units are used to address the vanishing or exploding gradient problem in traditional RNNs. LSTM can be extended with an extra time gate or using phased LSTM. Factorizing the weight matrix of LSTM and concatenating the results can accelerate convergence and reduce parameters. These optimizations, including LSTM, GRU, phased LSTM, and matrix factorization, enhance the performance and efficiency of RNNs.

Deep Learning Model Compression Techniques-

A- Pruning- Pruning is a compression technique that removes less important weights or filters from a trained model. There are three main approaches to pruning: magnitude-based, regularization-based, and energy-aware pruning. Magnitude-based pruning removes weights below a threshold, while regularization-based methods use a regularization term to achieve sparsity. Energy-aware pruning estimates energy consumption to determine which weights to prune. Each approach has advantages and limitations.

B- Quantization - Quantization is a technique in deep learning that maps model parameters and activations to low-precision levels, reducing memory usage and improving inference time. It can be linear or nonlinear. Quantization provides benefits like memory saving, reduced complexity, and improved energy efficiency. Variable and mixed-precision quantization are used, and Binarization is an extreme type of quantization that uses a 1-bit representation for weights and activations.

Other Compression Techniques- Joint compression and weight sharing techniques are applied together with pruning and quantization to reduce the storage requirements of deep learning models. Weight sharing reduces the number of unique weights by using k-means clustering and Huffman coding, while vector quantization can also be employed to reduce the number of parameters. Multiply-and-accumulate optimization is another strategy for improving inference performance by optimizing multiplication operations. Computational optimization techniques include approximate computing, low-rank approximation, and stochastic computing. Knowledge distillation is a compression method that transfers knowledge from a large model to a lightweight model. Adaptive optimization techniques allow tuning during inference runtime based on the input, while Big Little classification uses two models of different sizes for classification tasks. Early existing techniques enable acceptable inference performance using only a portion of the complete model, reducing latency.

Metrics for Performance Evaluation- Model Size, Accuracy and Robustness, Area and Cost should be less, Power Consumption should be less, Energy Efficiency should be high, latency should be less, throughput should be high, Memory optimization is important for high energy efficiency and low power consumption, Test setup is an important aspect of evaluating the DL inference for different applications.

Software tools and techniques for edge Inference and codesign- Hardware acceleration in deep learning focuses on parallel computing, while software optimization focuses on pipelining, resource management, and efficient compiler design. Hardware specific development platforms like Nvidia Jetson TX2 and Intel Edison kit are available for edge and IoT devices. RSTensorFlow is an extension of TensorFlow designed for proper utilization of heterogeneous computing resources. Caffe is focused on DL model building for edge devices. Graph compilers generate optimized instructions for specific hardware architectures. In the field of deep learning, several technologies and frameworks have been developed to optimize and accelerate the inference process on edge devices. One approach is compression-compiler codesign, which combines DL model compression and compiler optimization to achieve energy efficiency and faster inference. Algorithms and hardware are co-designed to create application-specific DNN accelerators. Frameworks such as CNNdroid, CUDA, and Cappuccino enable parallel computing on CPUs, GPUs, and DSPs, leading to improved speed and energy reduction. Efficient memory access techniques, like data reuse and dataflow schemes, minimize the number of accesses to power-consuming DRAM. Intel OpenVINO Toolkit optimizes pretrained NN models for Intel hardware platforms, while MATLAB Deep Learning HDL Toolbox supports inference on FPGAs and SoCs. X-CUBE-AI converts pretrained DL models for inference on microcontroller units, and XILINX DNNDK accelerates DL inference in programmable logic. NVIDIA TensorRT and CEVA CDNN offer optimized inference solutions for embedded and automotive devices. Qualcomm Neural Processing SDK enables DL model compatibility with Snapdragon mobile platforms, Cadence Stratus HLS allows realization and evaluation of DL inference in RTL. Software frameworks like PyTorch, TensorFlow, DeepSpeed, and Horovod support large-scale distributed training, while cloud platforms provide convenient DL model training and performance evaluation. These advancements contribute to efficient and optimized deep learning inference on edge devices.

Edge H/W Platforms for deep learning Inference- GPUs and TPUs are popular in the cloud, but their high energy cost makes them unsuitable for edge devices. Customized neural accelerators are needed for edge inference, and the choice of hardware depends on factors like latency, area, energy, cost, and flexibility. Several Hardware devices are used for efficient and accelerated execution of EI such as General Purpose CPU, GPU, System-on-Chip and Application-Specific Integrated Circuit, Field-Programmable Gate Array and Reconfigurable Architectures are two types of hardware accelerators for DL models. FPGA is a fine-grained reconfigurable architecture that can achieve low latency, high throughput, and energy efficiency. Vision Processing Units are customized microprocessor chips for EI for machine vision applications. NeuFlow is a general-purpose vision processor for real-time object detection, recognition, and localization. Microcontroller Units (MCUs) are devices that can bring intelligence to billions of products, but have limitations such as less memory, slow processing, lack of parallelism, low clock frequency, and general-purpose processors. Emerging Memory-Based Neural Accelerators (BNNs) split matrix-vector multiplication into numerous MAC operations to perform the operations cyclewise.

Distributed edge training (DL) is a technique of learning from data in different edge devices and servers. It is performed in proximity to the data collection for cost-effective communication in data transfer and ensuring user privacy. Federated learning is an emerging technique of edge training using data from multiple nodes, and DL model splitting is another edge training approach that transfers processed data instead of raw data to the cloud server. Parallelism in data and model can facilitate computation load balancing.

Challenges and Future Scope- Research directions in Edge Inference should include adapting to data heterogeneity through techniques like data augmentation and representation learning. Developing tools for automatic mapping of trained DL models to hardware-compatible versions is essential to streamline the deployment process. Establishing benchmark standards with diverse datasets and DL models is necessary for evaluating and comparing EI performance accurately. Automatic compression techniques and algorithm-hardware co-design can improve efficiency by reducing the manual intervention required. Neural Architecture Search (NAS) algorithms need to be enhanced to generate efficient and compact network architectures tailored to specific edge hardware. Adding flexibility to ASICs and SoCs can enhance performance efficiency and cost-effectiveness. Specialized hardware architectures should be developed to handle sparsity in compressed DNN models. Research should expand beyond feedforward and convolutional neural networks to include other DL techniques such as sequence analysis for voice recognition and natural language processing. Edge training with retraining capability and transfer learning can support continuous learning from new data. Efficient communication strategies need to be explored to address communication challenges in data collection, cloud transfer, edge deployment, and distributed training/inference. Incorporating explainable AI into edge devices remains an unexplored research direction for ensuring transparency and explainability in EI.

Deep neural networks have been successful in many areas, but deploying them to end-user applications remains challenging. Edge devices, such as mobile phones, CPSs, wearables, IoT, embedded and autonomous systems, and intelligent sensors, have limited memory, computing resources, and power-handling capability. To reduce cloud transmission cost, optimization techniques have been developed to handle DL deployment efficiently on the edge.

Mainly, four research directions have been pursued for efficient DL inference on edge devices: 1) novel DL architecture and algorithm design 2) optimization of existing DL methods 3) development of algorithm–hardware codesign 4) efficient accelerator design for DL deployment

Machine learning algorithms extract and learn characteristic features from data for automatic decision-making. Deep learning or Deep neural network is one of the subfields of ML. The three main layers of an NN are the input layer, hidden layers, and output layer. Training is the refinement of the output through an iterative learning process that includes both forward and backward propagation. Weights and biases are the parameters learned and updated through the training process. Backpropagation is an efficient way to update the weights and biases.

The deployment of DL models for inference can be of two main types: 1-cloud-based processing 2- Inference on edge Cloud-based processing requires data transmission from the acquisition site to the cloud, while edge devices have limited memory, computation, and power budget.

Edge inference has many advantages over cloud-based computing, such as faster and real-time use cases, better security, bandwidth efficiency, scalability, and reliability. AI applications are migrating from the cloud to embedded edge devices, such as virtual assistants, smartphones, wearable devices, remote monitoring of patients, thermal screening, and disease prediction. Edge devices contemplate the best tradeoffs between performance, power, energy efficiency, latency, cost, and size.

Researchers are studying Deep Learning Edge inference due to its ability to perform tasks as well as humans in smart applications. Edge computing is a new way of doing computing where processing happens closer to where the data is created, making it easier to use DL in real-life situations. There are surveys available on topics related to DL, such as theories, designs, algorithms, and uses. Discussions on compression performance evaluation are limited, software frameworks are absent, and there is a lack of reviewing hardware acceleration platforms.

Edge training is slower compared to training on the cloud.

Edge AI- Edge AI combines edge computing and AI to leverage data generated at the edge for various applications. It involves four key elements: edge caching, edge training, edge inference, and edge offloading. Edge caching involves collecting and storing data from edge devices to reduce computational complexity and inference time. Edge training enables devices to learn patterns from the cached data, either independently or collaboratively with cloud support. Edge inference performs real-time computations on trained ML models, and edge offloading distributes computing tasks to connected centers or cloud servers when edge hardware capacity is insufficient.

ML algorithms such as RFs, decision trees, SVMs, K-nearest neighbors (KNNs) and linear regression (LR) are more appropriate than DL in many real-world applications. ProtoNN is an implementation of KNN on edge devices demonstrating real-time inference and minimal storage.

Deep Learning - Deep learning models are like the brain's neurons and can learn things automatically. To choose a model, we consider how complicated it is, how well it performs, how easy it is to maintain, and how many resources it needs. There are two main types of machine learning algorithms: supervised and unsupervised learning. Supervised learning uses a "loss function" to measure how different our predictions are from the actual correct answers. Neural networks are used in both supervised and unsupervised learning, and semi-supervised learning combines labeled and unlabeled data. Reinforcement learning is another technique where the model learns by trying different things and getting feedback on what works and what doesn't. There are different types of deep learning architectures, such as feed-forward neural networks and recurrent neural networks. To improve accuracy, we often make the deep learning models more complex by adding more layers and using lighter computations. Compression techniques can help make the models smaller and more efficient by reducing their complexity and memory usage.

Edge Compatible Deep Learning Model Design in Two Ways- 1- Neural Architecture Search- NAS helps us find optimal DNN architectures and components that work well on resource-constrained edge hardware devices, allowing us to strike a balance between compact computations, reduced parameters, and acceptable accuracy. Neural Architecture Search (NAS) is an algorithm used to find the best DNN architectures and components for edge hardware devices. It has three main parts: the search space, the search strategy, and performance estimation. The search space is defined by setting constraints on the possible network architectures, operations, layers, and repeated patterns. The search strategy considers candidate architectures based on the performance of previously explored solutions that worked well. Performance estimation is a way to evaluate the performance of different architectures without training them from scratch.

2- Compact Network Design-

In optimized convolution for convolutional neural networks (CNNs), techniques like input size reduction and pointwise convolution are used to decrease the number of computations. MobileNet and MobileNet-v2 employ these optimizations, along with additional techniques like residual connections and intermediate data encoding. SqueezeNet reduces computations by decreasing the number of input filters, using 1x1 filters instead of 3x3 filters, and downsampling with global average pooling. ShuffleNet reduces operations by applying convolutions in parallel to different parts of the inputs. For recurrent neural networks (RNNs), improvements are made at the unit and network levels. LSTM and GRU units are used to address the vanishing or exploding gradient problem in traditional RNNs. LSTM can be extended with an extra time gate or using phased LSTM. Factorizing the weight matrix of LSTM and concatenating the results can accelerate convergence and reduce parameters. These optimizations, including LSTM, GRU, phased LSTM, and matrix factorization, enhance the performance and efficiency of RNNs.

Deep Learning Model Compression Techniques-

A- Pruning- Pruning is a compression technique that removes less important weights or filters from a trained model. There are three main approaches to pruning: magnitude-based, regularization-based, and energy-aware pruning. Magnitude-based pruning removes weights below a threshold, while regularization-based methods use a regularization term to achieve sparsity. Energy-aware pruning estimates energy consumption to determine which weights to prune. Each approach has advantages and limitations.

B- Quantization - Quantization is a technique in deep learning that maps model parameters and activations to low-precision levels, reducing memory usage and improving inference time. It can be linear or nonlinear. Quantization provides benefits like memory saving, reduced complexity, and improved energy efficiency. Variable and mixed-precision quantization are used, and Binarization is an extreme type of quantization that uses a 1-bit representation for weights and activations.

Other Compression Techniques-

Joint compression and weight sharing techniques are applied together with pruning and quantization to reduce the storage requirements of deep learning models. Weight sharing reduces the number of unique weights by using k-means clustering and Huffman coding, while vector quantization can also be employed to reduce the number of parameters. Multiply-and-accumulate optimization is another strategy for improving inference performance by optimizing multiplication operations. Computational optimization techniques include approximate computing, low-rank approximation, and stochastic computing. Knowledge distillation is a compression method that transfers knowledge from a large model to a lightweight model. Adaptive optimization techniques allow tuning during inference runtime based on the input, while Big-Little classification uses two models of different sizes for classification tasks. Early exiting techniques enable acceptable inference performance using only a portion of the complete model, reducing latency.

Metrics for Performance Evaluation- Model Size, Accuracy and Robustness, Area and Cost should be less , Power Consumption should be less, Energy Efficiency should be high, latency should be less, throughput should be high, Memory optimization is important for high energy efficiency and low power consumption, Test setup is an important aspect of evaluating the DL inference for different applications.

Software tools and techniques for edge Inference and codesign- Hardware acceleration in deep learning focuses on parallel computing, while software optimization focuses on pipelining, resource management, and efficient compiler design. Hardware-specific development platforms like Nvidia Jetson TX2 and Intel Edison kit are available for edge and IoT devices. RSTensorFlow is an extension of TensorFlow designed for proper utilization of heterogeneous computing resources. Caffe is focused on DL model building for edge devices. Graph compilers generate optimized instructions for specific hardware architectures. In the field of deep learning, several technologies and frameworks have been developed to optimize and accelerate the inference process on edge devices. One approach is compression-compiler codesign, which combines DL model compression and compiler optimization to achieve energy efficiency and faster inference. Algorithms and hardware are co-designed to create application-specific DNN accelerators. Frameworks such as CNNdroid, CUDA, and Cappuccino enable parallel computing on CPUs, GPUs, and DSPs, leading to improved speed and energy reduction. Efficient memory access techniques, like data reuse and dataflow schemes, minimize the number of accesses to power-consuming DRAM. Intel OpenVINO Toolkit optimizes pretrained NN models for Intel hardware platforms, while MATLAB Deep Learning HDL Toolbox supports inference on FPGAs and SoCs. X-CUBE-AI converts pretrained DL models for inference on microcontroller units, and XILINX DNNDK accelerates DL inference in programmable logic. NVIDIA TensorRT and CEVA CDNN offer optimized inference solutions for embedded and automotive devices. Qualcomm Neural Processing SDK enables DL model compatibility with Snapdragon mobile platforms. Cadence Stratus HLS allows realization and evaluation of DL inference in RTL. Software frameworks like PyTorch, TensorFlow, DeepSpeed, and Horovod support large-scale distributed training, while cloud platforms provide convenient DL model training and performance evaluation. These advancements contribute to efficient and optimized deep learning inference on edge devices.

Edge H/W Platforms for deep learning Inference- GPUs and TPUs are popular in the cloud, but their high energy cost makes them unsuitable for edge devices. Customized neural accelerators are needed for edge inference, and the choice of hardware depends on factors like latency, area, energy, cost, and flexibility. Several Hardware devices are used for efficient and accelerated execution of EI such as General Purpose CPU, GPU, System-on-Chip and Application-Specific Integrated Circuit, Field-Programmable Gate Array and Reconfigurable Architectures are two types of hardware accelerators for DL models. FPGA is a fine-grained reconfigurable architecture that can achieve low latency, high throughput, and energy efficiency. Vision Processing Units are customized microprocessor chips for EI for machine vision applications. NeuFlow is a general-purpose vision processor for real-time object detection, recognition, and localization. Microcontroller Units (MCUs) are devices that can bring intelligence to billions of products, but have limitations such as less memory, slow processing, lack of parallelism, low clock frequency, and general-purpose processors. Emerging Memory-Based Neural Accelerators (BNNs) split matrix-vector multiplication into numerous MAC operations to perform the operations cyclewise.

Challenges and Future Scope-

Research directions in Edge Inference (EI) should be to include adapting to data heterogeneity through techniques like data augmentation and representation learning. Developing tools for automatic mapping of trained DL models to hardware-compatible versions is essential to streamline the deployment process. Establishing benchmark standards with diverse datasets and DL models is necessary for evaluating and comparing EI performance accurately. Automatic compression techniques and algorithm-hardware co-design can improve efficiency by reducing the manual intervention required. Neural Architecture Search (NAS) algorithms need to be enhanced to generate efficient and compact network architectures tailored to specific edge hardware. Adding flexibility to ASICs and SoCs can enhance performance efficiency and cost-effectiveness. Specialized hardware architectures should be developed to handle sparsity in compressed DNN models. Research should expand beyond feedforward and convolutional neural networks to include other DL techniques such as sequence analysis for voice recognition and natural language processing. Edge training with retraining capability and transfer learning can support continuous learning from new data. Efficient communication strategies need to be explored to address communication challenges in data collection, cloud transfer, edge deployment, and distributed training/inference. Incorporating explainable AI (XAI) into edge devices remains an unexplored research direction for ensuring transparency and explainability in EI.