IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication 2022 00, 0000, date of current version 2022 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2022.0092316

On Applicability of Imagery-based CNN to Computational Offloading Location Selection

ALEKSANDR OMETOV[1](Senior Member, IEEE), ANZHELIKA MEZINA[2], and JARI NURMI[1](Senior Member, IEEE)

[1]Electrical Engineering Unit, Faculty of Information Technology and Communication Sciences, Tampere University, Finland
[2]Brno University of Technology, Faculty of Electrical Engineering and Communications, Dept. of Telecommunications, Technicka 12, Brno, Czech Republic

Corresponding author: Aleksandr Ometov (e-mail: aleksandr.ometov@tuni.fi).

This work was supported by Ministry of the Interior of the Czech Republic, no. VK01010153, VJ02010019.

ABSTRACT The progress in computational offloading is heavily pushing the development of the modern Information and Communications Technology domain. The growth in resource-constrained Internet of Things devices demands the development of new computational offloading strategies to be sustainably integrated in beyond 5G networks. One of the solutions to said demand is enabling Mobile Edge Computing (MEC) powered by advanced methods of Machine Learning (ML). This paper proposes the application of ML-powered computational offloading strategy in a wireless cellular network by applying the traditional fundamental Travelling Salesman Problem (TSP) on computational offloading location selection. The main specificity of the proposed approach is the use of imagery data. Thus, the paper executes a literature review to identify existing strategies. It further proposes a novel method utilizing the location-like imagery data to identify the most suitable computational location by executing the search for an identified route between locations using the proposed Deep Learning (DL) model. The model was evaluated and achieved MAE – 1,575, MSE – 10 119 205, R^2 – 0.98 on the testing dataset, which outperforms or is comparable with other well-known architectures. Moreover, the training time is proven to be 2-10 times faster. Interestingly, the MAE values are relatively low compared to the target values that should be predicted (despite rather high MSE results), which is confirmed by the almost perfect R^2 value. It is concluded that the proposed neural network can predict the target values, and this solution can be applied to real-world tasks.

INDEX TERMS Computational offloading, Machine Learning, Mobile Edge Computing, Optimization

I. INTRODUCTION

At the active stage of introducing (beyond) 5G technology, it is necessary to understand exactly how the computational offloading processes would be arranged in the next generation networks [1]. Proper use of computational and communication resources should become one of the main principles in designing new network architecture. Simultaneously, these principles should not contradict the existing ones: energy efficiency, flexibility, high speed of information transfer, integration with the Internet of Things (IoT), etc. [2].

One of the ways to provide support for said dynamics and handle the growing demands while decreasing the overall system complexity is to move the network resources closer to the devices, e.g., to apply MEC for computational offloading and/or caching, depending on the task [3], [4]. In MEC, this proximity is usually measured in round-trip latency, so MEC locations are often ten milliseconds or less from the data source compared to public Cloud data centers, which can be more than one hundred milliseconds away [5]. Here, the main advantage of MEC is the emergence of additional opportunities for personalization, reducing latency, improving the user-experienced throughput, and ensuring higher energy efficiency of the end nodes for a very broad range of applications [6].

Especially important, the MEC-enabled topology is expected to be heavily dynamic [7]. The fact that each user, through their device or the set of smart devices forming wearable networks [8], will further dynamically connect to the infrastructure network, which, in turn, aims at providing the best Quality of Service (QoS), and is expected to funda-

mentally change the observed network operation in terms of computational offloading. Various other IoT devices, namely, autonomous vehicles, drones, etc., are also expected to utilize computational offloading for their tasks' execution [9]. The need to correctly model the resulting network dynamics will inevitably lead to the need for new complex analytical models for the data management [10], which may become scalably-problematic as the amounts of traffic and uses grow [11].

By their very nature, the State-of-the-Art (SOTA) data processing approaches applied on MEC sometimes fail to capture the patterns found in the real system. Simultaneously, using human resources to process/control information should be, in principle, excluded as an option due to the large number of factors leading to gross errors, which are naturally related to routing or offloading location selection. For example, the tasks from the discrete optimization field (e.g., TSP) could be solved with heuristics since the quality of the solution is greatly reduced with an increase in the number of discrete values [12]. Statistical approaches, in this example, will work only while the data distribution remains stable. Therefore, ML and Artificial Intelligence (AI) could be suitable solutions to this scalability problem.

Today, the field of AI/ML is one of the most attractive technological developments in the Information and Communications Technology (ICT) domain mainly due to the growth in computational capacity and introduction of new mathematical methods [13]. BBC estimated the global ML market was worth $1.4 billion in 2017 and is estimated to reach $8.8 billion by the end of 2022, growing at a Compound Annual Growth Rate of 43.6% between 2017 and 2022 [14]. The cornerstone of this growth is mathematical algorithms that allow extracting useful information from a large amount of data is a root of ML.

Generally, ML methods in the MEC application domain may be divided into Supervised Learning (SL) and Unsupervised Learning (UL) [15]. The principle of building any traditional SL model is related to the empirical data accumulated over a certain period. In its simplest form, such a dataset is divided into training and testing sets. In the first stage, the model attempts to find and learn the patterns in the data using the training set. In the second stage, the performance of the trained model is evaluated on a testing set using different metrics. For example, the classification methods, one of the most widely used ones, are SL for modeling or predicting discrete values.

In contrast, UL is applied when it is impossible or extremely difficult to obtain labeled samples [16]. The goal is to model the latent or underlying structure of the distribution in the data, e.g., by Clustering. The Clustering problem can be formulated as organizing objects into groups according to the rules the algorithm generates from the data. An example of such a task could be the Clustering of Base Stations (BSs) in several locations according to various characteristics or to create distributed MEC clusters within the operator infrastructure.

To contribute to the body of knowledge, this work looks deeper into the ML algorithms applied to the offloading location selection within MEC. By these means, we attempt to understand how the predictive ability of various network characteristics could be increased, allowing the managed resources to potentially be utilized more flexibly in future networks. The sole use of mathematical methods embedded in predictive models only partially improves the system operation significantly, while embedding those into the hypervisor of the MEC system may provide slightly better results.

Specifically, this work focuses on studying and developing DL methods, which can be applied in managing resources and deployed to MEC systems. This work aims to find a (sub-)optimal desired parameter in the given network, especially when the information about links/deployment is of low quality, e.g., during network planning in developing countries. Instead of processing 1D data, for example, vectors of different parameters, 2D data are utilized (graphical map). The proposed solution is based on the Convolutional Neural Network (CNN), which predicts the optimal length from given 2D data.

The main contributions of this paper are:

- We transfer the problem of prediction of desired parameters for MEC onto the processing of location-related 2D data;
- We propose a CNN capable of solving the mentioned regression problem;
- We prove that the proposed model is less time consuming but can outperform the well-known architectures;
- We prove that imaging-based TSP could be applied to the MEC problem and define the architecture that could utilize this approach.

To the best of the authors' knowledge, the literature lacks work that applies a DL model over 2D data for MEC optimization. This paper shows the results of the ML application experiment over imagery data to predict the optimal target value. Considering that there are no datasets for MEC or communication networks with 2D data, we have utilized the publicly available dataset for "Traveling Salesman Computer Vision" [17]. The approach could be further used in any network where the network planning information is available in GEO-like data.

The rest of the paper is organized as follows. Section II provides general information on the use of ML for solving network-related optimization problems. Section III extends the background information with a specific focus on imagery data. Next, Section IV proposes an ML model to solve the offloading location selection problem. Further, Section V outlines selected numerical results. Potential future integration aspects of a practical infrastructure-based system are given in Section VI. The last section concludes the paper.

II. BACKGROUND ON TRAFFIC OPTIMIZATION WITH ML FOR MEC

The introduction of MEC and the joint use of ML algorithms is hampered by two main factors: resource limitations and network dynamics. Network resources (e.g., wireless band/bandwidth) and the computational budget of Edge nodes (e.g., computational capabilities and energy dependency). Edge nodes can be cellular BSs, WiFi Access Points (APs), or mobile energy-independent nodes located in different geographic locations (not necessarily on the BSs in contrast to the conventional Edge computing model). They may dynamically join or leave the system, and the wireless connections and their state may change for various reasons. Hence, the network topology may frequently change during the model training, which may be a significant limitation for traditional analytical models. The following set of literature highlights existing works executed closer to this domain, while the next section specifically focuses on the imagery-related specifics.

In [18], the authors primarily optimize the network topology through decentralized Peer-to-Peer (P2P) communications. Given a set of MEC nodes, they seek to build a topology by choosing wireless channels to ensure the convergence of the model's learning with their custom algorithm under resource constraints. The work describes the proposed Reinforcement Learning (RL) algorithm, which is evaluated on Fashion-MNIST and CIFAR-10 datasets using the following metrics: accuracy, loss, the cost of communication, and the training time of the models. The work highlights that the highest quality (in terms of all indicators) provides PSGD and L2PL algorithms; the L2PL algorithm learns a little slower than D-PSGD by about 12.5%; The convergence of the L2PL and PSDG algorithms is statistically the same.

From the MEC perspective, the devices could be divided into clusters based on their computational budget [19]. The work states that a particular model is better suited for a specific *type* of device momentarily on a lower level or by branching. Then, the data from underlying models could be transferred to the general model, employing Federated Learning (FL), where the final clustering decision could be made. At the same time, each of the underlying models can exchange information with each other, thereby improving the overall system performance. The idea is similar to the hierarchical clustering algorithm, where clusters are formed in a tree or hierarchy. Each node in the tree represents a different cluster, and the clusters in the hierarchy are known as dendrograms.

Similarly, the authors of [20] give a specific example of using FL to optimize work with Edge devices without reproducing the concept of Distributed Learning. Instead, they consider the homogeneity of the common server where the model is trained. In the FL process, the global model is shared with a server that is not heterogeneous. Each client creates a local model by examining their data and the global model and sends an updated version back to the central server for merging.

Due to the heterogeneous nature of the system, this diversity can create several challenges for resource-constrained nodes while timely executing ML tasks and, thus, can cause a delay in overall learning progress. To address this issue, an approach considering multiple global and local models based on available client resources is proposed in [20]. It aims at minimizing the global model's convergence time by collecting the nodes' computational power updates for building a more reliable system, i.e., low-power devices have a higher learning delay than high-power devices that have already learned. In addition, it optimizes the learning process and the process of application of algorithms to new data.

Traditionally, training an AI model requires considerable computing resources and can only be supported on powerful Cloud servers [21] or, potentially, on emerging MEC nodes clustered under the Fog umbrella [22]. From the model retraining perspective, it is a rather computationally-intensive process as it is necessary to allocate considerable computing power to train a new model, as well as to collect enough labeled data with the new scenarios is essential, which can also be time-consuming.

The solution to this problem may be a self-learning model architecture based on a Generative Adversarial Network (GAN) [21]. Multiple generators are trained to produce synthetic data that can capture the traffic data distribution generated by multiple services at different locations within the network coverage area. The authors introduce the so-called Rand index to assess the quality of the work of a self-learning architecture, adapting to dynamic vehicular scenarios. For comparison, the simplest k-means algorithm was used to demonstrate that the quality of the GAN model, depending on the sample size, is better, on average, by 24%.

Liu et al. [23] also attempted to solve the network offloading problem to optimize the overall network load distribution. First, the authors demonstrated the architectural scheme, which includes a set of Edge macro BS, micro BS, Remote Radio Node (RRN), Roadside Unit (RSU), etc., and one dedicated Edge server connected to a hub by a high-bandwidth backhaul link or is located directly to the hub. The model is a relatively simple feedforward neural network architecture, and the dataset is split into three parts for validation reasons – training (80% of data), validation for selecting model hyperparameters (15% of data), and testing for evaluating results (5% of data). The simulations showed that the computational efficiency increased from 93% to 96%.

Similarly, the authors of [24] developed a cost-effective coordination strategy to maximize the average learning utility with limited resources mapped to the budget-limited multi-armed bandit problem by searching for optimal hand sequences. The specific goal was to maximize the average reward per hand while keeping the total cost of the hand below-given budget. This statistical approach utilized fairly fast ML algorithms: k-means and Support Vector Machine (SVM), thus, maintaining resource use efficiency while increasing the accuracy of predictions by 12%, compared with existing solutions.

To summarize, the methods for working with computational offloading of significant data volumes are fundamental in forming the future architecture of MEC-powered networks. It is primarily due to the amount of data constantly increasing as the number of new devices generating this data also grows. In addition, the more recent generations of networks are becoming more user-centric. Therefore, ML becomes more actual in this technology field. However, the outlook of using ML from an imagery perspective in MEC scenarios remains an open task.

III. PROBLEM FORMULATION SCOPE

As a separate subdomain, image processing has already been applied in different ICT-related research directions, e.g., classification, object detection, segmentation, etc., as well as in the communication network research field. For example, the authors of [25] proposed an image-based signal strength prediction method utilizing environmental images as the input to their DL method.

Moreover, some approaches for path loss prediction were proposed in other works [26], [27]. 2D data was also applied in traffic control, as presented in work [28]. The authors proposed the DL model based on CNN for intelligent network traffic control. The main motivation is that existing traffic control methods could have been more effective since those lack a closed loop, i.e., do not learn from their previous experiences and abnormalities.

One of the fundamental networking problems currently solved with imagery ML is related to TSP, which has many variations that are directly used to solve real problems in abstract networks. The authors of [29] were studying it from Dubins Vehicle perspective, which is similar to the classical TSP and, in particular, to the Euclidean TSP (ETSP), where the shortest path between any two target locations is a straight line. The practical motivation for learning ETSP naturally arises in robotic and drone applications.

Overall, genetic algorithms can also be used to solve the TSP problem. With this respect, the work [30] investigates the application of genetic algorithms to TSP by examining combinations of different algorithms for binary and unary operators to obtain better solutions and minimize the search space. Three binary and two unary operations were tested in this work.

Boffa et al. [31] also demonstrated a similar approach with Graph Neural Network (GNN). The key idea is to use the GNN architecture to solve the Power and Channel Allocation Problem (PCAP), which is of practical importance for the allocation of radio resources in wireless networks. Experimental results show that existing architectures are still unable to capture the structural features of a graph and are not suitable for tasks where actions on a graph directly impact the attributes of the graph itself. Although the authors' results do not exceed the existing solutions in terms of metrics, it is important to note that this approach is very promising. In essence, the TSP data is presented as dependencies, and applying ML algorithms directly related to finding a structure in graphs appears logical. Using this architecture, the authors showed the effectiveness of the distance coding method for the multipurpose optimization of model parameters, which also served as our motivation to apply it to the MEC domain.

In summary, ML plays a significant role in optimizing MEC operation in cellular networks and can be further utilized for different tasks than the ones listed above. Notably, FL and Reinforced FL are some of the most frequently used methods of ML in this field of research. Another gap in this field is a lack of datasets for ML, which would allow to experiment with different models and compare results with other approaches, which is a reason why some works utilize similar, yet imagery-based, datasets to solve this task [18], [20], [32], [33]. Therefore, pushing our research one step closer to more efficient MEC operation.

IV. METHODOLOGY

In this section, we propose solving the NP-hard problem using DL by considering its connection with MEC and mobile networks. This section covers the problem definition, data processing, and description of the proposed neural network.

A. THE PROBLEM DEFINITION

The aim of this experiment is to propose a DL model for the prediction of the defined value, in our case, an optimal length of the path in the given depicted network. All stages of the model development are presented below (see Figure 1).

The problem is solved on the "Traveling Salesman Computer Vision" dataset [17] and has a rather straightforward analogy with the task of optimizing Edge Computing. We can draw a parallel between cities and the distances between them in the form of the Edge BS and the same distance between them, for example, latency.

Formulating the problem in terms of ML, we arrive at:

- Features (input to DL model) – map images with MEC Nodes and routes between them (2D data);
- Target (output of DL model) – the optimal distance between MEC Nodes (1 numeric value);
- Metrics – Mean Absolute Error (MAE), Mean Squared Error (MSE), and R^2.

The model development process consists of a common set of steps for most ML tasks: data preparation, data analysis, model development, training process, and evaluation.

B. DATA DESCRIPTION

The dataset was taken from the open resource Kaggle "Traveling Salesman Computer Vision" [17]. Note, it is worth clarifying that **this particular task does not aim to find the shortest route between cities but to find the length of the best (in terms of minimum distance) route marked on the map**. Although this is not a TSP task in its purest form, reducing it to its classic version is straightforward if needed.

The imagery data in the dataset is a randomly generated set of graphical maps with various attributes: size, number of nodes, and optimal connections. Figure 2 shows a map with

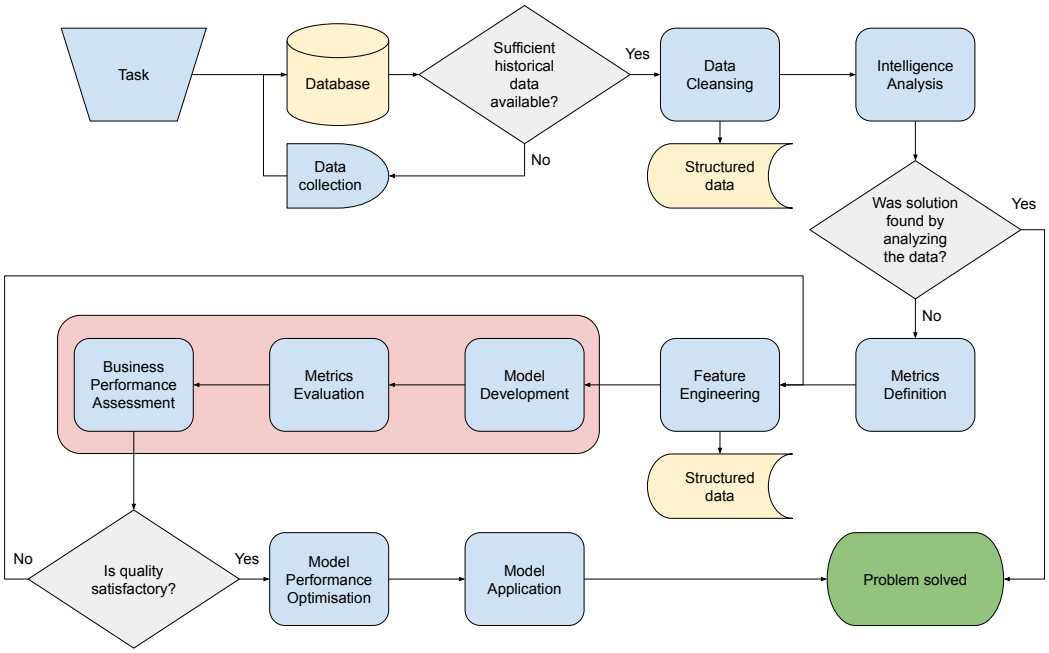

FIGURE 1. The algorithm of model development.

several cities and one best route. However, some maps are larger and contain multiple routes, e.g., in Figure 3.

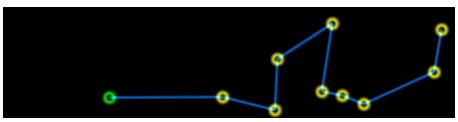

FIGURE 2. Example of a small map from a dataset.

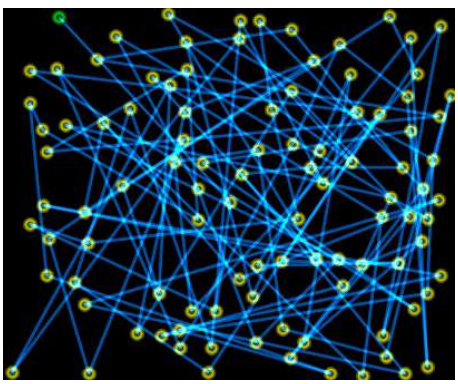

FIGURE 3. Example of a large map from a dataset.

In addition to images, there are tabular data (see Table 1), which describe the maps – the number of nodes, distance, and the size of the image.

Image processing in DL is a computationally expensive process. In this regard, the training of the model occurs iteratively. At each iteration (step in epoch), a defined number of images are fed into the model using tensors (batch).

TABLE 1. Example preprocessed tabular data.

id	key	filename	distance	height	width	nodes
12641	971x292-281-6677.jpg	12641.jpg	6677	971	292	281
7294	923x353-104-6808.jpg	7294.jpg	6808	923	353	104
15600	786x429-395-10261.jpg	15600.jpg	10261	786	429	395
9390	425x599-130-36703.jpg	9390.jpg	36703	425	599	130
10051	692x889-29-3623.jpg	10051.jpg	3623	692	889	29

As described previously, the entire data set should be divided into several phases for a correct assessment of the quality of the model: training, validation, and testing. In this dataset, a testing one is provided separately. Data selection for validation occurs randomly from the training set in a percentage ratio of 35 to 65. The size of the resulting number of samples in each set is as follows: Training set – $10,411$ images; Validation set – $5,607$ images; Testing set – $4,005$ images.

The sizes of the images fed to the input of the model were studied for all samples, see Figure 4 and 5. The distribution of the target variable is shown in Figure 6.

The statistics obtained on the training, validation, and testing sets regards to the route length are represented in Table 2.

TABLE 2. The statistics over training, validation, and testing sets.

	Training	Validation	Testing
Median	9,512	9,519	9,603
Average	17,684	17,931	17,856
Dispersion	25,147	26,165	25,381

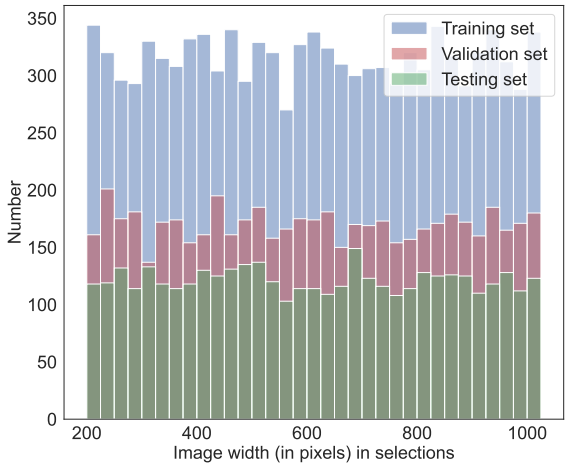

FIGURE 4. Statistics over image width.

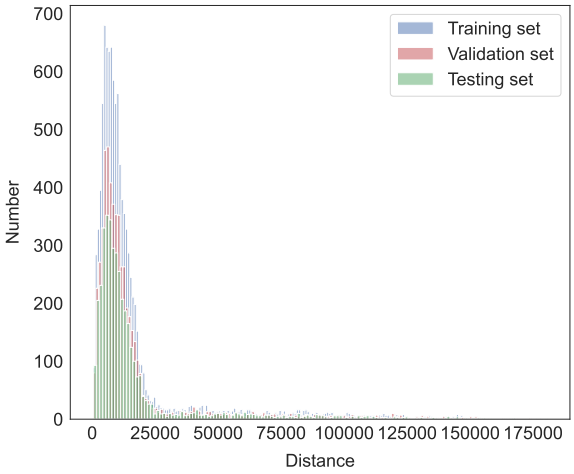

FIGURE 6. Distribution of the target value on different samples.

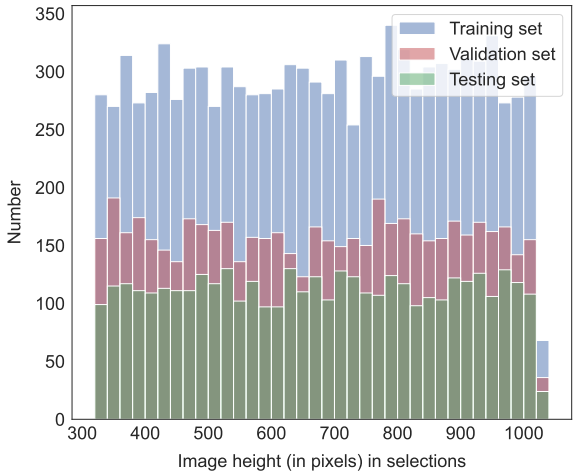

FIGURE 5. Statistics over image height.

C. BASELINE

Since there is no approach to solving this problem for the imagery dataset (described in subsection IV-B), several well-known SOTA architectures of neural networks were used for comparison with the proposed method. For this purpose, the following architectures were selected: DenseNet121 [34], ResNet34 [35], and VGG16 [36] as architectures that are frequently used in computer vision tasks and there is an opportunity to use pretrained models.

In this experiment, the mentioned models were utilized as the backbone, meaning that the existing pretrained model was used. Application of the backbone allows to extract features from the input and to prepare it for the oncoming processing and the target value prediction. After feature extraction, the fully connected network is added to predict the target value. This part is represented with a flattened layer and 3 linear with ReLU activation function (a more detailed description of these layers is introduced in subsection IV-D). The training parameters are: a loss function is MSE, an optimizer is Adam with a learning rate 0.001, and a number of epochs is 10.

D. PROPOSED DEEP LEARNING MODEL

To find the best route's length, a CNN was developed consisting of two convolutional layers, five layers with ReLU activation functions, two max-pooling layers, one flatten layer, and four linear layers. The complete network architecture is shown in Figure 7. The programming language is Python, and Pytorch was chosen as the main library for building a neural network model.

Description of each layer of the neural network (the reproduced from PyTorch documentation[1]) is as follows:

- **Convolutional layer** applies 2D convolution to an input signal that consists of multiple input planes. Mathematically, this operation is illustrated by

$$out(N_i, C_{out_j}) = bias(C_{out_j}) + \\ + \sum_{k=0}^{C_{in}-1} weight(C_{out_j}, k) \star input(N_i, k), \quad (1)$$

  where $\star$ – 2D cross-correlation operator, N – batch size, C – number of channels, H – height of input, W – width of input.
- **ReLU layer** applies a rectified linear unit function element by element

$$ReLU(x) = (x)^+ = max(0, x), \quad (2)$$

  where x – input to the function.

[1]See "PyTorch: From research to production": https://pytorch.org/docs/stable/nn.html

- **Max pooling layer** selects the largest value in each pooling region. In this layer, the resolution of feature maps is reduced. Thus, the important information is preserved [37].
- **Flatten layer** flattens the input matrix into a one-dimensional tensor;
- **Linear layer** applies a linear transformation to the input data as

$$y = xA^T + b, \tag{3}$$

where y – output data, x – input data, A – weights, b – bias.

As depicted in Figure 7, the model consists of 3 blocks. The first one has a convolutional layer with a kernel size of 3 and 6 feature maps. ReLU activation function and Max pooling layer with a kernel size of 2 and stride of 2. The second block also consists of the convolutional layer with a kernel size of 3 and 16 feature maps, ReLU activation function, and Max pooling layer with a kernel size of 2 and stride of 2. These blocks perform feature extraction. The third block is aimed at predicting the expected value. It consists of Flatten layer, 3 ReLU activation functions and 4 linear layers with 128, 64, 32, and 1 neuron.

Importantly, each transformation is a differentiable operation. Therefore, gradient methods may be applied to optimize the weights of our model. The model is trained on GPU Nvidia GForce RTX 2080 Ti [38]. The applied loss function is MSE. The number of epochs for training is 10. The optimizer used is Adam, with a learning rate of 0.001.

The reasons for using this optimizer are [39]:

- Computational efficiency;
- Hyperparameters have a visual interpretation and usually do not require much tuning;
- Small memory requirements;
- Invariant to diagonal scaling of gradients;
- Suitable for problems with very noisy or sparse gradients (in our case, the distance measure has a fairly high variance, in which noise is not excluded).

The model parameters optimization occurs using the error-back propagation method, where the gradient calculation occurs backward through the network, with the gradient of the last weight layer computed first and the gradient of the first weight layer computed last. Partial gradient calculations for one layer are reused when calculating the gradient for the previous layer. Such backward flow of error information makes it possible to efficiently compute the gradient on each layer compared to the naive approach of calculating the gradient of each layer separately [40].

V. EVALUATION AND NUMERICAL RESULTS

This section elaborates on the main metrics of interest, provides selected numerical results and related discussion toward the integration with the infrastructure-based network.

A. METRICS FOR EVALUATION

The metrics that reflect the quality of the trained model are:

- Mean Absolute Error:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|, \tag{4}$$

- Mean Square Error:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{5}$$

- Coefficient of determination:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}, \tag{6}$$

where n is the number of objects in the sample; y_i – real value of the target variable; $\hat{y}_i$ – the predicted value of the target variable; $\bar{y}_i$ – the mean value of the target variable.

B. NUMERICAL RESULTS

Training time of proposed model was 23 minutes and 31 seconds (+/- 20 seconds variation was also observed).

Graphs showing the variation of the loss function and metrics on the training and validation sets during training are provided in Figures 8, 9, and 10. The model code is also available in Open Access on the GitHub platform for reproducibility [41].

Based on the obtained results, it takes on average 2 minutes 30 seconds for one iteration of model training, which allows, if necessary, to retrain it with updated data in a time-efficient manner. Naturally, the neural network allows for better predictions on training and validation sets over each iteration.

Note, the results in the last epoch seem to be getting worse, however, it could be due to the following. During the training, the intermediate results can change either positively or negatively. It is a part of the training process, and these changes can be insignificant on the larger number of epochs. Looking at the training process's general trend is much more important.

Moreover, the difference between training and validation results during the training process is relatively small. Therefore, the model is not overfitted. However, it is essential to note that the training and validation sets are statistically very similar, so the overfitting of the model should be evaluated carefully with the standard deviation in the data of more than $26,165$.

The results for all tested models are presented in Table 3. The methods are evaluated with metrics: MAE, MSE, R^2, where the best results, which can be achieved, for MAE and MSE is 0, and R^2 is 1. Here, the proposed model achieved results on the validation set: MAE – $1,200$, MSE – $4,762,818$ and R^2 – 0.99, and on the testing set: MAE – $1,575$, MSE – $10,119,205$ and R^2 – 0.98. The results achieved are promising. It seems that MSE values are too large. However, it is important to note that this value is a squared error; thus, large values may appear, especially when the distance values are relatively large (see Figure 11).

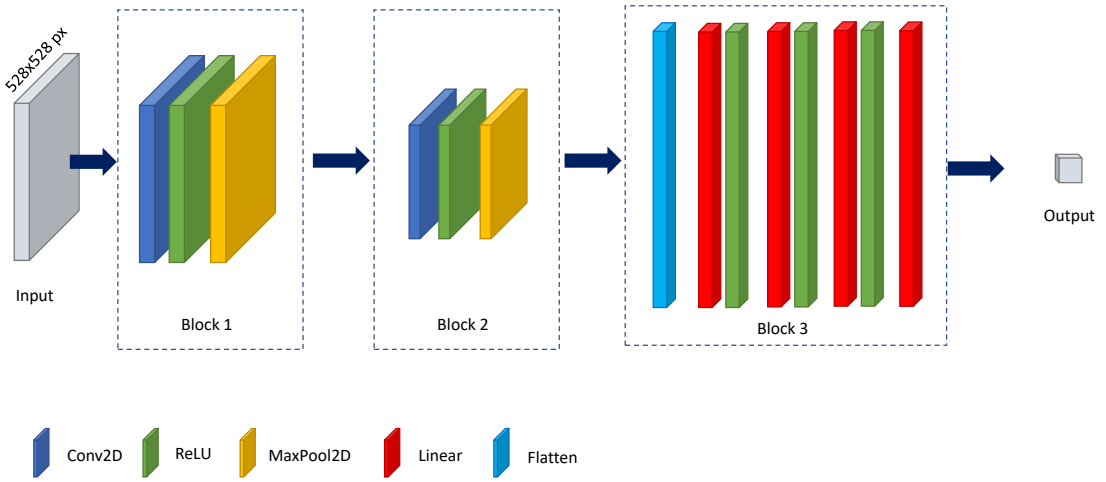

FIGURE 7. The architecture of proposed CNN.

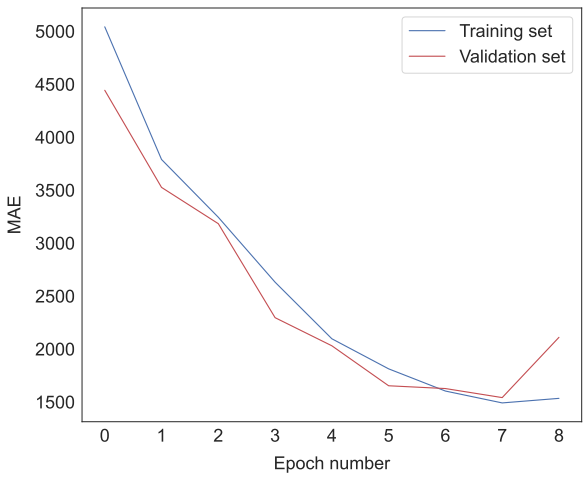

FIGURE 8. The value of the MAE metric during model training.

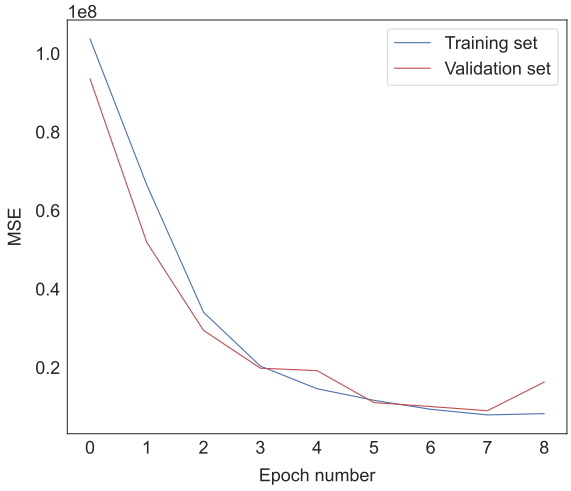

FIGURE 9. The value of the MSE metric during model training.

However, the absolute error MAE is much smaller, which is a good result.

According to Figure 11, the distribution of the target and predicted values on the testing set are almost similar. Thus, it could be concluded that the model could learn patterns for the correct distance prediction. This fact is supported by the result of R^2, which is almost ideal (0.98).

On the other hand, the baseline models perform worse than the proposed one. All of them have MAE in the range of $1,548 - 2,182$ on the validation set, and $1,482 - 2,278$ on the testing set, MSE is in the range of $8,394,001 - 15,004,180$ on the validation set and $10,220,651 - 15,369,350$ on the testing set, R^2 is in range of $0.98 - 0.99$ on the validation set, 0.98 on the testing set. MAE and MSE achieved relatively large values in comparison with the proposed CNN. However, the VGG16 model achieved better results on the validation set for metric MAE. Despite that, the proposed model still has better results with other metrics over the validation and testing sets. One of the possible reasons for the worse results of models from baseline is the complexity of the architectures. This dataset doesn't require complex architecture for feature extraction, which is why the designed model of the neural network is more suitable in this case.

The next aspect worth noting is training time. ResNet34 needs 5.1 min to train, DenseNet121 – 7.5 min, VGG16 – 11.5 min. Compared with the training time of the proposed model (2.5 min), the baseline models are more time-consuming. Consequently, it leads to the necessity of more

TABLE 3. Comparison of results on validation and testing sets with SOTA approaches.

Metric	MAE		MSE		R2		Time per epoch, mins
Type of set	Validation	Testing	Validation	Testing	Validation	Testing	
ResNet34	2,182	2,278	8,394,001	10,220,651	**0.99**	**0.98**	5.1
DenseNet121	1,644	1,635	15,004,180	15,369,350	0.98	**0.98**	7.5
VGG16	1,548	**1,482**	14,526,834	13,261,871	0.98	**0.98**	11.5
Proposed	**1,200**	1,575	**4,762,818**	**10,119,205**	**0.99**	**0.98**	**2.5**

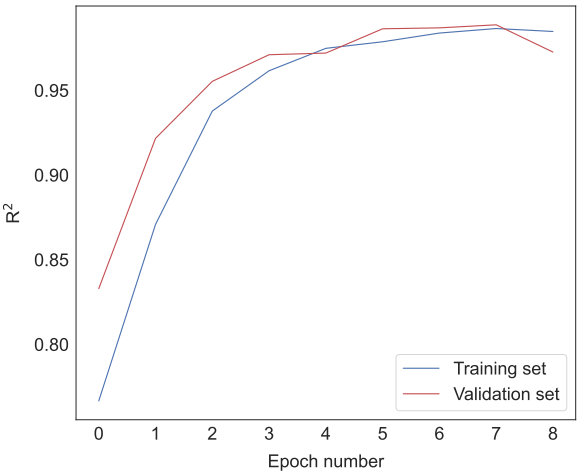

FIGURE 10. Coefficient of determination during model training.

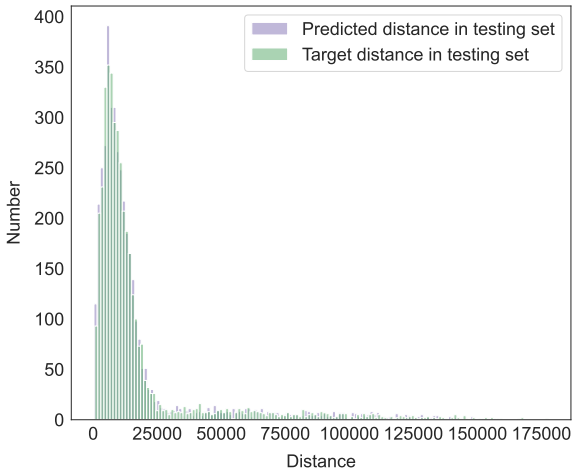

FIGURE 11. Distance values distribution in testing set and the predicted values.

time for retraining, which can have a bad impact on dynamic systems. At the same time, this model gives better results according to the objective metrics. It can be concluded that the proposed model is more efficient in contrary to other compared models.

Despite the represented results, there are possible options for improvement:

- Modification of model by adding additional layers;
- Application of other architectures of neural networks, such as RL, stacking with simpler algorithms, generative adversarial networks, neural graph networks, etc.;
- Weight regularization can help to reduce overfitting and improve the generalization of the model, which is useful in the real-world application;
- Use some additional information (features), but not only the pictures themselves, for example, additional knowledge about objects – about locations, through which the length of the optimal route is calculated;
- Additional experiments with different optimizers, loss functions, etc.;
- Application of cross-validation to reduce overfitting.

Finally, the initial preprocessing of images is worth mentioning as those had different sizes, and the model at the input requires a picture of the same size. In this regard, all images were scaled down to uniform resolution. It is a good first approximation, however, with such a transformation, some information may be lost in larger images, and noise is added in small ones, degrading the quality of the model.

In addition, the data has the following noted issues:

- Some paths could overlap, causing the ratio of total pixels to total length to be misleading;
- The resulting color becomes brighter as the paths overlap with other path segments and nodes.

In the current ML implementation, some information may need to be recovered due to image compression to a smaller size. This can be fixed, for example, by using so-called padding, i.e., when iterating over the data, the maximum possible size is taken from all the images, and all others are padded with constant pixel values. During training, an additional task of the model may be to determine the padded parts of the picture.

VI. POTENTIAL INTEGRATION WITH CELLULAR NETWORKS

The potential architecture combining all the technologies described above may solve some emerging problems, i.e., on the one hand, lack of capacity in data centers and, on the other hand, low computational capabilities of the end devices, as well as heterogeneity of the environment. In particular, using the proposed model would allow selecting

the offloading location not with NP-time depending on the number of available Edge nodes but flexibly and adaptively. Notably, the aim of this work is not to give the answer where the decisions about performing the offloading should be made but rather to highlight the importance of ML-powered decision making, which may allow to effective select the offloading location taking into consideration the target budget (being it computational needs, latency, energy efficiency, incentive or any other target value).

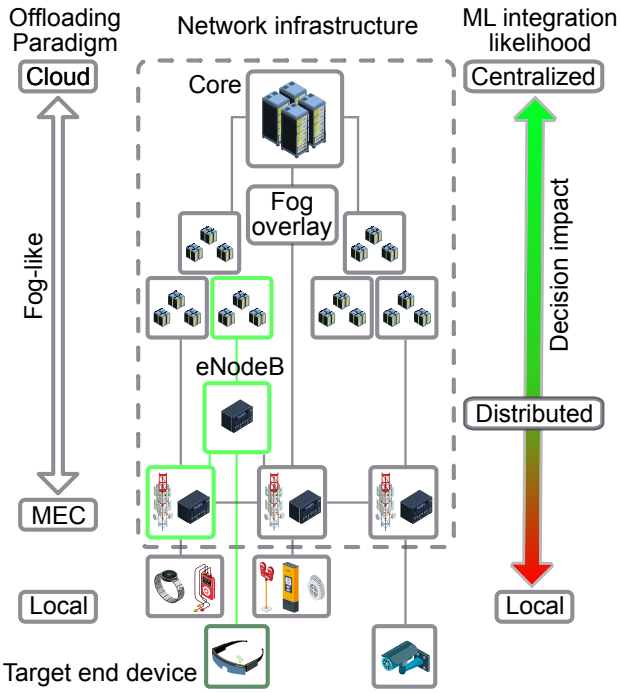

FIGURE 12. Architectural aspects of ML-powered offloading scenario.

Figure 12 shows an example case of the cellular network with a set of MEC-powered nodes. As those nodes are expected to be a part (or somewhat connected) of the cellular infrastructure, a natural hypervisor of the cellular network core is introduced. For simplicity, it could also be given the role of the MEC orchestrator (to be potentially integrated as the core module similarly to the authors' previous work [42]), which would get the ability to control the processing of a large amount of information due to its high computing capabilities. Naturally, the decision could be made in a centralized manner, however, this approach is not scalable and, thus, some decisions could also be made closer to the end-users, therefore, localized and/or distributed in a Fog/Cloud-like manner [43] on towards the eNodeB side [44]. Thus, there may be an impact of the (semi-)distributed Cloud on the actual offloading nodes' operation. It should also be noted that the more centralized the decision-making is, the more likely it is to be integrated with existing cellular operator infrastructure.

By introducing the orchestration rules, the execution of the ML process may be considered one of the most promising approaches for location selection [45]. The orchestration can be organized at a lower level through DL algorithms in conjunction with other ML models. For example, if a given computational offloading demand (target value) of a specific resource-constrained device is given to a trained ML network, it may provide the orchestrator (or the device itself, depending on the scenario) with a set of potential computational offloading locations that fulfill the demand/budget on the fly.

Indeed, defining and having access to the SOTA information of the network operation is critical from many perspectives [46]. The BSs are physical devices with fixed locations, and thus, traditional MEC operation may be ensured. Fog nodes could be considered weakly floating virtual spaces whose task is quickly identifying important dataflow from BSs while acting as a part of the orchestrator knowledge. End devices are still mobile and highly resource-constrained, but this aspect cannot be omitted in the dynamics of modern networks.

Nonetheless, there would be a need to obtain additional information concerning the wireless channels' load/quality/delays/etc., thus, have more knowledge of the system operation. It should be chosen optimally for each task. Still, it sounds unrealistic if the computational environment is separated from the cellular network infrastructure. However, it could be modeled with, e.g., probabilistic approaches [47]. Looking closely, the connections between the end devices and the Edge nodes are reminiscent of the images on which we trained the neural network in the previous sections, e.g., by finding the length of the optimal path. The interesting point is that *(sub)-optimality* can be understood in terms of distance and information transfer time.

Finally, the life cycle of an ML model is essential to be analyzed, keeping in mind that each stage of designing is an important part of the development. Without this consideration, it may be difficult to demonstrate a qualitative improvement from the introduction of ML in operation. The previous section showed that specific solutions to the TSP problem from the combinatorial optimization section are still relevant. An open-source dataset allows one to apply an ML algorithm to find the optimal path length by extrapolation, thus, choosing the optimal path or, for example, a data transmission channel between the BS and the remote Cloud node (or group of Fog nodes), optimizing the route calculation.

VII. CONCLUSIONS

Summarising, this paper outlines the analysis of scientific literature and formulates the main points that should be taken into account in future research or the design of a system that includes Edge Computing and ML by the analogy of finding an optimal route in imagery-based TSP. As a result, a general scheme of a mobile network was formed using Edge computing and CNN.

The training process of the proposed model is depicted in the graphs, and the results are represented in tables and discussed. In particular, we showed that the proposed model

outperforms the well-known architectures in almost all cases on the testing set: MAE result is $1,575$, MSE is $4,762,818$, R^2 is 0.98. Despite some large metrics results such as MSE and MAE, this model is still competitive since the comparable models achieved the same or worse results. Since the proposed model is light, it works faster than other architectures from baseline, which is an advantage over other methods in an applied problem. Additionally, possible improvements and future work are also introduced. The code is published on the GitHub platform under the MIT license [41].

In future work, we plan to continue the study of existing approaches, focusing on the functionality of Edge devices and architectural solutions in future-generation networks and a deeper dive into the TSP and its varieties from the point of view of mobile networks. Moreover, we plan to conduct experiments to improve the developed model, compare it with other approaches and apply it to the architectural scheme identified earlier. In addition, models will be trained for linking the Cloud-cellular and separately for the data center. After that, the entire system will be tested at a specially organized test stand. If such a concept is successful, implementation will be implemented, which involves application on real data, subsequent rapid scaling, and high-quality development.

ACKNOWLEDGMENT

We express our gratitude to N. Stepanov for his assistance and contributions to ML model developments.

ACRONYMS

AI Artificial Intelligence
AP Access Points
BS Base Station
CNN Convolutional Neural Network
DL Deep Learning
FL Federated Learning
GAN Generative Adversarial Network
GNN Graph Neural Network
ICT Information and Communications Technology
IoT Internet of Things
MAE Mean Absolute Error
MEC Mobile Edge Computing
ML Machine Learning
MSE Mean Squared Error
P2P Peer-to-Peer
PCAP Power and Channel Allocation Problem
QoS Quality of Service
RL Reinforcement Learning
RRN Remote Radio Node
RSU Roadside Unit
SL Supervised Learning
SOTA State-of-the-Art
SVM Support Vector Machine
TSP Travelling Salesman Problem
UL Unsupervised Learning

REFERENCES

[1] M. Wang, Y. Lin, Q. Tian, and G. Si, "Transfer Learning Promotes 6G Wireless Communications: Recent Advances and Future Challenges," IEEE Transactions on Reliability, vol. 70, no. 2, pp. 790–807, 2021.
[2] F. Guo, F. R. Yu, H. Zhang, X. Li, H. Ji, and V. C. Leung, "Enabling Massive IoT Toward 6G: A Comprehensive Survey," IEEE Internet of Things Journal, vol. 8, no. 15, pp. 11891–11915, 2021.
[3] Y. Zhang, "Mobile Edge Computing for Beyond 5G/6G," in Mobile Edge Computing, pp. 37–45, Springer, 2022.
[4] P. McEnroe, S. Wang, and M. Liyanage, "A Survey on the Convergence of Edge Computing and AI for UaVs: Opportunities and Challenges," IEEE Internet of Things Journal, 2022.
[5] A. Ometov, O. L. Molua, M. Komarov, and J. Nurmi, "A Survey of Security in Cloud, Edge, and Fog Computing," Sensors, vol. 22, no. 3, p. 927, 2022.
[6] Y. Siriwardhana, P. Porambage, M. Liyanage, and M. Ylianttila, "A Survey on Mobile Augmented Reality with 5G Mobile Edge Computing: Architectures, Applications, and Technical Aspects," IEEE Communications Surveys & Tutorials, vol. 23, no. 2, pp. 1160–1192, 2021.
[7] "Pai-driven aeronautical ad hoc networks for 6g wireless: Challenges, opportunities, and the road ahead,"
[8] N. Mäkitalo, D. Flores-Martin, J. Berrocal, J. Garcia-Alonso, P. Ihantola, A. Ometov, J. M. Murillo, and T. Mikkonen, "The Internet of Bodies Needs a Human Data Model," IEEE Internet Computing, vol. 24, no. 5, pp. 28–37, 2020.
[9] S. A. Huda and S. Moh, "Survey on Computation Offloading in UAV-Enabled Mobile Edge Computing," Journal of Network and Computer Applications, p. 103341, 2022.
[10] C. Sergiou, M. Lestas, P. Antoniou, C. Liaskos, and A. Pitsillides, "Complex Systems: A Communication Networks Perspective Towards 6G," IEEE Access, vol. 8, pp. 89007–89030, 2020.
[11] I. Alawe, A. Ksentini, Y. Hadjadj-Aoul, and P. Bertin, "Improving Traffic Forecasting for 5G Core Network Scalability: A Machine Learning Approach," IEEE Network, vol. 32, no. 6, pp. 42–49, 2018.
[12] A. H. Halim and I. Ismail, "Combinatorial Optimization: Comparison of Heuristic Algorithms in Travelling Salesman Problem," Archives of Computational Methods in Engineering, vol. 26, no. 2, pp. 367–380, 2019.
[13] F. Tang, B. Mao, Y. Kawamoto, and N. Kato, "Survey on Machine Learning for Intelligent End-to-End Communication toward 6G: From Network Access, Routing to Traffic Control and Streaming Adaption," IEEE Communications Surveys & Tutorials, vol. 23, no. 3, pp. 1578–1598, 2021.
[14] "Machine learning: Global markets to 2026." https://www.bccresearch.com/market-research/information-technology/machine-learning-global-markets.html [Accessed 24.11.2022], 2021.
[15] Q. Cui, Z. Gong, W. Ni, Y. Hou, X. Chen, X. Tao, and P. Zhang, "Stochastic Online Learning for Mobile Edge Computing: Learning from Changes," IEEE Communications Magazine, vol. 57, no. 3, pp. 63–69, 2019.
[16] D. Alekseeva, N. Stepanov, A. Veprev, A. Sharapova, E. S. Lohan, and A. Ometov, "Comparison of Machine Learning Techniques Applied to Traffic Prediction of Real Wireless Network," IEEE Access, vol. 9, pp. 159495–159514, 2021.
[17] J. Heaton, "Traveling Salesmen Computer Vision," 2022.
[18] Z. Meng, H. Xu, M. Chen, Y. Xu, Y. Zhao, and C. Qiao, "Learning-Driven Decentralized Machine Learning in Resource-Constrained Wireless Edge Computing," in Proc. of IEEE Conference on Computer Communications (INFOCOM), pp. 1–10, IEEE, 2021.
[19] P. J. Kaur et al., "Cluster Quality Based Performance Evaluation of Hierarchical Clustering Method," in Proc. of 1st International Conference on Next Generation Computing Technologies (NGCT), pp. 649–653, IEEE, 2015.
[20] K. M. Ahmed, A. Imteaj, and M. H. Amini, "Federated Deep Learning for Heterogeneous Edge Computing," in Proc. of 20th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 1146–1152, IEEE, 2021.
[21] Y. Xiao, G. Shi, Y. Li, W. Saad, and H. V. Poor, "Toward Self-Learning Edge Intelligence in 6G," IEEE Communications Magazine, vol. 58, no. 12, pp. 34–40, 2020.
[22] S. Hosseinalipour, C. G. Brinton, V. Aggarwal, H. Dai, and M. Chiang, "From Federated to Fog Learning: Distributed Machine Learning over Heterogeneous Wireless Networks," IEEE Communications Magazine, vol. 58, no. 12, pp. 41–47, 2020.

[23] X. Liu, "Resource Allocation in Multi-Access Edge Computing: Optimization and Machine Learning," in Proc. of 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), pp. 0365–0370, IEEE, 2021.

[24] Q. Han, S. Yang, X. Ren, C. Zhao, J. Zhang, and X. Yang, "OL4EL: Online Learning for Edge-Cloud Collaborative Learning on Heterogeneous Edges with Resource Constraints," IEEE Communications Magazine, vol. 58, no. 5, pp. 49–55, 2020.

[25] J. Thrane, B. Sliwa, C. Wietfeld, and H. L. Christiansen, "Deep Learning-based Signal Strength Prediction Using Geographical Images and Expert Knowledge," in Proc. of IEEE Global Communications Conference (GLOBECOM), pp. 1–6, IEEE, 2020.

[26] P. Wang and H. Lee, "Indoor Path Loss Modeling for 5G Communications in Smart Factory Scenarios Based on Meta-Learning," in Proc. of 12th International Conference on Ubiquitous and Future Networks (ICUFN), pp. 438–443, IEEE, 2021.

[27] S. P. Sotiroudis, P. Sarigiannidis, S. K. Goudos, and K. Siakavara, "Fusing Diverse Input Modalities for Path Loss Prediction: A Deep Learning Approach," IEEE Access, vol. 9, pp. 30441–30451, 2021.

[28] F. Tang, B. Mao, Z. M. Fadlullah, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "On Removing Routing Protocol from Future Wireless Networks: A Real-Time Deep Learning Approach for Intelligent Traffic Control," IEEE Wireless Communications, vol. 25, no. 1, pp. 154–160, 2017.

[29] K. Savla, E. Frazzoli, and F. Bullo, "Traveling Salesperson Problems for the Dubins Vehicle," IEEE Transactions on Automatic Control, vol. 53, no. 6, pp. 1378–1391, 2008.

[30] F. Frahadnia, "A New Method Based on Genetic Algorithms for Solving Traveling Salesman Problem," in Proc. of International Conference on Computational Intelligence, Modelling and Simulation, pp. 11–16, IEEE, 2009.

[31] M. Boffa, Z. B. Houidi, J. Krolikowski, and D. Rossi, "Neural Combinatorial Optimization beyond the TSP: Existing Architectures Under-Represent Graph Structure," arXiv preprint arXiv:2201.00668, 2022.

[32] M. Merluzzi, P. Di Lorenzo, and S. Barbarossa, "Wireless Edge Machine Learning: Resource Allocation and Trade-offs," IEEE Access, vol. 9, pp. 45377–45398, 2021.

[33] C. Feng, Y. Wang, Z. Zhao, T. Q. Quek, and M. Peng, "Joint Optimization of Data Sampling and User Selection for Federated Learning in the Mobile Edge Computing Systems," in IEEE International Conference on Communications Workshops (ICC Workshops), pp. 1–6, IEEE, 2020.

[34] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in Proc. of IEEE Conference on Computer Vision and Pattern Recognition, pp. 4700–4708, 2017.

[35] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in Proc. of IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778, 2016.

[36] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv preprint arXiv:1409.1556, 2014.

[37] D. Yu, H. Wang, P. Chen, and Z. Wei, "Mixed Pooling for Convolutional Neural Networks," in International Conference on Rough Sets and Knowledge Technology, pp. 364–375, Springer, 2014.

[38] NVIDIA Corporation, "GeForce RTX 2080 Ti and RTX 2080 Review Roundup," 2018.

[39] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," arXiv preprint arXiv:1412.6980, 2014.

[40] J. McGonagle, G. Shaikouski, C. Williams, A. Hsu, J. Khim, and A. Millerr, "Backpropagation," Brilliant Math & Science Wiki, brilliant. http://org/wiki/backpropagation/. Accessed 25.12.2022, vol. 3, 2018.

[41] "Link to the source code repository." https://github.com/aimezina/tsp [Accessed 28.11.2022], 2022.

[42] A. Ometov, P. Masek, J. Urama, J. Hosek, S. Andreev, and Y. Koucheryavy, "Implementing Secure Network-Assisted D2D Framework in Live 3GPP LTE Deployment," in Proc. of IEEE International Conference on Communications Workshops (ICC), pp. 749–754, IEEE, 2016.

[43] X. Jiang, F. R. Yu, T. Song, and V. C. Leung, "A Survey on Multi-Access Edge Computing Applied to Video Streaming: Some Research Issues and Challenges," IEEE Communications Surveys & Tutorials, vol. 23, no. 2, pp. 871–903, 2021.

[44] Y. Wang, J. Zhang, X. Zhang, P. Wang, and L. Liu, "A Computation Offloading Strategy in Satellite Terrestrial Networks with Double Edge Computing," in Proc. of IEEE International Conference on Communication Systems (ICCS), pp. 450–455, IEEE, 2018.

[45] K. Zhang, J. Cao, and Y. Zhang, "Adaptive Digital Twin and Multiagent Deep Reinforcement Learning for Vehicular Edge Computing and Networks," IEEE Transactions on Industrial Informatics, vol. 18, no. 2, pp. 1405–1413, 2021.

[46] N. Hassan, K.-L. A. Yau, and C. Wu, "Edge Computing in 5G: A Review," IEEE Access, vol. 7, pp. 127276–127289, 2019.

[47] K. Ray and A. Banerjee, "Prioritized Fault Recovery Strategies for Multi-Access Edge Computing Using Probabilistic Model Checking," IEEE Transactions on Dependable and Secure Computing, 2022.

BIOGRAPHIES

ALEKSANDR OMETOV (Senior Member, IEEE) received the M.Sc. degree in Information Technology and the D.Sc. (Tech.) degree in Telecommunications from the Tampere University of Technology (TUT), Finland, in 2016 and 2018, respectively. He also holds a Specialist degree in Information Security from the Saint Petersburg State University of Aerospace Instrumentation (SUAI) from 2013. He is a Senior Research Fellow at Tampere University (TAU), Finland, and the coordinator of the CONVERGENCE of Humans and Machines research field funded by the Jane and Aatos Erkko Foundation. He is a Project and Training Manager of EU H2020 MCSA A-WEAR and APROPOS ITN projects. His research interests include wireless communications, information security, computing paradigms, blockchain technology, and wearable applications. He was recognized as a Publisher of the Year in 2022 by TAU.

ANZHELIKA MEZINA received her Bachelor's degree and Master's degree in Information security at the Brno University of Technology (BUT), Czech Republic in 2018 and 2020, correspondingly. Currently, she is pursuing her Doctoral Degree in Information Security at BUT. The focus of her research is mainly leaning towards developing and applying DL methods for various real-world scenarios. Currently, she is involved in national-level funded research projects focusing on security and the medical fields (in cooperation with the Palacky University Olomouc and the Ministry of Interior of the Czech Republic). Her research interests are deep learning, information security, anomaly detection, computer vision, and image processing.

JARI NURMI (Senior Member, IEEE) received the D.Sc. degree in technology. He works as a Professor at the Electrical Engineering Unit, Tampere University, TAU (formerly Tampere University of Technology, TUT), Finland, since 1999. He works on embedded computing systems, System-on-Chip, approximate computing, wireless localization, positioning receiver prototyping, and software-defined radio and software-defined networks. He held various research, education, and management positions at TUT, since 1987, (e.g., an Acting Associate Professor from 1991 to 1994) and was the Vice President of the SME VLSI Solution Oy from 1995 to 1998. Since 2013, he has also been a Partner and Co-Founder of Ekin Labs Oy, a research spin-off company, now headquartered in Silicon Valley as Radiomaze, Inc. He has supervised 27 Ph.D. and about 150 M.Sc. theses and has been an opponent or a reviewer of 48 Ph.D. theses for other universities worldwide. He is a member of the Technical Committee on VLSI Systems and Applications at IEEE CAS. In 2004, he was one of the recipients of the Nokia Educational Award and a recipient of the Tampere Congress Award in 2005. In 2011, he received the IIDA Innovation Award and the Scientific Congress Award in 2013, and the HiPEAC Technology Transfer Award. He is a steering committee member of three international conferences (chairman in two). He has edited five Springer books and has published over 350 international conference and journal articles and book chapters. He is also an associate editor/handling editor of two international journals. He is the Director of the national DELTA doctoral training network of about 200 Ph.D. students, the Coordinator of the European doctoral training network APROPOS, and the Head of the A-WEAR European joint Ph.D. degree Program at TAU.