Accelerator-Aware Computation Offloading under Timing Constraints

Vincent Latzko*, Christian Vielhaus*, Mahshid Mehrabi‡, Frank H. P. Fitzek*§
*Deutsche Telekom Chair of Communication Networks, Technische Universität Dresden, 01062 Dresden, Germany
‡Barkhausen Institute
§Centre for Tactile Internet with Human-in-the-Loop (CeTI), Technische Universität Dresden, 01062 Dresden, Germany

***Abstract*—The rise of chiplets in personal and high performance computing is mirrored in System on Chip (SOC) in mobile devices. Both paradigms allow vendors and designers to integrate dedicated circuitry for accelerating computation. Implementations like cryptographic or vector engines are well known, and nowadays Machine Learning (ML) blocks are often included to accelerate Deep Neural Network (DNN) inference. The shift toward diverse device architectures, as exemplified by RISC-V, is poised to gain momentum. The widespread integration of accelerators in smartphones, tablets, SoCs, and dedicated server systems, is opening up exciting new innovations. In this short paper we present computation offloading for specific workloads in the framework of Multi-Access Edge Computing (MEC) and energy optimisation. We honour inter-task dependency through use of a Directed Acyclic Graph (DAG). Our system model with multiple mobile users, Device-to-Device (D2D) links between User Equipments (UEs), and edge servers enables computational and communication cooperation. The system's energy efficiency is significantly improved by introducing accelerators to the UEs and the MEC. We study the capabilities of the devices (accelerators) and propose an effective solution.**

***Index Terms*—Computation Offloading, Multi-Access Edge Computing, Heterogeneous Computing**

I. INTRODUCTION

Cloud computing has taken the world by storm, and is firmly established in industry. The economies of scale and reduced capital expenditures for renting instances imply that so called *hyperscalers* drive central cloud development, deployment styles, and adoption. Increasingly, use cases demand lower access times than centralised cloud locations can serve [1]. This mainly concerns interactions between humans and machines [2], but also intense workloads on mobile devices. The Multi-Access Edge Computing (MEC) concept addresses these requirements, and provisions computational and storage resources spatially close to the User Equipments (UEs). Essentially, the cloud is moved to the edge of the network [3], commonly into the Base Stations (BSs) or WiFi access points. The recent enhancements in the computing power of mobile devices, particularly in their Central Processing Units (CPUs), present an enticing opportunity to accelerate the implementation of edge computing. Leveraging the resources of nearby devices through Device-to-Device (D2D) connections can meet application demands and enhance overall system performance. This potential becomes particularly noteworthy when applications can be broken down into tasks that can be executed in a distributed manner.

With Augmented and Virtual Reality expected to rise, Machine Learning (ML) workloads will play a major role [4] for processing at the end user. The flexibility and widespread use of Deep Neural Networks (DNNs) motivated industries to invest in ways to ease the cost for inference [5]. The resulting Domain-Specific Architecture (DSA) units are significantly faster, more efficient, and consequently are becoming more widespread [6]. An important use case is also Federated Learning, a technique that immediately benefits from on-device acceleration.

In the literature, [7] and [8] are closest to our works of [9] and [10]. However, their analysis is limited to few (three) nodes in total and only one node has tasks to offload. [11] focus on game theoretical approaches. [12] include D2D concepts from fog computing, and implement an software-defined networking controller for decision making. But synchronising between UEs is not included, the scenario is static, and only one node offloads tasks. [13] and [14] strictly include UEs mobility into the problem. In [15], heterogeneous devices are considered in the scenario. Their optimisation scheme is similar to [9], but the scenario is assumed static while computation takes place.

To address the issues in state-of-the-art studies and moving towards a more modern and complete computation offloading scenario, we propose an energy- and accelerator-aware framework. Its goal is to find optimal computation and communication cooperation offloading strategies for generalised task dependency graphs. Partially, tasks are accelerable. To our knowledge, it is the first multi-user multi-task dynamic computation offloading scenario in a D2D-assisted MEC/accelerator network which minimises the energy consumption of mobile devices, considering the low-latency and dependancy requirements of recent applications.

We continue to specify the System Model in Section II. Next is the problem formulation in Section III, followed by the solver approach and the Genetic Algorithm (GA) for offloading (Section IV), and finally summarise results in Section V.

II. SYSTEM MODEL

We analyse a system as in Figure 1, with N UEs, $\mathcal{N} = \{1, 2, ..., N\}$. Every UE may have an application to execute, which is subdivided into K finite-sized and inter-dependent tasks (colored squares). Because we optimise globally, all tasks are appended into set $\mathcal{K}$. The UEs are placed randomly in the small-cell BS with its attached MEC server (without any clustering assumptions). In our model, the BS as the logically centralised point coordinates D2D and cellular link management, as well as offloading decisions.

UEs may be able to communicate directly depending on the distance $d(\cdot,\cdot)$ between them. D2D communication breaks down at a certain range $R^i > d$, leading to available devices as helpers or as relay nodes

$$\mathcal{H} = \bigcup_{n\in\mathcal{N}} \{i \in \mathcal{N} \ : d(n,i) \leq R_i\}.$$

We operate under the assumption that the wireless channel remains largely stable throughout the transmission and execution phases. Furthermore, we adopt the common assumption that the output from each task is significantly smaller than the input data. This allows us to focus solely on the transmission of task input and execution times. In our

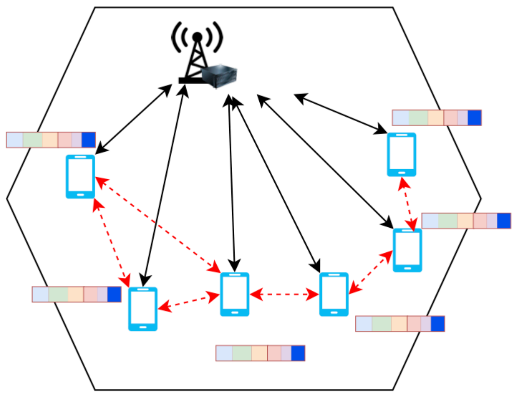

Fig. 1: The system model; Black arrows indicate cellular links, red dashed arrows are D2D. Tasks are squares in sequence.

scenario, the devices are mobile. To predict their paths, we use the learned model PECNet [16]. The MEC server is resposible for obtaining the information about the users' location and their previous paths, which are input to PECNet. The predicted movements for the future are returned. Furthermore, based on the time that users are in the others (UEs and MEC) vicinity, we have the sojourn time concept. For example, the sojourn time of a UE in helper i's coverage from time t is then given as

$$T_{s,i,t} = \max\ m, \text{ s.t. } \mathrm{cov}(\hat{x}_i^\tau)\ \forall\tau \in [t, t+m] \subset \mathbb{Z}.$$

A. Task model

Applications' tasks are modelled using a Directed Acyclic Graph (DAG), with tasks V, edges $e_{i,j}$ and the graph $G = (V, E)$. Nonzero edges $e_{i,j}$ signal a directed relationship: task i is the predecessor of task j (cf. [17]). For each task k of device n, parameters $\{b_{n,k}, c_{n,k}, d_{n,k}\}$ describe: $b_{n,k}$, the *b*itsize of task data, $c_{n,k}$ the *c*ycles of computation needed per bit of task description, and their product $d_{n,k} = b_{n,k} \cdot c_{n,k}$ as the total required computational resources. The deadline T_n^{max} indicates an upper bound for the execution of the whole application of device n.

B. Communication and computation models

In the communication channel model, the uplink data rate is obtained using Shannon's theorem (the downlink is not considered due to the small size of computed tasks [1]):

$$r_{n,k} = B \log_2\left(1 + P_k^{tr} H_k/\sigma^2\right),$$

where P_k^{tr}, is sender's transmission power, B and H_k are the channel bandwidth and channel gain respectively and σ^2 is the variance of the Gaussian channel noise.

Four cases of task execution emerge:

1) Local execution: The computation time model for a task k in this mode is obtained by:

$$T_{n,k}^l = \frac{d_{n,k}}{f_{n,k}^l}, \forall k \in \mathcal{K}, \forall n \in \mathcal{N}.$$

Where, $f_{n,k}^l$ is the dedicated computation resource of UE n for task k. In addition, using the effective switched capacitance (modelled as $\varepsilon = \lambda(f_{n,k}^l)^2$) and depending on the chip architecture [18], the energy is modelled as

$$E_{n,k}^l = \lambda(f_{n,k}^l)^2\, d_{n,k}, \forall k \in \mathcal{K}, \forall n \in \mathcal{N}.$$

2) On Helper execution: The computation time model for a task k of UE n in this mode is obtained by:

$$T_{n,k}^i = \frac{b_{n,k}}{r_{n,k}^i} + \frac{d_{n,k}}{f_{n,k}^i}, \forall k \in \mathcal{K}, \forall n \in \mathcal{N}, \forall i \in \mathcal{H}$$

Where, $f_{n,k}^i$, is the dedicated computation resource of UE i. In addition, the energy is modelled as

$$E_{n,k}^i = P_{n,k}^{tr}\left(\frac{b_{n,k}}{r_{n,k}^i}\right) + \lambda(f_{n,k}^i)^2\, d_{n,k}, \forall k \in \mathcal{K}, \forall n \in \mathcal{N}, \forall i \in \mathcal{H}$$

(where, $P_{n,k}^{tr}$ is the transmission power of UE n).

3) On Edge Cloud execution: The computation time model for a task k in this mode is obtained by:

$$T_{n,k}^s = \frac{b_{n,k}}{r_{n,k}^s} + \frac{d_{n,k}}{f_{n,k}^s}, \forall k \in \mathcal{K}, \forall n \in \mathcal{N}, \forall i \in \mathcal{H}.$$

Where, $f_{n,k}^s$, is the dedicated computation resource of MEC server s for task k's. The energy consumption for for computation is neglected due to the utility grid. Therefore, the energy is obtained as

$$E_{n,k}^s = P_{n,k}^{tra}\left(\frac{b_{n,k}}{r_{n,k}^s}\right), \forall k \in \mathcal{K}, \forall n \in \mathcal{N}, \forall i \in \mathcal{H}.$$

4) On Edge Cloud execution via Relay: The computation time model for a task k in this mode is obtained by

$$T_{n,k}^{i,s} = \frac{b_{n,k}}{r_{n,k}^i} + \frac{b_{n,k}}{r_{n,k}^{i,s}} + \frac{d_{n,k}}{f_{n,k}^s}$$

In addition, the energy is obtained by

$$E_{n,k}^{i,s} = P_{n,k}^{tra}\left(\frac{b_{n,k}}{r_{n,k}^s}\right) + P_{n,k}^{tra,i}\left(\frac{b_{n,k}}{r_{n,k}^s}\right),$$

both $\forall k \in \mathcal{K}, \forall n \in \mathcal{N}, \forall i \in \mathcal{H}$.

C. Accelerator Model

Typically, not all parts of an application consist of ML workloads. Consequently, tasks that can be accelerated receive an indicator (binary) flag $f_{n,k}$. These tasks can then be mapped to accelerators on devices equiped with such, and then be executed faster and more efficiently. We introduce accelerator parameters $a_p > 1$ for faster calculation, and $a_e > 1$ for efficiency gains. This results in directly scaling the execution times on devices with accelerators, e.g. $T^l_{n,k} = \frac{d_{n,k}}{f^l_{n,k} a_p}$ in the case of local computing ($a_p > 1$ ensures faster execution). For the energy terms, an equivalent modification takes place, i.e. $E^l_{n,k} = \lambda \left(f^l_{n,k}\right)^2 \frac{d_{n,k}}{a_e}$, for local computing ($a_e > 1$ ensures more efficiency).

III. Problem Formulation

According to [19], there are two important definitions for task dependencies: **Ready time:** The earliest time that the execution of a task k can get started (all predecessors completed). **Finish time:** The final time slot occupied by execution of k. The ready time and finish time of task k of UE n can be defined depending on the offloading modes. In calculation of each the following factors should be considered: The completion time of the execution of predecessors for task k, the availability time of the user which tasks is executed on as well as the time that the D2D links are free.

A. Optimization formulation

For any given task k of device n, the total energy effort is precisely determined by the offloading strategy, i.e., $E_{n,k} = \ell^l_{n,k} E^l_{n,k} + \ell^i_{n,k} E^i_{n,k} + \ell^{i,s}_{n,k} E^{i,s}_{n,k} + \ell^s_{n,k} E^s_{n,k}$. Consequently, the goal is to find a global minimum of energy expenditure over all tasks of all devices, and a minimum of execution time. The optimization problem is formulated as follows:

$$\begin{aligned}
\min_{\boldsymbol{d}} \quad & \sum_{n=1}^{N}\sum_{k=1}^{K} E_{n,k} \\
s.t. \quad & \forall k \in \mathcal{K}, i \in \mathcal{N}, m \in \mathcal{N} \\
& C1: \quad \ell^l_{n,k}, \ell^i_{n,k}, \ell^{i,s}_{n,k}, \ell^s_{n,k} \in \{0,1\} \\
& C2: \quad \ell^l_{n,k} + \sum_{i=1}^{I} \left(\ell^i_{n,k} + \ell^{i,s}_{n,k}\right) + \ell^s_{n,k} = 1 \\
& C3: \quad \ell^l_{n,k} + \sum_{n \in \mathcal{N}-i} \left(\ell^n_{i,k} + \ell^{i,n}_{i,k}\right) \le 1 \\
& C4: \quad \max\{FT_{n,p}\} \le ST_{n,k} \\
& C5: \quad FT_{n,K} \le T^{max}_n \\
& C6: \quad FT_{n,k} \le T_{i,s}, \quad if \;\; i \ne m
\end{aligned}$$

wherein $\mathbf{d}$ is the offloading location,

$$\mathbf{d} = \Big[\ell^l_{1,1}, \ell^1_{1,1}, \dots\dots, \ell^I_{1,1}, \ell^{1,s}_{1,1}, \dots, \ell^{I,s}_{1,1}, \ell^s_{1,1}, \ell^l_{1,2}, \ell^1_{1,2}, \dots, \ell^s_{N,K}\Big]^T.$$

Here, C1 present the fact that only one destination per task is possible for offloading, C2 assures each task is executed exactly once. C3 guarantees any device can only compute locally, or serve as helper or relay for another device i. C4 honors the order of tasks. C5 is related to the task dependencies while C6 prohibits a decision that violates any deadline constraint. C7 shows that task k execution must be finished before it leaves the coverage area.

IV. Accelerator-aware Computation Offloading Problem

The optimisation problem in the previous section is a mixed integer nonlinear programming type. In order to solve such a problem, we employ a hybrid Genetic-PSO Algorithm. Holland [20] was the first to introduce a new algorithm called GA. This algorithm contains some candidate solutions, namely chromosomes for a problem. Each chromosome contains some smaller parts (genes), which are binary offloading decision variables in our proposed method. Kennedy [21] introduced the Particle Swarm Optimisation (PSO). The main process in this algorithm is updating the velocity and location of some particles to obtain the optimal value. In the Hybrid Genetic-PSO algorithm, which is proposed in [22], a population of the best chromosomes with the highest fitness values is chosen and are further optimised by the PSO algorithm. This significantly decreases the convergence time of our proposed offloading algorithm. Following equations show the velocity and location of a particle which reduces the complexity of the hybrid algorithm:

$$v_i(t+1) = d_1 f_1[g(t) - x_i(t)],$$

$$x_i(t+1) = x_i(t) + v_i(t+1).$$

v_i is the velocity of particle i, d_1 is the learning factors between 0 and 2, f_1 is a random number between 0 and 1, x_i is the current position of particle i and g is the global best.

Since the goal of our computation offloading algorithm is minimising the total energy usage of the system [17], [23], [24], our fitness function (with the constraints as additive penalties) is defined as follows:

$$f_\lambda = \sum_{n=0}^{N-1}\sum_{k=0}^{K-1} E_{n,k} + \theta\Big[\max\{0, FT_{n,K} - T^{\max}_n\} + \max\{0, \left(FT^i_{n,k} - T_{i,s}\right)\} + \max\{0, (FT^{i,s}_{n,k} - T_{i,s})\}\Big].$$

Here, θ is a very large number. The chromosome population is sorted based on their fitness value and the top 40% are used in PSO. This increases the convergence speed of the algorithm significantly. The Tournament selection method is applied to the other 40% of the sorted chromosomes and the rest of the next generation chromosomes are prepared with mutation. The worst chromosomes get replaced by random chromosomes.

The mutation is enacted with probability p_m, where the worst chromosome is completely replaced with random parameters and the process is iterated N times.

V. Numerical Results

We mainly mirror prior approaches for evaluation. Essentially, we compare the scenarios for their total effort based on the fitness-values:

- **All local:** All tasks are executed on their UEs;
- **All server:** All tasks are completely offloaded to the edge;
- **Computation cooperation:** UEs can act as helper nodes, executing incoming tasks;
- **Communication cooperation:** UEs can act as relays, transmitting tasks to the edge for execution on the MEC;
- **Accelerated cooperation:** Full flexibility with acceleration consideration on N_{acc} out of N of the UEs;
- **Accelerated MEC:** Full flexibility, but only the BS is equipped with an accelerator.

We specify all used[1] parameters in Table I. To exclude statistical flukes and random noise, we average the scores for all experiments over 50 runs. In Figure 2, higher computational load of tasks leads to increased margin of our approach. The accelerated scheme beats all baselines.

TABLE I: Simulation parameters

Parameters	Value
Number of tasks of each user (K)	10
Number of users in the network (N)	6
Number of users with accelerators (N_{acc})	3
Task deadline (T_d)	4.5 s
Data size of task k (b_k)	$200-500$ kb
Required CPU cycles per bit of task k (c_k)	110 cycles/bit
Channel bandwidth (B)	5 MHz
Channel gain (UE n to UE i) ($\mathrm{H}_{k,h}$)	$1-1.5\times10^{-2}$
Channel gain (UE n to MEC) ($\mathrm{H}_{k,s}$)	$1-1.5\times10^{-5}$
Variance of the Gaussian channel noise (σ^2)	10^{-9}
Transmission power of UEs (P_{tra}^{UE})	$0.1-0.15$ mW
CPU cycles frequency of UEs (f_{UE})	$4-7\times10^8$ cycles/s
Maximum CPU frequency of MEC (f_S^{max})	10×10^9 cycles/s
Effective switched capacitance (λ)	10^{-27} F
Area considered for UE mobility	500×500 m^2
D2D range	100 m
Accelerator performance a_p	5
Accelerator efficiency a_e	5
Accelerable workload percentage $f_{n,k}$	60
Number of GA iterations	150
Population size of GA	30
Population size of PSO	12
Crossover probability p_c	0.5
Mutation probability p_m	0.05

Next, we change the data size from 0 to 375 kB (six times larger than [14]), to stress the communication aspect. Consequently, more energy needs to be utilised for eventual offloading. This additionally increases the scores of all methods, as expected. Our framework even increases its lead, which is shown in Figure 3.

We finally analyse a situation where telecommunications providers are looking to differentiate their offerings. We inspect the impact of added dedicated accelerators to the servers. We model the MEC server as before, but now equip it with an accelerator. As a comparison, we also inspect the impact of doubling its CPU, i.e. general purpose, capabilities. When we sweep the computational complexity (c_k), we see a clear ranking, cf. Table II. Deploying accelerators should be preferred, as it clearly has larger impact. Larger accelerator deployments are also clearly a path that should be explored in this situation. This mirrors findings from [6].

[1]Our code will be made public on the authors' GitHub

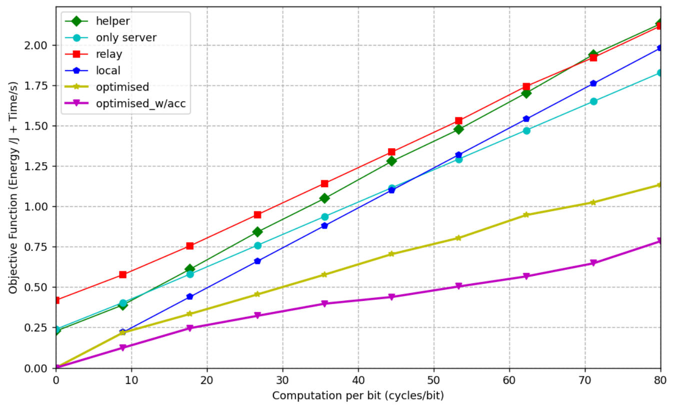

Fig. 2: Average objective function values versus computational demands.

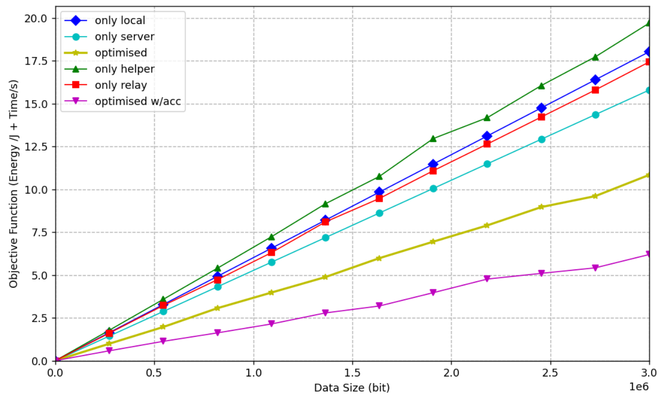

Fig. 3: Average energy and time expended versus data size.

TABLE II: Objective function values for sweeped complexity for different deployment strategies for the edge server.

Complexity	8	18	26	37	62	71	80
MEC	0.13	0.25	0.33	0.4	0.57	0.65	0.78
MEC double CPU	0.12	0.21	0.26	0.31	0.48	0.52	0.59
MEC accelerator	0.13	0.18	0.23	0.26	0.35	0.39	0.42

Finally, we present Figure 4 in a bid to strengthen our confidence in the robustness of the results. Our method converges fairly quickly for three exemplary scenarios, and for accelerated cases, converges slightly faster. It shows the effectiveness of the chosen GA and its applicability for obtaining results.

VI. Conclusion & Future Works DONE

We studied the optimisation of a computation offloading decision algorithm in a scenario with multiple devices,

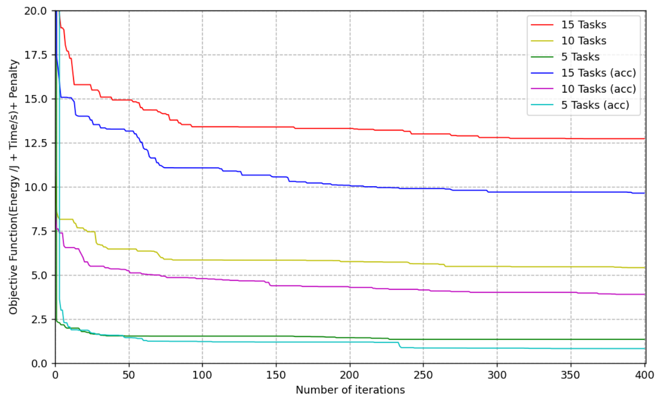

Fig. 4: Fitness function versus number of iterations show the convergence of select experiments.

multiple tasks, a MEC system, time deadlines, task interdependency, and device heterogeneity. For the first time, dedicated accelerator chips for ML workloads were modelled in this context. We turned to a genetic algorithm to realise the target of optimised UE battery expenditure under constraints of latency, mobility and cooperation. Simulation results indicate the usefulness of taking accelerators into consideration. The accelerator-aware solver achieves best performance. We differentiate the results along multiple axes, highlighting a pareto-optimal solution by our algorithm, i.e., no performance degradation in any case. For accelerable workloads, edge deployments benefit greatly from DSA enhancements.

Future steps could tackle the scenario to directly *learn* a heuristic as replacement for the solver. From a modelling perspective, frequency reuse and more efficient bandwidth usage may be explored.

Acknowledgment

This research has been supported in part by the European Union, ECSEL JU Project AI4CSM, the German Ministry of Education and Research (BMBF, FKZ 16MEE0173), and 6G-life (BMBF FKZ 16KISK001K), and by the German Research Foundation (DFG, Deutsche Forschungsgemeinschaft) as part of Germany's Excellence Strategy - EXC 2050/1 - Project ID 390696704 - Cluster of Excellence Centre for Tactile Internet with Human-in-the-Loop (CeTI) of Technische Universität Dresden.

References

[1] M. Mehrabi, D. You, V. Latzko, H. Salah, M. Reisslein, and F. H. P. Fitzek, "Device-enhanced mec: Multi-access edge computing (mec) aided by end device computation and caching: A survey," *IEEE Access*, 2019.

[2] G. P. Fettweis, "The tactile internet: Applications and challenges," *IEEE Vehicular Technology Magazine*, vol. 9, no. 1, pp. 64–70, 2014.

[3] S. Guan and A. Boukerche, "A novel mobility-aware offloading management scheme in sustainable multi-access edge computing," *IEEE Transactions on Sustainable Computing*, 2021.

[4] T. Zhao, Y. Xie, Y. Wang, J. Cheng, X. Guo, B. Hu, and Y. Chen, "A survey of deep learning on mobile devices: Applications, optimizations, challenges, and research opportunities," *Proceedings of the IEEE*, vol. 110, no. 3, pp. 334–354, 2022.

[5] Y.-H. Chen, R. Sarokin, J. Lee, J. Tang, C.-L. Chang, A. Kulik, and M. Grundmann, "Speed is all you need: On-device acceleration of large diffusion models via gpu-aware optimizations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 4650–4654.

[6] N. Jouppi, C. Young, N. Patil, and D. Patterson, "Motivation for and evaluation of the first tensor processing unit," *IEEE Micro*, vol. 38, no. 3, 2018.

[7] X. Cao, F. Wang, J. Xu, R. Zhang, and S. Cui, "Joint computation and communication cooperation for mobile edge computing," in *2018 16th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*. IEEE, 2018, pp. 1–6.

[8] G. Qiao, S. Leng, and Y. Zhang, "Online learning and optimization for computation offloading in D2D edge computing and networks," *Mobile Networks and Applications*, pp. 1–12, 2019.

[9] M. Mehrabi, S. Shen, V. Latzko, Y. Wang, and F. H. Fitzek, "Energy-aware cooperative offloading framework for inter-dependent and delay-sensitive tasks," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE, 2020, pp. 1–6.

[10] M. Mehrabi, S. Shen, Y. Hai, V. Latzko, G. P. Koudouridis, X. Gelabert, M. Reisslein, and F. H. P. Fitzek, "Mobility- and energy-aware cooperative edge offloading for dependent computation tasks," *Network*, vol. 1, no. 2, pp. 191–214, 2021.

[11] D. Wang, Y. Lan, T. Zhao, Z. Yin, and X. Wang, "On the design of computation offloading in cache-aided D2D multicast networks," *IEEE Access*, vol. 6, pp. 63 426–63 441, 2018.

[12] Q. Jia, R. Xie, Q. Tang, X. Li, T. Huang, J. Liu, and Y. Liu, "Energy-efficient computation offloading in 5g cellular networks with edge computing and d2d communications," *IET Communications*, vol. 13, no. 8, pp. 1122–1130, 2019.

[13] M. Mehrabi, "Computing on the edge of the network," 2022.

[14] V. Latzko, O. Lhamo, M. Mehrabi, C. Vielhaus, and F. H. P. Fitzek, "Energy-aware and fair multi-user multi-task computation offloading," in *2023 International Conference on Computing, Networking and Communications (ICNC)*, 2023, pp. 231–236.

[15] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-efficient offloading for mobile edge computing in 5g heterogeneous networks," *IEEE Access*, vol. 4, 2016.

[16] K. Mangalam, H. Girase, S. Agarwal, K.-H. Lee, E. Adeli, J. Malik, and A. Gaidon, "It is not the journey but the destination: Endpoint conditioned trajectory prediction," in *European Conference on Computer Vision*. Springer, 2020, pp. 759–776.

[17] S. Guo, J. Liu, Y. Yang, B. Xiao, and Z. Li, "Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing," *IEEE Transactions on Mobile Computing*, vol. 18, no. 2, pp. 319–333, 2018.

[18] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-Optimal Mobile Cloud Computing under Stochastic Wireless Channel," *IEEE Transactions on Wireless Communications*, vol. 12, no. 9, pp. 4569–4581, 2013.

[19] T. Yang, R. Chai, L. Zhang, and Q. Chen, "Worst case latency optimization-based joint computation offloading and scheduling for interdependent subtasks," in *2020 International Conference on Wireless Communications and Signal Processing (WCSP)*. IEEE, 2020.

[20] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[21] R. Eberhart and J. Kennedy, "Particle swarm optimization," in *Proceedings of the IEEE international conference on neural networks*, vol. 4. Citeseer, 1995, pp. 1942–1948.

[22] C.-F. Lu and C.-F. Juang, "Evolutionary fuzzy control of flexible ac transmission system," *IEE Proceedings-Generation, Transmission and Distribution*, vol. 152, no. 4, pp. 441–448, 2005.

[23] U. Saleem, Y. Liu, S. Jangsher, Y. Li, and T. Jiang, "Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing," *IEEE Transactions on Wireless Communications*, 2020.

[24] W. Fan, Y. Liu, B. Tang, F. Wu, and H. Zhang, "Exploiting Joint Computation Offloading and Data Caching to Enhance Mobile Terminal Performance," Dec. 2016, pp. 1–6.