# SURVEY ON DEEP LEARNING MODELS ON EDGE DEVICES FOR IOT APPLICATIONS

Mr. G .MUTHUPANDI,
Assistant Professor/CSE,
Ramco Institute of Technology,
Rajapalayam, Tamil Nadu,

Mr. S. MANICKAM, PhD Scholar,
Assistant Professor/CSE,
Saveetha Engineering College,
Chennai, Tamil Nadu,

## ABSTRACT

In recent years Deep learning algorithms are used in many applications such as vision recognition, speech recognition, bioinformatics and so on. The Internet of Things is the next booming technology for real-time applications, Augmented reality, Self-driving cars, Environmental monitoring, Agriculture, health care Industrial applications and so on. Implement Deep learning with high accuracy comes under high energy and computing capabilities which are offered by cloud computing, but it has some drawbacks when comes to real-time applications such as latency, scalability, and privacy. IoT devices run on limited capacity and computing power but the recent advancements in hardware technologies to make IoT devices more powerful and capable to run Deep learning algorithms on them. The Deep learning algorithm running on Edge devices will reduce the latency delay and make the applications quick responsive. TinyML is the new technology which enables to deploy of deep learning models on Embedded devices and low-powered microcontrollers. In this paper, we discussed what are the various ways to run a Deep-learning algorithm on the Edge-devices and microcontrollers and how the accuracy and memory will affect while converting the Deep Learning model for Edge devices.

**Keywords:** Edge computing, Deep learning, IoT, Embedded device ML, TinyML.

## 1.INTRODUCTION

Connected Devices of IoT (Internet of Things) have increased exponentially over the past few years and are predicted to reach 1 trillion across various market segments such as AR, Health care, smart home, smart industry and so on by 2035. These IoT devices typically consist of

 https://doi.org/10.5281/zenodo.13934928

sensors to collect data including audio, video, GPS, Temperature, humidity etc. Most of the collected data from sensors are noisy and these raw data are processed by analytics tools in the cloud to enable a wide range of applications. Running the analytics tool on the cloud, have some drawbacks such as latency, scalability, and privacy.

**Latency:** Real-time inference is critical to many applications. For example, healthcare, an autonomous vehicle, and a voice-based-assistive application needs quick response.

**Scalability:** Network access to the cloud can become a bottleneck as the number of connected devices increases. Uploading all data to the cloud is also inefficient in terms of network resource utilization, particularly if not all data from all sources are needed for deep learning.

**Privacy:** Sending data to the cloud risks privacy concerns from the users who own the data or whose behaviours are captured in the data.

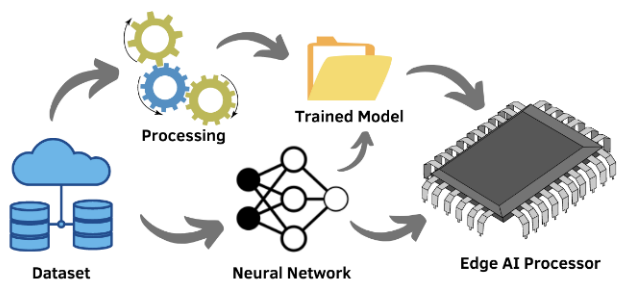



**Fig 1.1. Deep learning Model on Edge devices**

Edge computing is a viable solution to meet the latency, scalability, and privacy challenges described earlier in this section. Edge Computing is computation done on the edge-devices or edge server which is close to the source of data.

The research gap in data analytics of IoT applications in edge computing using deep learning is evident from the need for techniques that can handle large volumes of data, heterogeneity and variability of data, deployment optimization, and privacy and security concerns. Addressing these gaps will help unlock the full potential of IoT applications for real-world use cases.

## 2. DEEP LEARNING METHODS FOR EDGE COMPUTING

**RBM** Restricted Boltzmann machine (RBM) is a kind of probabilistic graphical models that can be interpreted as stochastic neural networks. A typical two-layer RBM includes a visible layer that contains the input we know and a hidden layer that contains the latent variables.


https://doi.org/10.5281/zenodo.13934928

**Auto Encoders** An autoencoder includes an input layer and an output layer that are connected by one or multiple hidden layers. The shape of the input layer and the output layer are the same.

**DNN** A deep neural network (or deep fully connected neural network) usually has a deeper layer structure for more complicated learning tasks. DNN consists of an input layer, several hidden layers, and an output layer, where the output of each layer is fed to the next layer with activation functions

**CNN** Convolutional neural networks (CNNs) are designed to process data that comes in the form of multiple arrays. CNN receives 2D data structures and extracts high-level features through convolutional layers which is the core of CNN architecture

**RNN** Different from CNNs that are good at abstracting spatial features, recurrent neural networks (RNNs) are designed for processing sequential or time-series data. The input to an RNN includes both the current sample and the previously
observed samples.

**DRL** Deep reinforcement learning (DRL) is a combination of deep learning (DL) and reinforcement learning (RL) It aims to build an agent that can learn the best action choices over a set of states through the interaction with the environment.

### 2.1. DEEP NEURAL NETWORK MODEL ON EDGE DEVICES

Edge devices have their own limitation on energy, memory, and computation capabilities. Implementing deep learning models on edge devices is challenging because of these limitations. With the new advancement of computing technology and hardware technology, it is possible to implement deep learning on edge devices**.** In this paper, we review the suitable Deep learning algorithms run on edge devices and the methods to fix the gap between the deep learning from the cloud to the edge devices.

To develop an efficient deep learning model for IoT applications is the need for lightweight neural networks that require less computational resources. While there has been significant progress in developing deep learning models for various applications, these models are often too large and computationally intensive to be deployed on edge devices with limited resources [12],[13],[17],[18]. There is a need for research that focuses on developing more efficient deep learning models that can run on low-power devices such as microcontrollers [22],[23],[24], without sacrificing accuracy. Additionally, there has been researching on developing deep learning models that are specifically designed for specific IoT applications, such as activity recognition [22],[23] or anomaly detection in sensor data

Deep learning can be used to perform both supervised learning and unsupervised learning. The metrics of success depend on the application domain where deep learning is being applied. To convert the deep learning model runnable on edge devices majorly use the following techniques. model compression [2],[3] such as Layer pruning [4],[7] drop out [3], reduce the number of neurons available in the model [9], parameter quantization [4],[5],[6] and so on.

*Model Design:* When designing DNN models for resource-constrained devices, machine learning researchers often focus on designing models with a reduced number of parameters in the DNN model [6],[7], thus reducing memory and execution latency, while aiming to preserve high accuracy, there are many techniques for doing so, and we briefly mention several popular deep learning models for resource-constrained devices drawn from computer vision.

*Model Compression:* Compressing the DNN model is another way to enable DNNs on edge devices. Such methods usually seek to compress the existing DNN models [7],[9] with minimal accuracy loss compared with the original model. There are several popular model compression methods: parameter quantization [16], parameter pruning, and knowledge distillation

*Hardware:* To speed up inference of deep learning, hardware manufacturers are leveraging existing hardware such as CPUs and GPUs, as well as producing custom application-specific integrated circuits (ASICs) for deep learning, such as Google’s tensor processing unit [8], [10] (TPU).

*Software:* software accelerator [10] for low-power devices, RS TensorFlow for light devices to construct DNN model. The different techniques to deploy the DNN model in Edge devices is analysed in Table 2.1.

| SI. No | Author | Methodology | Algorithms | Application | Devices | Significance |
|---|---|---|---|---|---|---|
| 1. | HeLi, et.al | Adaptive Deep Learning -Prediction of DNN model based on the Input image [2] | Inception, Resent and mobile net | Image Classification | NVIDIA Jetson X2 | Reduce inference |

 https://doi.org/10.5281/zenodo.13934928

| | | | | | | |
|---|---|---|---|---|---|---|
| 2. | S. Liu, Y. Lin, Z. Zhou 2017 | DeepIoT-Model Compression -Dropout (Get existing weight of original model and adjust based redundancy) [3] | LeNet, VGGNET And others | Text, Image and Speech recognition | Intel Edison computing platform | Still use existing libraries Reduce model size, shorten execution time and consume less energy |
| 3. | Lai et.al | CMSIS-NN, a library of optimized software kernels to enable deployment of NNs Using Fixed point quantization [4] | 7-layer CNN | Image classification, Keyword spotting | Arm cortex M Micro controller | Minimal memory, S. Han et al suitable for keyword spotting application |
| 4. | S. Han et al | ESE – Efficient Speech Recognition Engine with Sparse LSTM on FPGA [5] 1.Load balance aware pruning 2.Quantaization | LSTM | Speech recognition | XCKU060 FPGA | Reduce the model size and increase accuracy |
| 5. | Bhattacharya et.al | sparse factorizations Separation of Deep Learning Layers for Constrained Resource Inference on Wearables [6] | Alexnet, VGGnet | Speech and image recognition | Qualcom snapdragon 400 | Optimize convolution filters, Mobile and IoT platforms |
| 6. | S. Yao et.al | AdaDeep – On demand Deep model compression Automatic Hyperparameter Optimization [7] | Lenet, AlexNet and VGGNet | Image,Activity and Audio | Smartphones and wearables | Automatic compression based on user requirements |

| | | | | | | |
|---|---|---|---|---|---|---|
| 7. | N. Loc et.al | Mobile GPU-Deep Mon – reuse intermediate partial results while processing convolutional layers caching mechanism reduces this repetitive computation significantly [8] | YOLO, MatConvNet | Object Detction | Samsung galaxy S7 | Suitable for video frames 1 and 2 per second |
| 8. | Y. Wang et.al | RSTensorFlow [9] | 24-layer CNN, LSTM | Hand gesture recognition | Nexus 5x | supports heterogeneous computing resources for commodity Android device |
| 9. | N. D. Lane et al | DEEPX Software Accelerator [10] | Alexnet and others | Speech and image | Qualcom snapgragon 800 | Support mobile machine learning inference |

Table 2.1. Deep learning models on Edge Devices.

## 2.2. DEEP LEARNING ON MICROCONTROLLERS

Tiny Machine Learning (TinyML) brings cognitive capabilities to resourced-constrained IoT devices such as Microcontroller units. The MCUNet is used to jointly design the neural network architecture (TinyNAS) and the inference library (TinyEngine), enabling deep learning on tiny hardware resources [22],[23],[24]. It enables low-latency, low power and low bandwidth model inference at edge devices. While a standard consumer CPUs consume between 65 watts and 85 watts and a standard consumer GPU consumes anywhere between 200 watts to 500 watts, a typical microcontroller consumes power in the order of milliwatts or microwatts. That is around a thousand times less power consumption. This low power consumption enables the TinyML devices to run unplugged on batteries for weeks, months, and in some cases, even years, while running ML applications on edge. The Deep Learning model deployed on various microcontrollers is analysed

 https://doi.org/10.5281/zenodo.13934928

Figure 2.1. LATENCY RESULTS

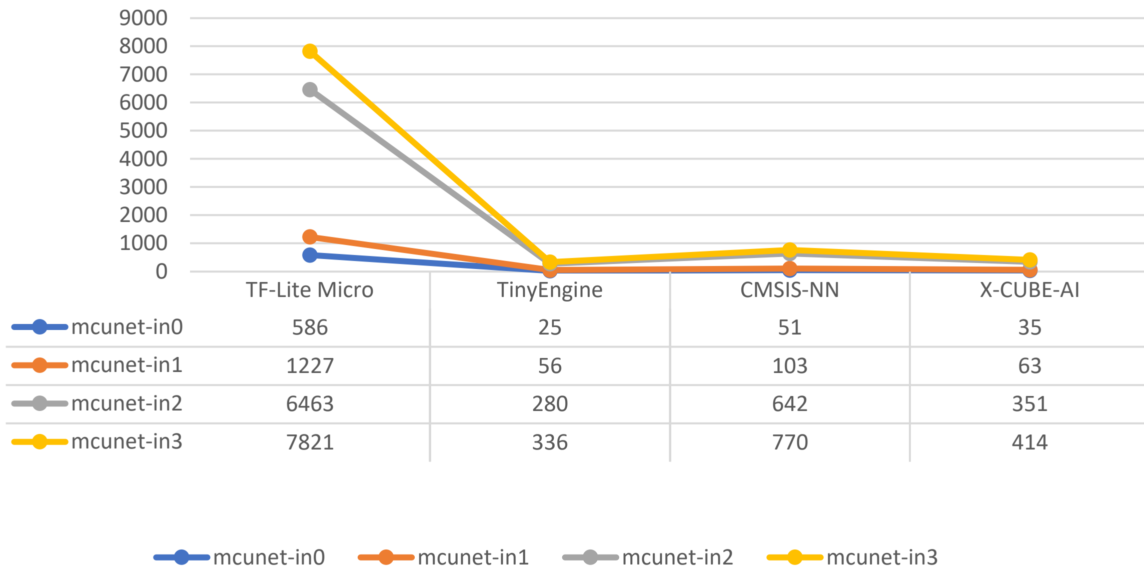



Figure 2.2. PEAK MEMORY RESULTS

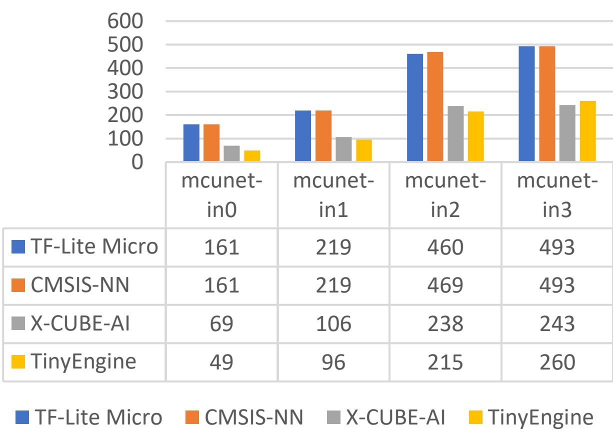
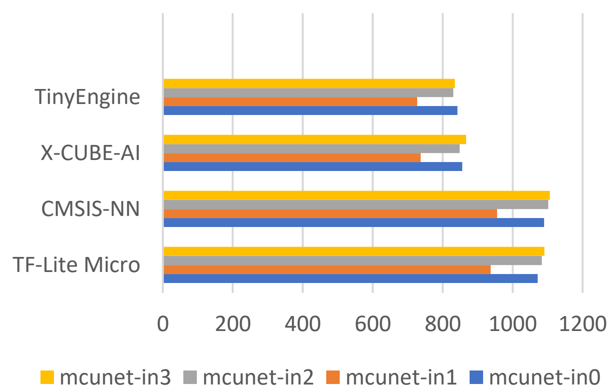



Figure 2.3. FLASH MEMORY USAGE




in Table 2.1. The TinyEngine Model is compared with the other MCUNET model in terms of latency as milli-seconds displayed in Figure 2.1, Memory usage as KB displayed in Figure 2.2 and flash memory as KB usage displayed in Figure 2.3.

| SI.NO | Author | Methodology | Algorithms | Application | Devices | Significance |
|---|---|---|---|---|---|---|
| 1 | Lin, J., et al. | Memory-Efficient Patch-based Inference [22] | MCUNetV2 | Wake word prediction | 256kB SRAM/1MB Flash and 512kB SRAM/2MB Flash | Wearable devices & achieve >90% accuracy under only |

 https://doi.org/10.5281/zenodo.13934928

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | 32kB SRAM, |
| 2 | Lin, Ji, et al. | Tiny NAS Tiny Engine [23] | MCUNet | Wake word prediction | Micro Controllers Arduuino nano BLE Sense | ImageNet accuracy (70.7%) |
| 3 | Lin, Ji, et al. | Quantization-Aware Scaling [24] | Tiny Training Engine | Wake work predication | Micro controller 256KB SRAM and 1MB Flash | Reduce Training memory 1000X compared with PyTorch and TensorFlow) |

Table 2.1 Deep Learning Models on Micro controllers

## CONCLUSION

In conclusion, deep learning on edge devices for IoT is an emerging area of research that has gained significant attention in recent years. In this paper, we conducted a survey of the state-of-the-art deep learning techniques for edge devices in IoT applications. We discussed the challenges associated with deploying deep learning models on resource-constrained edge devices and highlighted various approaches that have been proposed to overcome these challenges. We also reviewed various hardware and software frameworks (TinyML) that can be used to deploy deep learning models on edge devices, including TensorFlow Lite, PyTorch Mobile, and Edge TPU. Furthermore, we discussed the performance metrics that are commonly used to evaluate the effectiveness of these frameworks, including accuracy, latency, and energy consumption.

## REFERENCES

1. J. Chen and X. Ran, "Deep Learning With Edge Computing: A Review," in Proceedings of the IEEE, vol. 107, no. 8, pp. 1655-1674, Aug. 2019, doi: 10.1109/JPROC.2019.2921977.
2. B. Taylor, V. S. Marco, W. Wolff, Y. Elkhatib, and Z. Wang, “Adaptive deep learning model selection on embedded systems,” in *Proc. LCTES*, 2018,pp. 31–43.
3. S. Liu, Y. Lin, Z. Zhou, K. Nan, H. Liu, and J. Du, “DeepIoT: Compressing deep neural network structures for sensing systems with a compressor-critic framework,” in *Proc. SenSys*, 2017, pp. 1–4

 https://doi.org/10.5281/zenodo.13934928

4. L. Lai and N. Suda, “Enabling deep learning at the IoT edge,” in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, 2018, p. 135.
5. S. Han et al., “ESE: Efficient speech recognition engine with sparse LSTM on FPGA,” in Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays (FPGA), 2017, pp. 75–84.
6. S. Bhattacharya and N. D. Lane, “Sparsification and separation of deep learning layers for constrained resource inference on wearables,” in *Proc. 14th ACM Conf. Embedded Netw. Sensor Syst. CD-ROM (SenSys)*, 2016, pp. 176–189.
7. S. Yao, Y. Zhao, Z. Aston, L. Su, and T. Abdelzaher, “On-demand deep model compression for mobile devices: A usage-driven model selection framework,” in *Proc. MobiSys*, 2018, pp. 389–400.
8. N. Loc Huynh, Y. Lee, and R. K. Balan, “DeepMon: Mobile GPU-based deep learning framework for continuous vision applications,” in *Proc. ACM MobiSys*, 2017, pp. 82–95.
9. M. Alzantot, Y. Wang, Z. Ren, and M. B. Srivastava, “RSTensorFlow: GPU enabled tensorflow for deep learning on commodity android devices,” in Proc. 1st Int. Workshop Deep Learn. Mobile Syst. Appl. (EMDL), 2017, pp. 7–12.
10. N. D. Lane et al., “DeepX: A software accelerator for low-power deep learning inference on mobile devices,” in Proc. 15th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw. (IPSN), 2016, p. 23.
11. Manogaran, Gunasekaran, et al. "Wearable IoT smart-log patch: An edge computing-based Bayesian deep learning network system for multi access physical monitoring system." *Sensors* 19.13 (2019): 3030.
12. Huda, SM Asiful, and Sangman Moh. "Survey on computation offloading in UAV-Enabled mobile edge computing." *Journal of Network and Computer Applications* (2022): 103341.
13. Sulieman, Nour Alhuda, Lorenzo Ricciardi Celsi, Wei Li, Albert Zomaya, and Massimo Villari. "Edge-oriented computing: A survey on research and use cases." *Energies* 15, no. 2 (2022): 452.


https://doi.org/10.5281/zenodo.13934928

14. Quy, Vu Khanh, et al. "Smart healthcare IoT applications based on fog computing: architecture, applications and challenges." *Complex & Intelligent Systems* 8.5 (2022): 3805-3815.
15. Hartmann, Morghan, Umair Sajid Hashmi, and Ali Imran. "Edge computing in smart health care systems: Review, challenges, and research directions." *Transactions on Emerging Telecommunications Technologies* 33.3 (2022): e3710.
16. Zhang, Michael, et al. "Seneca: Fast and low cost hyperparameter search for machine learning models." *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*. IEEE, 2019.
17. Zhang, Michael, Chandra Krintz, and Rich Wolski. "Edge-adaptable serverless acceleration for machine learning Internet of Things applications." *Software: Practice and Experience* 51.9 (2021): 1852-1867.
**18.** Zhang, Michael, Chandra Krintz, and Rich Wolski. "Stoic: Serverless teleoperable hybrid cloud for machine learning applications on edge device." *2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 2020.
19. Mazumder, Mark, et al. "Few-shot keyword spotting in any language." *arXiv preprint arXiv:2104.01454* (2021).
20. Kumar, Yogesh, Apeksha Koul, and Chamkaur Singh. "A deep learning approaches in text-to-speech system: a systematic review and recent research perspective." *Multimedia Tools and Applications* 82.10 (2023): 15171-15197.
21. Banbury, Colby R., et al. "Benchmarking tinyml systems: Challenges and direction." *arXiv preprint arXiv:2003.04821* (2020).
22. Lin, J., et al. "MCUNetV2: Memory-Efficient Patch-based Inference for Tiny Deep Learning. arXiv 2021." *arXiv preprint arXiv:2110.15352*.
23. Lin, Ji, et al. "Mcunet: Tiny deep learning on iot devices." *Advances in Neural Information Processing Systems* 33 (2020): 11711-11722.
24. Lin, Ji, et al. "On-device training under 256kb memory." *arXiv preprint arXiv:2206.15472* (2022).