Mahdi Abbasi Research Statement

Artificial Intelligence (AI) and Machine Learning (ML) techniques are rapidly being adopted in our daily lives due to advancements in data, algorithms, and hardware. However, the challenge of designing and implementing systems that can effectively support machine learning (ML) models in different deployment scenarios, ranging from edge to cloud, persists. The gap between the potential of machine learning (core ML algorithms and methods) and its practical application on various computing platforms is a significant factor.

My main focus is computer architecture. I always enjoy innovative computing architectures and hardware/software co-design techniques that accelerate computing tasks, especially for efficient AI/ML computing in Internet of Thing (IoT) networks. my research focuses on bridging the gap between machine learning and hardware systems, which is a new and emerging field of study. My research focuses on AI/ML on modern computing platforms such as CPU, GPU, as well as programmable accelerators like TCAMs, FPGAs, smart network interface cards (NICs), and programmable data plane switches. Additionally, I explore in-network acceleration of simple yet highly practical ML models, especially in 5G/6G wireless sensors networks where flow-based processing of the internet packets is highly demanded by intelligent applications that analyze big volumes of data in a limited time. I am extending my research domain toward the new horizons of co-optimization of deep learning algorithms and hardware, efficient training methods for deep learning, and support for complex deep learning models on edge of IoT.

Past research

[P1] In-network Computing for Accelerating ML-based Security Functions

After completing my PhD, I started teaching a Network Processor course in our department, leveraging my expertise in the field. Teaching this course not only expanded my understanding of modern programmable network processor systems, but also provided me with valuable insight into the potential of in-network computing.

In-network computing involves offloading computing tasks to programmable network elements that already exist within the network and are used to forward traffic. The computation task can refer to a program, process, operations, or network functions. Additionally, the network elements involved in the process, such as programmable switches and routers, TCAMs, FPGAs, and smart NICs, can be programmed to perform computations in addition to their routing or forwarding tasks. In the case of not using the network elements, the computation must be done by general purpose processors in multi-core or many-core systems (e.g., server, controller)[1].

I began my research during the years 2012 to 2018, when this novel technology was in its infancy. I supervised over 20 postgraduate theses, each exploring the utilization of In-network computing. The primary function in this technique is packet classification[2, 3]. The packet classification is to categorize packets by comparing their header fields (and sometimes their payload) using a specific classification algorithm against a set of rules in the form of condition-action pairs. These rulesets in software switches are stored in SRAM, while in hardware switches, they are stored in TCAM. The main concept behind utilizing programmable switches for in-network computing is to employ a parser that converts the desired AI/ML function into a series of packet classification rules [4, 5]. A high-performance computing architecture, such as a modern general-purpose computing system with multiple processing elements like CPUs or GPUs, or a hardware switch with a fully-parallel TCAM-based pipeline or FPGA-based accelerator board, can efficiently perform packet classification on incoming packets. My research in this era focuses on two following main aspects.

[P1-1] We first examined the feasibility of mapping decision tree (DT)-based machine learning models for internet flow classification to packet classification rules. Such models are frequently utilized in intelligent flow-based data pruning systems or dynamic ML-based security mechanisms[6], including firewalls [7], access control lists[8], and IP chains[9]. These ML-based intelligent systems typically operate on powerful servers. Our primary research results showed that it is possible to convert basic decision-tree-based machine learning models for flow classification into packet classification rulesets [10-12].

[P1-2] In the next step, we focused on hardware-software co-design techniques to accelerate packet classification, which is the core process of in-network computing. We utilized co-design techniques to expedite this fundamental process and achieve network speeds of terabits per second. Our research spans the following fields.

[P1-2-1] In some of our research, we utilized advanced algorithmic design and data structure enhancements to improve packet classification on traditional network processor architectures [13-16]. In a previous study [13], we introduced *MBitCuts*, an innovative approach to packet classification that reduces memory usage and latency by modifying the method of bit selection in the cutting of the geometric subspace model of each decision tree node. Additionally, in our *leaf-pushed KD-tree* [16], we utilized leaf-pushing techniques to enhance the KD-tree algorithm. This algorithm is frequently used in firewalls. Our proposed algorithm, which utilizes a bloom filter data structure and a hash table, can significantly increase packet classification speed by up to 24 times compared to the traditional KD-tree method.

[P1-2-2] We conducted a study on the efficient management of TCAM resources on pipelined architectures within modern network processors. The goal was to optimize their usage and minimize energy consumption [17-19]. In the related master project, my student Shakoor introduced a new mapping technique to decrease memory usage in TCAM blocks, which are utilized in hardware packet classifiers [19]. Our algorithm optimizes rule distribution in TCAM blocks. The results demonstrated that the proposed method achieves a more balanced distribution of rules in the TCAM block compared to competitor architectures. Additionally, it reduces the number of rules assigned to the general TCAM block. We then created a new multi-layer approach for encoding port ranges and storing them in TCAM [19]. The proposed method not only increases the speed of the classification architecture but also utilizes the capacity of TCAM more efficiently. In the final phase of the project, we aimed to decrease power consumption in the TCAM layers during parallel search and rule updates [18]. We utilized the key concept of minimizing search ranges and reducing the number of displacements during TCAM updates. The high-level synthesis and evaluation of the proposed method showed a reduction of over 50% in resource utilization and nearly a two-fold improvement in search and update latency.

[P1-2-3] In another project, my student, Navid used the programmability of FPGA elements to classify flows within the network. We developed a pipelined flow classification engine for intelligent data processing at the edge layer of IoT [20, 21]. In this project, we developed a micro-architecture for packet classification in high-speed, flow-based wireless network systems. The classifying micro-core uses hashing to fix the length of the rules. Then, by combining SRAM and BRAM memory cells and implementing two ports on Virtex®6 FPGAs, the architecture achieves a power consumption of 1.294w. Furthermore, with a frequency of 233 MHz, its throughput exceeds 147 Gbps.

[P1-2-4] I conducted research projects to explore the potential for speeding up packet classification, which is a core function of in-network computing, on parallel multicore and many-core platforms [22, 23], as well as hybrid clusters of CPUs and GPUs. This platform is primarily utilized in wireless network base stations, where multiple sensor nodes are connected through base stations. The base stations typically consist of network switches equipped with standard processors. Our results demonstrated that utilizing efficient multi-threading techniques such as TBB can significantly enhance the classification process, achieving up to an 8 times acceleration on a quad-core processor system wireless switch. Milad, Azad, and Azam, my master students, did research on accelerating decision-tree based in-network flow classification using GPUs [24, 25], CPU clusters [24, 25], and hybrid clusters of CPU and GPU [26, 27]. During our study on GPU acceleration, we combined asymptotic and calibrated analysis frameworks to develop a more efficient framework. This framework not only facilitates the design of efficient parallel flow classification algorithms for various GPU architectures, but also serves as a powerful analysis tool for predicting empirical results. During our study on cluster architectures of CPUs and GPUs, we examined the best ways to use distributed hierarchical storage and computing resources in hosts and devices for in-network flow classification.

Ongoing Research

[O1] **Online Edge Orchestration**

The concept of edge computing can be traced back to the 1990s when Akamai launched its content delivery network (CDN). Edge computing involves processing data closer to the source (edge devices) rather than relying on centralized cloud or data centers.

Intelligent edge devices, including billions of mobile phones and IoT devices, equipped with advanced sensors, have become a common part of our daily lives. Deploying AI to edge devices is attractive for a range of real-world applications, including smart homes, smart retail, smart manufacturing, autonomous driving, and more. Regarding my previous research results in in-network computing, I have considered the following steps for my research team.

[O1-1] **Rule-based MLs**: Following our recent research on "the feasibility of mapping decision tree-based machine learning models to equivalent sets of condition-action rules" (please, recall part A of this document), we looked for more complex ML paradigms that *are inherently rule-based.* In rule-based machine learning (RBML), the ML models identify, learns, or evolves 'rules' to *store*, manipulate or apply online. The rules are typically taking the form of a condition-action expression. The rule-based machine learning methods typically comprise a set of rules, or knowledge base, that collectively make up the online prediction model. Such systems can identify and adapt their own set of rules. Among different RBMLs, we found Learning Classifier Systems (LCS) the most implicational models [28, 29]. We selected XCS learning classifier systems [30]as our base RBML.

[O1-2] **RBMLs for Online Edge Orchestration:** We studied four ingredients of the architecture of edge intelligence to realize the best framework for assessing the efficiency of in-network computation of XCS-based RBMLS. Finally, from among edge caching, edge training, edge inference, and edge offloading, we selected Edge offloading. Offloading is an abstract service in edge intelligence where an edge device, in order to make the best use of all available resources for AI/ML computation, can offload some of its tasks like edge-training, edge caching or edge inference to other edge devices, to edge servers, or to the cloud servers in network.

I defined two master projects to asses if XCS-RBML can efficiently afford dynamic and continuous offloading service in the programmable switches of edge/fog layer? In first Project[31, 32], we developed two methods based on XCS learning classifier systems (LCS), namely, XCS and BCM-XCS, to balance the power consumption at the edge of the network and to reduce delays in the processing of workloads. In this project, the condition part each rule includes binary representation of the cross-layer state of the available computational resources of the network elements, network links, and energy resources, while the action part specifies the computation share of each edge system from whole received tasks. The results of our experiments are indicative of the superiority of BCM-XCS over the basic XCS-based method. The proposed methods distribute the AI workloads in a way that the delay in their processing and the communication delay between the cloud and edge nodes are both minimized. In addition to considerable advantages in controlling the fluctuations of the processing delay, the proposed methods can simultaneously reduce the processing delay by 42% by using a moderate power consumption at the edge of the network. The proposed methods can also recharge the renewable batteries used at the edge of the network about 18 percent more than the best state-of-the-art method[33].

The second project we approximated the load distribution function among edge systems with two methods based on batch learning systems called XCSF and BCM-XCSF in edge/fog nodes. These two methods differ from our methods in former project in having a memory to store the best rules. Experiments show that these two methods outperform XCS and BCM-XCS, in load distribution, and specially the BCM-XCSF method, in addition to reducing the computational overhead can reduce the processing delay about 60% while optimizes energy consumption.

Our results at this stage made us sure that if we run the RBML-based offloading mechanism in the form of network processing, we can not only achieve excellent performance in managing resources and running distributed AI services at the edge of the network, but it can also run at a higher speed.

[O1-3] **RBML Compression (Tiny RBMLs):** Given the limitations of computing resources (such as memory and power) at the programmable network elements, we needed to produce a tiny model of our XCS-based offloader. This tiny models must fit in storage, computation, and energy requirements to the programmable elements in edge switches. Hence, in project of *Tiny-offloader*, we used fine-tuning and model compression techniques to achieve tiny forms of DT-based and XCS-based offloader models that can easily be embedded in programmable elements of edge switches[34]. The results demonstrate that the proposed TinyXCS outperforms TinyDT model, and both models with less than 3KB size, can reduce the processing delay and save energy, like XCS-based offloader.

In the future works, we can exploit programmable network elements (like smart NIC, TCAM, FPGA, or programmable SDN switch) to run our tiny models. As a part of our future research, the in-network computing will accelerate the TinyDT and Tiny-XCS edge orchestrators.

[O2] Efficient Deep Learning at Edge

Deep learning (DL) techniques are enjoying rapidly increasing adoption in our daily life, due to the synergistic advancements across data, algorithm, and hardware. However, designing and implementing systems that can efficiently support DL models across various deployment scenarios from edge to cloud remains a significant obstacle, in large part due to the gap between deep learning's promise (core DL algorithm and method) and its real-world utility (diverse and heterogeneous computing platforms from edge to cloud). Hence, I am conducting two research area at the intersection of machine learning and hardware systems to bridge the gap.

[O2-1] ResFed: Light Federated Multi-Shot Pre-Trained Model on Edge Devices [35]

Many real-world applications require real-time prediction on resource-constrained devices. Therefore, predictive models should be designed in a way that they impose little storage space overhead and low computational complexity, without significant damage to their accuracy. Furthermore, these devices often suffer from limited sample availability, adversely affecting the learning and adaptation capabilities of the deployed models. The use of federated learning enables devices with limited data access to increase the learning and adaptation capabilities of machine learning models without sharing their private data. The use of meta-learning algorithms, especially for Few-Shot learning, is suitable for edge computing in industrial IoT where non-uniform and highly personalized decentralized training data exist due to their fast adaptation and good generalization to new tasks.

[O2-1-1] In the first step of project, a federated meta-learning method specifically tailored for Few-shot classification has been developed by our research team. Despite recent advances, the use of metric-based meta-learning methods remains elusive due to their simplicity, as well as their development to improve model learning and accuracy in a federated IoT environment. Overcoming the existing challenges associated with metric-based meta-learning methods, ResFed leverages the advantages of pre-trained models and data augmentation within the federated meta-learner, resulting in good performance. Our initial experimental results demonstrate that in our proposed method for settings with limited data in federated IoT environments, the accuracy has increased by 8.24-25.58% and the resource cost has been reduced by x1.02 to 2.62 [35].

[O2-1-2] As the second step, we are designing a bi-level cognitive orchestrator mechanism based on recently developed TinyXCS model. At higher level, a central cognitive coordinator is being considered in the cloud environment for coordinating edge nodes. This model decides if an edge node will contribute to update of the centralized model (via federated aggregation) based on the received state of the node. At the second level, a TinyXCS-based intelligent orchestrator is being designed. This orchestrator decides if the received computational task should be offloaded to the edge nodes or not. Although some objects such as mobile phones, industrial IoT devices, and smart vehicles can perform some artificial intelligence calculations, the computing capability and energy consumption still limit their abilities in artificial intelligence calculations. Therefore, we suggest that objects have a two-way communication with their edge and, if necessary, transfer their tasks to the edge and use the obtained results.

In the future works, we will consider in-network computing to accelerate intelligent offloading, Inference, and training.

[O2-2] FTDCNN: Efficient Fault Tolerant Distributed Convolutional Neural Networks [36]

Large convolutional neural networks (CNNs) can extract better features, but additional layers come at the cost of added memory, latency, and energy usage. Moreover, they cannot be trained on a small training set. While CNNs can be distributed on the edge and cloud, distributing them in the cloud can increase privacy risks and latency.

To address these issues, in this research, we distribute CNNs over the edge (fog) and end devices. This approach involves dividing a large neural network model into several smaller neural networks and distributing them across the end devices. This not only maintains accuracy but also solves the problem of resource limitations on the end devices. However, the accuracy decreases in case of end device failure. To mitigate this, we introduce a modifier module at the edge to improve results in the event of network or end device failure. The modifier module is a CNN that is built using the neural networks on the end devices and runs on one active edge device as needed. The modifier works in conjunction with an aggregator in edge level. Our recent results have shown that, in the case of computational failure or intermittent network communications in one edge device, this fault modifier can preserve the accuracy by approximately 1% tolerance as compared to the normal scenario.

In the future works, the in-network computing will be exploited to accelerate the modifier and aggregation process.

[O3] **DeepADD: Approximating DNNs based on ADD and Pre-Computed Exponents [37]**

In this work, a novel hardware design paradigm utilizing approximate computing using shift and add operations for accelerating deep neural networks (DNNs) on Field-Programmable Gate Arrays (FPGAs) through High-Level Synthesis (HLS) is being developed. By leveraging approximate computing, we balance precision and efficiency, leading to faster and more energy-efficient DNN inference. Our methodology capitalizes on the inherent redundancy and error tolerance of DNN computations, approximating weight multiplication using shift and add operations. This simplifies hardware structures, reduces power consumption, and maintains acceptable accuracy across various DNN tasks. The integration of shift-based approximate computing within HLS streamlines hardware module development, demonstrates successful real-world implementation and competitive performance gains. Our experimentation examines resource utilization, performance, and energy trade-offs, confirming the effectiveness of our approach and the potential for FPGA-accelerated DNNs using shift operations. This project has been implemented in two steps.

[O3-1] In the first step of this project, we deployed our technique using Python programming language on a Perceptron neural network with limited number of neurons. we reduce the operation of multiplying the weights and the inputs of different layers of the neural network to the simple addition operation. For this purpose, we round the inputs as well as weights of each layer to the nearest power of 2. Precomputing these limited powers of two, and storing them in a list of key-value format in SRAM cells inside FPGA, lets us use addition instead of multiplication and power computation. In the backward propagation process, gradients are calculated and weights are updated in a standard way. To avoid gradient explosion in backward propagation, we use gradient clipping and L2 normalization techniques. Our initial experimental results show that by using the proposed DeepADD mechanism in on MNIST dataset, compared to the original model, the speed of the approximated computations has increased by 3.38% and 9.30% in the training and testing phases respectively, while preserving the accuracy with 3% of tolerance.

[O3-2] In the second step, we use High-Level Synthesis tools in of Xilinx to generate the low-level netlist for FPGA synthesis. The results show that our method, reduces the number of lookup Tables (LUTs) and Flip-Flops (FFs) from 88, and 46 to 85, and 38, respectively.

In the future works, the ideas gained from the implementation of this project would be utilized for acceleration of large language models (LLMs) in edge.

Future Research

Recent developments in ML are set to revolutionize pharmaceutical, mechanical autonomy, commerce, transportation, and numerous other angles of our lives. Such transformative impacts are predicated on giving high-performance compute capabilities that empower learning of compute-intensive ML models, and continually progressing ML calculations that can receive to ever-changing application needs. Current ML systems mostly depend on completely offloading insights to the cloud. A promising solution is recently predicted by introducing edge intelligence.

I believe that creating new knowledge requires development of learned knowledge. Hence, In-line with ongoing research, my first planned research for future will focus on hardware-software co-design techniques efficient machine learning at the edge, while my second planned research for future will investigate innovative co-design techniques to promote computational architectures for efficient computation of novel models of AI/ML at Edge.

[F1] full stack solution for In-Network ML computing

Regarding the objectives of our ongoing research [O1] for online edge orchestration, currently we develop tiny models of rule based machine learning models, named TinyXCS and TinyDT models for light but efficient orchestration of tasks in edge. As fully explored in description of our first ongoing research [O1], the data structure and algorithmic features of these ML models satisfy the feasibility requirements of deployment by in-network computing. Also, in our second era of ongoing research [O2] we noticed that distributed architectures for ML training, and especially federated learning (FL), provide a platform to accelerate learning, inference and aggregation tasks by pushing them on programmable network elements[2].

Hence, regarding this research we plan to provide a full computing stack that constitutes programming language constructs, high-level optimization and synthesizer plugins for compiler, system software, template architecture, and target-aware circuit/table generators, which enable acceleration of MLs on different programmable network elements at scale. Our planned computing stack would get the model description code and, regarding the targeted programmable network element will produce an optimized netlist for FPGA or ASIC implementation, as well as rule-set representation for implementation on TCAM and programmable switches.

Nonetheless, our solution does not require ML programmer to delve into the time-consuming task of system software development or hardware design.
Our full-stack solution will reach required levels of efficiency, programmability, and automation in generating the hardware and software codes from its high-level descriptions. The high-level multi-objective optimizer will allow our solution to competently exploit the numerous resources that are becoming available on modern programmable network elements. The high-level synthesizer module of our solution can offer a specialized system software that optimizes task scheduling, resource allocation, and internode communications over a set of heterogeneous programmable network elements. Our planed system will provide a high-level programming interface for automated in-network acceleration of important classes of ML.

[F2] **Designing Hardware Accelerators for Transformer Neural Networks in Edge Computing Using High Level Synthesis on FPGA**
Transformer deep learning models, which were first introduced to solve machine translation problems, were quickly adopted as a question-and-answer model in the field of natural language processing. Then, new researches showed that transformer networks have an extra capability in performing different tasks. The use of these models has been rapidly expanded in other fields as well. Today, in the field of machine vision and speech recognition, models based on transformers provide much better results compared to traditional models such as convolutional and recurrent neural networks. The transformer network uses a structure called self-attention to solve the limitation of recurrent loops in recurrent neural networks (for understanding long-term dependencies in sentences). Transformer networks are heavy models of ML which includes various components that generally consist of two encoder and decoder parts. One of the first networks based on this structure is the two-way representation of transforms (BERT) neural model, which is used for natural language processing[38]. Other networks based on this structure include massive language models such as GPT, Llama and PaLM [39]. Models such as ViT, DeiT have also been presented with the aim of using transforms in machine vision [40].
Reconfigurable chips and especially FPGAs are used as an effective hardware framework to increase the speed of training and inference of neural networks. Compared to the central processor (CPU) and the graphics processor (GPU) which their hardware and software are designed independently, FPGA chips can be used exclusively for target algorithms, which enables the co-design of the logic and computation required in hardware. In addition, FPGA chips can be used as fine-grained systems in in-network computing. They have a perfect ability to balance between latency and high performance with better energy efficiency.
Existing FPGA accelerators are mainly designed for traditional networks such as convolutional and recurrent neural networks. Due to the existence of structural changes between traditional models and transformers, the current methods of designing FPGA accelerators cannot be used in the case of transformer networks. On the other hand, design space of the transformer accelerators for edge systems, is limited by memory size, power consumption and response time of the edge systems.
In this planned research, significant challenges and issues in using transformer networks in resource-constrained network edge devices, such as mobile devices are being fully investigated. Despite the very high performance of this type of MLs, due to their high consumption of computing resources, issues in edge systems such as low response time and minimized energy consumption limit the possibility of using such models in portable systems. For example, translating a 30-word sentence into a huge language model requires 13 billion floating point operations and 20 seconds of time on a Raspberry Pi device [41]. Long latency can affect the user experience in edge systems and reduce their satisfaction level. Hence, regarding the ever-increasing use of mobile devices, there is a need to develop accelerators for transformers to run them efficiently and quickly on such devices with limited computing resources. In hardware accelerator design, in addition to issues such as real delay in execution, due to the variety of parameters that change the structure of transformers, high flexibility the design should be considered.
The key ideas that will help us in this project, mostly come from our results in our ongoing DeepADD project. The main ideas explored in the following.

- Using the approximate calculation method simplifies the multiplication and accumulation operation (MAC) in the process of multiplying the matrix in the network vector. Also, in addition to reducing the resources consumed by the FPGA, we can increase the working frequency and throughput.
- By using data quantization (reduced fixed-point representation), the number of bits to represent the network weights can be reduced. As a result, more coefficients can be stored in the internal memory of the same FPGA, which leads to a reduction in external memory access and energy consumption.
- Pruning the pre-trained weights of the network leads to a reduction in the number of multiplication operations in the accelerator architecture, which can be used to solve the limitation of processing resources available in the FPGA chips of edge hardware.

Publications

[1] S. Kianpisheh and T. Taleb, "A Survey on In-Network Computing: Programmable Data Plane and Technology Specific Applications," IEEE Communications Surveys & Tutorials, vol. 25, no. 1, pp. 701-761, 2023, doi: 10.1109/COMST.2022.3213237.

[2] C. Zheng et al., "IIsy: Practical in-network classification," arXiv preprint arXiv:2205.08243, 2022.

[3] Z. Xiong and N. Zilberman, "Do switches dream of machine learning? toward in-network classification," in Proceedings of the 18th ACM workshop on hot topics in networks, 2019, pp. 25-33.

[4] M. C. Luizelli, R. Canofre, A. F. Lorenzon, F. D. Rossi, W. Cordeiro, and O. M. Caicedo, "In-network neural networks: Challenges and opportunities for innovation," IEEE Network, vol. 35, no. 6, pp. 68-74, 2021.

[5] C. Jia, C. Li, Y. Li, X. Hu, and J. Li, "An Observation of Packet Classification: Most Rules are at the Top," in IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2022: IEEE, pp. 1-6.

[6] P. Karunadsa, U. Annakkage, and B. MacDonald, "Dynamic security control using secure regions derived from a decision tree technique," in 2000 Power Engineering Society Summer Meeting (Cat. No. 00CH37134), 2000, vol. 3: IEEE, pp. 1861-1865.

[7] S. Vakilian, "A TCAM-based packet classifier considering memory bounds," Master, Isfahan University of Science and Technoogy, 2017. [Online].

[8] M. Dalton, C. Kozyrakis, and N. Zeldovich, "Nemesis: Preventing authentication & [and] access control vulnerabilities in web applications," 2009.

[9] L. Zhang, "An implementation of scada network security testbed," arXiv preprint arXiv:1701.05323, 2017.

[10] A. Z. M. Esmaelipour, M. Abbasi, "Combining multi-bit tries and TCAMs to enhance packet classification performance in firewalls," presented at the 7th iranian conference on electrical and electronic engineering, Gonabad, 2015-08-19 2015.

[11] A. K. M. Abbasi, "A Traffic-aware packet classification using splay-trees," presented at the 4th National Conference on Information Technology?Computer & Telecommunication Mashhad, 2016, 2016. [Online]. Available: https://civilica.com/doc/668943/.

[12] Sh. Vakilian, M. Abbasi, and A. Fanian, "Increasing the efficiency of TCAM-based tree bundle using dynamic cuts in geometric space," ADST, vol. 6, no. 1, pp. 65-71, 2017.

[13] M. Abbasi, S. Vesaghati Fazel, and M. Rafiee, "MBitCuts: optimal bit-level cutting in geometric space packet classification," The Journal of Supercomputing, vol. 76, pp. 3105-3128, 2020.

[14] N. Khezrian and M. Abbasi, "Comparison of the performance of skip lists and splay trees in classification of internet packets," PeerJ Computer Science, vol. 5, p. e204, 2019.

[15] N. Khezrian, M. Abbasi, and M. Abbasi, "trees in classification of Internet packets."

[16] M. Abbasi, H. Rezaei, V. G. Menon, L. Qi, and M. R. Khosravi, "Enhancing the performance of flow classification in SDN-based intelligent vehicular networks," IEEE Transactions on Intelligent Transportation Systems, vol. 22, no. 7, pp. 4141-4150, 2020.

[17] M. Abbasi, S. Vakilian, A. Fanian, and M. R. Khosravi, "Ingredients to enhance the performance of two-stage TCAM-based packet classifiers in internet of things: greedy layering, bit auctioning and range encoding," EURASIP Journal on Wireless Communications and Networking, vol. 2019, pp. 1-15, 2019.

[18] M. Abbasi, S. Vakilian, S. Vakilian, M. R. Khosravi, and H. Abdoli, "Layered methods for updating AIoT-compatible TCAMS in B5G-enabled WSNs," EURASIP Journal on Wireless Communications and Networking, vol. 2022, no. 1, p. 52, 2022.

[19] S. Vakilian, M. Abbasi, and A. Fanian, "Increasing the efficiency of TCAM-based packet classifiers using intelligent cut technique in geometric space," in 2015 23rd Iranian Conference on Electrical Engineering, 2015: IEEE, pp. 625-630.

[20] S. N. Mousavi, F. Chen, M. Abbasi, M. R. Khosravi, and M. Rafiee, "Efficient pipelined flow classification for intelligent data processing in IoT," Digital Communications and Networks, vol. 8, no. 4, pp. 561-575, 2022.

[21] M. Abbasi, N. Mousavi, M. Rafiee, M. R. Khosravi, and V. G. Menon, "A CRC-Based Classifier Micro-Engine for Efficient Flow Processing in SDN-Based Internet of Things," Mobile Information Systems, vol. 2020, p. 7641073, 2020/05/18 2020, doi: 10.1155/2020/7641073.

[22] M. Abbasi and M. Rafiee, "Efficient parallelisation of the packet classification algorithms on multi-core central processing units using multi-threading application program interfaces," IET Computers & Digital Techniques, vol. 14, no. 6, pp. 313-321, 2020.

[23] M. Abbasi, M. Rafiee, and M. R. Khosravi, "Investigating the efficiency of multithreading application programming interfacesfor parallel packet classification in wireless sensor networks," Turkish Journal of Electrical Engineering and Computer Sciences, vol. 28, no. 3, pp. 1699-1715, 2020.

[24] M. Abbasi and A. Shokrollahi, "Enhancing the performance of decision tree-based packet classification algorithms using CPU cluster," Cluster Computing, vol. 23, no. 4, pp. 3203-3219, 2020/12/01 2020, doi: 10.1007/s10586-020-03081-7.

[25] M. Abbasi, A. F. Najafabadi, S. B. Elghali, M. Zerrougui, M. R. Khosravi, and H. Nasser, "Correction to: High-performance pseudo-anonymization of virtual power plant data on a CPU cluster," Cluster Computing, vol. 26, no. 1, pp. 513-513, 2023/02/01 2023, doi: 10.1007/s10586-022-03615-1.

[26] M. Abbasi, A. Shokrollahi, M. R. Khosravi, and V. G. Menon, "High-performance flow classification using hybrid clusters in software defined mobile edge computing," Computer Communications, vol. 160, pp. 643-660, 2020.

[27] A. Fazel-Najafabadi et al., "High-performance flow classification of big data using hybrid CPU-GPU clusters of cloud environments," Tsinghua Science and Technology, 2023, doi: 10.26599/TST.2023.9010088.

[28] P. L. Lanzi, Learning classifier systems: from foundations to applications (no. 1813). Springer Science & Business Media, 2000.

[29] P. L. Lanzi, "Learning classifier systems: then and now," Evolutionary Intelligence, vol. 1, pp. 63-82, 2008.

[30] M. V. Butz and S. W. Wilson, "An algorithmic description of XCS," in International Workshop on Learning Classifier Systems, 2000: Springer, pp. 253-272.

[31] M. Abbasi, M. Yaghoobikia, M. Rafiee, A. Jolfaei, and M. R. Khosravi, "Efficient resource management and workload allocation in fog–cloud computing paradigm in IoT using learning classifier systems," Computer Communications, vol. 153, pp. 217-228, 2020/03/01/ 2020, doi: https://doi.org/10.1016/j.comcom.2020.02.017.

[32] M. Abbasi, M. Yaghoobikia, M. Rafiee, A. Jolfaei, and M. R. Khosravi, "Energy-efficient workload allocation in fog-cloud based services of intelligent transportation systems using a learning classifier system," IET Intelligent Transport Systems, vol. 14, no. 11, pp. 1484-1490, 2020, doi: https://doi.org/10.1049/iet-its.2019.0783.

[33] M. Abbasi, M. Yaghoobikia, M. Rafiee, M. R. Khosravi, and V. G. Menon, "Optimal Distribution of Workloads in Cloud-Fog Architecture in Intelligent Vehicular Networks," IEEE Transactions on Intelligent Transportation Systems, vol. 22, no. 7, pp. 4706-4715, 2021, doi: 10.1109/TITS.2021.3071328.

[34] M.-R. Pourhosseini, "Using tiny-ML models to QoS-aware workload distribution at the network edge " MSc., Computer Engineering, Bu-Ali Sina, 1670480, 2023.

[35] M. A. Fazeleh Tavassoloan, Abbasi Ramezani, Amir Taherkordi, Mohammad R. Khosravi, "ResFed: An Accurate and Light Federated Multi-shot Pre-Trained Model on Edge Devices," ACM Transactions on Autonomous and Adaptive Systems, Research pp. 1-19, 2023 2023.

[36] M. A. Omid Jamshidi, Abbas Ramezani, Amir Taherkordi, "Fault-Tolerant Deep Neural Networks over the Edge " Bu Ali Sina University, Hamedan, 2023/08/15 2023.

[37] L. Mirzaei-Parsa, "An Efficient Mechanism to Improve Computations in Machine Learning Using Approximate Computing " MSc., Computer Engineering Department, Bu-Ali Sina University, Hamedan, 1674270, 2023.

[38] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018.

[39] W. X. Zhao et al., "A survey of large language models," arXiv preprint arXiv:2303.18223, 2023.

[40] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in vision: A survey," ACM computing surveys (CSUR), vol. 54, no. 10s, pp. 1-41, 2022.

[41] Z. Liu, G. Li, and J. Cheng, "Hardware acceleration of fully quantized bert for efficient natural language processing," in 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2021: IEEE, pp. 513-516.