Received 14 March 2023, accepted 28 March 2023, date of publication 20 April 2023, date of current version 10 May 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3268763

RESEARCH ARTICLE

Partial Discharge Detection by Edge Computing

LUKÁŠ KLEIN[1], PETR ŽMIJ[2], AND PAVEL KRÖMER[1], (Senior Member, IEEE)
[1]Department of Computer Science, VSB—Technical University of Ostrava, 70800 Ostrava, Czech Republic
[2]Industrial Engineering, Brose Group, 74221 Koprivnice, Czech Republic

Corresponding author: Pavel Krömer (pavel.kromer@vsb.cz)

This work was supported in part by the European Regional Development Fund within the Operational Program Research, Development and Education in projects "A Research Platform Focused on Industry 4.0 and Robotics in Ostrava Agglomeration" under Grant CZ.02.1.01/0.0/0.0/17_049/0008425; in part by the Doctoral Grant Competition, VSB—Technical University of Ostrava (VSB—TU Ostrava), under Grant CZ.02.2.69/0.0/0.0/19_073/0016945; in part by the Student Grant System under Grant SP2023/12; and in part by the Doctoral Grant Competition "Partial Discharge Detection in Insulation Systems," VSB—TU Ostrava, under Grant DGS/TEAM/2020-015.

ABSTRACT Edge computing is becoming a mainstream platform for practical applications of machine learning and in particular deep learning. Many systems capable of efficient execution of deep neural models in the context of edge computing are readily available or beginning to appear on the consumer market. The Jetson platform from NVIDIA, the Neural stick from Intel, and the Edge TPU designed by Google are examples of devices that enable the application of complex neural networks in edge computing. This work investigates the ability of selected edge devices to address a real-world classification problem from electrical power engineering. It consists of the detection of partial discharges (PDs) from covered conductors (CCs) on high-voltage power lines. The CCs are used in heavily forested and generally inaccessible areas where clearance zones cannot be maintained. Detection of PDs can prevent forest fires and other disasters potentially caused by prolonged contact of CCs with vegetation. The problem is suitable for an edge computing-based solution because Internet connectivity in remote areas is usually insufficient and a 2G (GSM) mobile network is available at best. Because such locations are difficult to access and usually without a suitable power supply, the proposed solution puts an emphasis also on PD detection latency and the associated power consumption. Two principal approaches to PD detection are considered. One is based on the classification of 1D time series (raw data). The second approach uses the signal transformed into a 2D spectrogram. In this case, two types of algorithms are evaluated. The first one is a novel custom stacking ensemble detector composed of 2D convolutional neural networks and a neural meta-learner on top of it. The second one uses the well-known and widely-used used ResNet deep neural model.

INDEX TERMS Edge computing, experiments, edge TPU, neural networks, NVIDIA jetson, partial discharge, real-world applications, stacking ensemble networks.

I. INTRODUCTION

The area of edge computing attracts increasing attention from both, academia and industry and many devices for edge computing are commercially available today [1], [2]. A major part of the development in this field is dedicated to systems that enable deep learning and, in particular, deep neural network inference on the edge [3], [4]. The Jetson platform from NVIDIA, the Neural stick from Intel, and the Edge TPU by Google are examples of edge devices capable of inference and/or training of deep neural models. The use of deep neural models on edge devices presents many challenges, for example, training and inference latency and general performance of the systems [5]. Energy efficiency is a challenge, too, as some devices are powered by batteries and the available energy is limited [6].

There are several studies on the latency and energy efficiency of edge devices that allow the inference of deep neural models. For example, Kljucaric et al. [7] recently analyzed the hardware architecture of several edge devices including Google Edge TPU, Intel Neural Compute Stick 2 (NCS2), and NVIDIA Jetson AGX Xavier (AGX), and benchmarked

The associate editor coordinating the review of this manuscript and approving it for publication was Tony Thomas.

them on classic CNNs architectures such as AlexNet and others. Other studies focused on the optimization of certain network architectures and their benchmarking [8], [9], [10] or implemented neural networks optimized for target devices [11]. Some of them focus on the comparison of a single neural network on multiple systems [12], [13], other provide a comparison of multiple convolutional neural networks (CNNs) on Edge TPU, Jetson Nano, and selected GPUs [14].

Only a handful of research works focus on the use of deep neural networks on edge devices for real-world problems and thoroughly analyze practical results and the performance of different approaches (models) on different devices. In this work, we contribute to this area and analyze model performance and design novel deep neural network architectures suitable for the use on edge devices. We compare their accuracy and performance on a specific real-world problem and a variety of widely available systems. The problem is a part of a broader solution allowing the detection and classification of partial discharges using a contactless antenna-based method [15].

The contribution of this work is threefold. First, it compares the latency and performance of several deep neural models on 4 edge devices (three NVIDIA Jetson SoCs and an Edge TPU) in the context of two principal PD detection approaches. The first one is time series classification using the 1D stacking heterogeneous ensemble neural network, which we have previously designed [16]. The second one is based on the physical properties of the signal and leverages spectrogram and 2D CNN networks, which is a novel approach to this kind of data. In the second approach, we compare the well-known and widely used ResNet architecture [17] and a novel lightweight stacking ensemble of 2D CNNs with a neural meta-learner.

Second, it investigates the feasibility and performance of a stacking ensemble of 2D CNNs with a meta-learner on the top. In particular, the Wide & Deep [18] network is adopted and investigated because of good past experience and simple architecture that can be easily implemented on various edge devices.

Third, it studies the algorithms and edge devices from the approximate power consumption point of view. It demonstrates that a stacking ensemble of simpler neural networks is a feasible model that achieves a reasonable combination of latency and power consumption.

The rest of this paper is organized as follows: section II provides the background on the partial discharge detection problem and stacking ensembles, the top-level classification approach used in this work. Section III describes the data used in the experiments. Section IV provides the details on the individual classification methods and section V on the test devices. The experiments are detailed and their main results summarized in section VI and further discussed in section VII. Finally, major conclusions are drawn in section VII.

II. BACKGROUND

A. PARTIAL DISCHARGE DETECTION

High-impedance faults, such as long-term direct contact with vegetation, represent a severe problem for medium-voltage overhead (22 kV) lines with covered conductors (CCs) [19]. CCs are designed to tolerate contacts with grounded objects such as trees fallen on the power lines, without immediate interruption of the power supply [20]. However, the use of CCs alone cannot eliminate all problems.

In remote locations where vegetation clearances cannot be set up or properly maintained, high-impedance faults appear as a result of contact between power lines and the surrounding vegetation [21]. The use of covered conductors prevents their immediate impacts but cannot prevent the long-term effects of such faults. The detection of high-impedance faults is not easy. The fault current is very low and cannot be identified by standard protection relays. The faults are accompanied by partial discharges (PDs) [22] that gradually degrade and eventually damage the CCs and can lead to XLPE insulation failures (usually in a matter of days) [23]. This leads to power failures, can cause forest fires, injury or even death under certain conditions (damage of step voltage), and results in an interruption of power supply. The process when a fault is formed and PDs happen takes a longer period of time and ought to be prevented whenever possible.

PDs are usually detected using the galvanic contact method. It is suitable for medium-voltage power lines but fairly expensive [24], [25]. The detection of PDs from the data obtained by galvanic contact has been thoroughly investigated [26], [27], [28]. In this work, we focus on another principal method for PD detection. PDs create in the electromagnetic field surrounding the CCs specific patterns that can be observed using an antenna [15]. Such contactless approach is an alternative to data acquisition by galvanic contact. The price of data collection is in the contactless case significantly lower and the measurement device can be installed without a power outage. However, the measured data (signal) is very noisy and essentially lacks the information known to be exploited by the best-performing detection algorithms used for the galvanic contact method of measurement [29].

CCs are used in areas that are hard to access such as national parks or remote mountainous regions. On-site detection of PDs in such locations is a suitable application scenario for edge rather than cloud computing. The primary reason is that the acquired data is fairly large (800 kiB) for this type of application and the Internet/5G connectivity or LPWAN is typically unavailable or insufficient in remote areas. In the ideal case, an on-site edge device would perform the detection fully offline and communicate only if a PD is detected. An important requirement of the detection system is energy efficiency. The detection devices in remote areas are likely to be powered by batteries and their lifetime ought to be maximized because battery replacement/maintenance would be expensive.

B. STACKING ENSEMBLES OF DEEP NEURAL NETWORKS

Stacking is a top-level ensemble machine learning strategy that allows the combination of multiple base classifiers into more powerful classification models [30]. The decisions of baseline (low-level) classifiers are processed by another top-level model (meta-learner) to obtain the overall prediction [31]. Stacking is an ensembling approach with good results in various application areas, e.g., biomedical data analysis [30], [32], [33], [34], detection of defects [32], or automated recognition of sign language [35]. Several studies have demonstrated that stacking ensembles of homogeneous or heterogeneous neural networks can outperform individual (baseline) models [30], [32], [33], [34], [35] and represent the state-of-the-art in time-series classification [36]. Moreover, it has been shown that stacking ensembles have a great potential for the use in the context of edge (IoT) computing [31].

While the use of neural models, e.g., 2D CNNs, in stacking ensembles is commonplace, the top-level meta-learner is often based on another machine learning algorithm such as Support Vector Machines [37]. In this work, we use models with neural network-based meta-learners to fully utilize the acceleration potential of specialized devices (mobile accelerators, TPUs).

C. DEEP LEARNING ON EDGE DEVICES

Traditional deep learning algorithms are typically trained and executed on powerful, centralized servers or computers that have vast computing resources and can easily process large amounts of data. The algorithms require large volumes of data in order to learn the input-output mappings associated with the solved problems, for example to make accurate predictions. The training (model learning) is often slow and resource-intensive [38].

Algorithms for edge computing have to be designed for execution on devices that are smaller, more lightweight, and restricted but located closer to the sources of data, such as smartphones and sensors [39]. They can process data in real time and do not require centralized processing. This makes them useful for applications where real-time processing or data privacy is important. There are many different algorithms and models that can be used for this type of applications [10], [40]. In our work, we focus on the general design of efficient NN architectures. We propose and evaluate ensembles of simple models that can be executed in parallel (the overall network is wider) and their outputs are consolidated by a single meta-learner. Increasing the width of the network has been identified as a potential pathway towards overall performance improvement on Edge TPU [41] but we evaluate this general architecture on other edge devices, as well.

Data quantization and potential loss of accuracy are techniques commonly used on edge devices, and the use of FPGAs is also being explored. Most deep neural networks are in such restricted environments used for inference only, as it is difficult to train them directly on the edge. However, the possibility of model training on the edge is an area of recent research [42] with a great potential for enabling online model adaptation, etc.

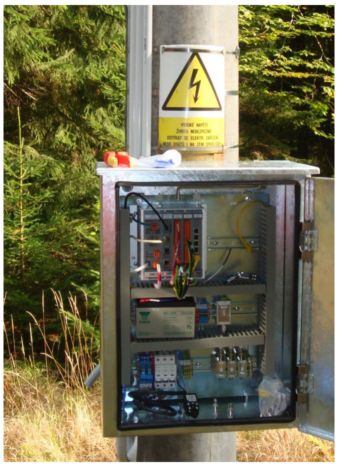


FIGURE 1. Data collection device.

To achieve the most efficient solution, it is best to consider using an ASIC-based system-on-chip with a processor that has integrated engines (hardware and software) designed specifically for machine learning [39], [43]. The processor can be used to control the engines and run non-deep learning services, but the dedicated engine should also have some flexibility to adapt to the current state of the deep learning algorithm (model). Examples of such devices include Edge TPU [8], Edge Board, and Hikey 970 [44].

III. DATA

A. DATA ACQUISITION

The data used in this work was collected with the help of measurement stations located in forested areas and challenging terrains. The stations were originally designed to detect PDs using the galvanic contact method and were monitoring medium voltage power lines with CCs in the Czech and Slovak Republics. Later, the contactless method was also implemented at these stations through the addition of new input to a DAQ card. The detection system can be seen in fig. 1. The hardware of the detection device is a proprietary solution manufactured by the ELVAC company. The detection devices contain ARM CPUs and run the Linux operating system.

Additionally, a BONI whip antenna was installed in the measurement stations. It is an omnidirectional active antenna that can detect PDs from any direction. The BONI whip is the ancestor of the popular Mini Whip antenna and is manufactured by Bonito, Germany. The antenna is shown in fig. 2. It has a frequency range of 20 kHz to 300 MHz but the data was only sampled up to 20 MHz. The antenna is 17 cm long and has a gain of 3dB. It is powered by a voltage supply of 14V-15V and has an IP3 of +32.5 dBm and an IP2 of +55 dBm.

The sensitivity of the antenna in the field can vary due to the characteristics of the power line. It is not possible to

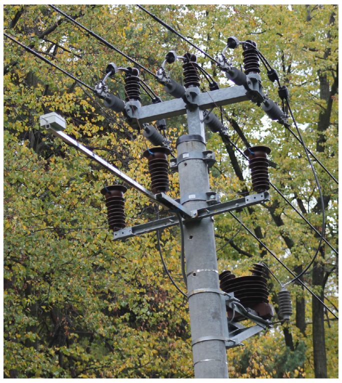


FIGURE 2. Antenna for capturing the signal installed on an overhead power line.

calibrate the antenna on-site because shutting down the entire power grid, which is required for calibration, is not allowed by the power grid operators.

B. DATASET DESCRIPTION AND ANALYSIS

The data for PD detection was captured by a wireless antenna, which received the electromagnetic signal from the CCs. However, the signal contains a large number of noise artifacts [45]. The noise usually compromises a significant portion of the signal because only the highest received frequency is captured. In addition, the signal changes its properties, such as magnitude, in time. The signal was processed by an 8-bit analog-to-digital converter (ADC), which returns values in the range of -128 to 127. One time series consisted of 800,000 observations, which present one full period (20 ms of a 50 Hz system).

A signal plotted as amplitude over time usually contains two larger clusters of higher electromagnetic activity, with the rest of the signal having relatively little activity. This corresponds to the three phases of an alternating current in overhead power lines. There are also usually visible peaks in the signal, which may indicate the presence of partial discharges or other noise.

The data set contains recordings from 18 different locations across the Czech Republic. Its content corresponds to real-world partial discharge occurrences and is largely unbalanced. The positive-to-negative case ratio is 1 : 3.5. The data set contains 3,992 time series corresponding to partial discharges (positive cases) and 13,767 time series without partial discharges (negative cases). The ground truth was evaluated by the reference galvanic contact method and such as could be partially wrong, but in our case, it is sufficient and the galvanic contact method algorithm has an accuracy of 96.7%, a sensitivity of 98.4%, a specificity of 99.8%, and a precision of 70%.

The data contained a large amount of noise, as well. In addition to PDs, it contained time series associated with other types of discharges, such as corona discharge or rime on covered conductors. An example of a signal from the data set can be seen in fig. 3, where the signal contains visible peaks, which could indicate possible PDs. For example, detecting high impedance failures in conductive cores caused by rime is still an unresolved problem.

IV. PD DETECTION METHODS

This work uses two principal approaches to PD detection. The first one is based on the analysis of raw data received by the contactless antenna, which we developed in our previous work [16], which was also designed to run on HW accelerators and the second one, the new one, makes use of spectrograms extracted from the raw signal.

A. 1-DIMENSIONAL TIME SERIES ANALYSIS

The raw data collected from CCs by the contactless antenna has the form of a 1-dimensional time series. To improve its usefulness, the data is first denoised with respect to the frequency of values in the time series. The denoising is defined as

$$x_{\text{denoised}} = x \cdot \left(\frac{h_x}{l}\right)^z, \tag{1}$$

where x is a single entry of a time series, h_x is the frequency of x in the time series, l is the number of occurrences of the most frequent entry in the time series, and z is a constant. An example of a denoised signal can be seen in fig. 4.

Then, max-pooling is applied in order to reduce the size of the time series. This step is performed because the original size of the input (800 kiB) is quite large to process by traditional deep learning models, especially in the constrained environment of edge devices. In the end, each time series was transformed into 1563 signed bytes.

After that, a stacking ensemble of neural networks was used to detect PDs. The model contained two 1D-CNNs and two autoencoders. The decisions of the models were passed to a Wide & Deep network [18] that performed the top-level detection. An overview of the model is shown in fig. 5.

B. SPECTROGRAM ANALYSIS

The second approach exploited the physical properties of the captured signal. It consists of electromagnetic waves and a detector can look for specific properties associated with partial discharges. In order to do that, the signal was converted to a spectrogram, as shown in fig. 6. Partial discharges on spectrogram are not distinguishable by the human eye. Also, the difference between samples with partial discharge is not sufficient for an expert PD identification, as illustrated in fig. 7. However, machine learning algorithms are capable of learning from such subtle differences.

A spectrogram is a 2D visual representation of the variations of a spectrum of signal frequencies over time. This enables better detection of partial discharges that are defined as wide-spectrum electromagnetic interference over a short period of time. The data is split into segments with respect to a floating window and overlap of particular sizes. The fast Fourier transformation is performed over each segment. The x-axis of a spectrogram represents time, the y-axis represents

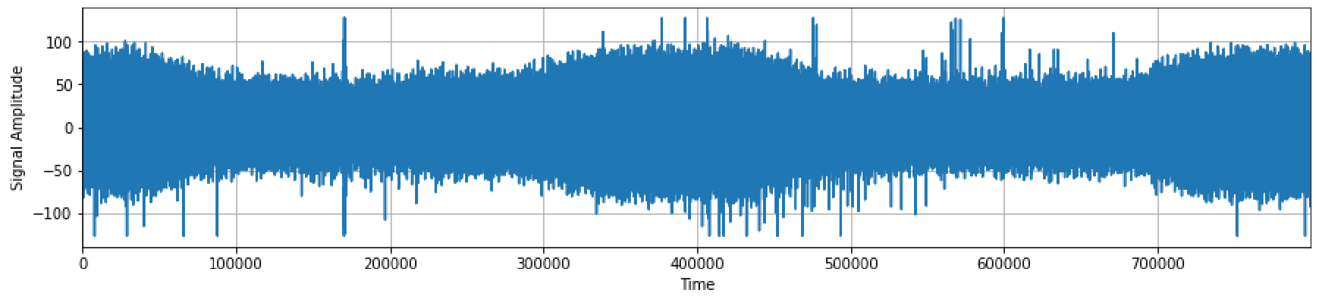


FIGURE 3. A signal, which contains PDs, that are visible even by the human eye.

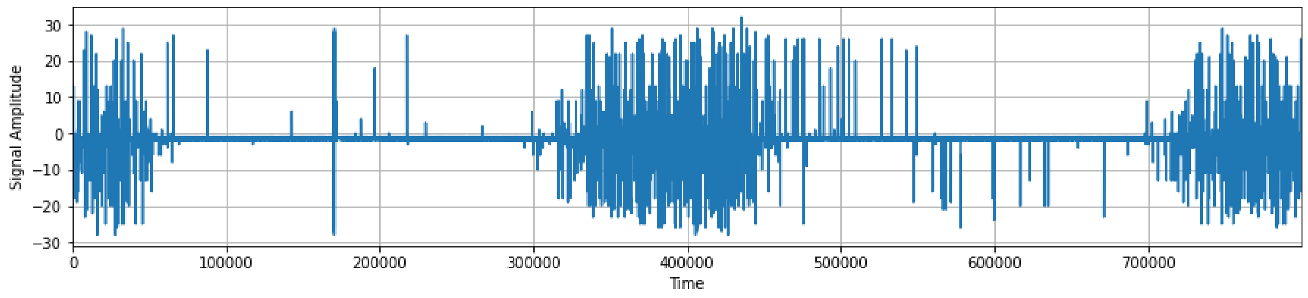


FIGURE 4. A time series with PDs after denoising, which highlights PDs.

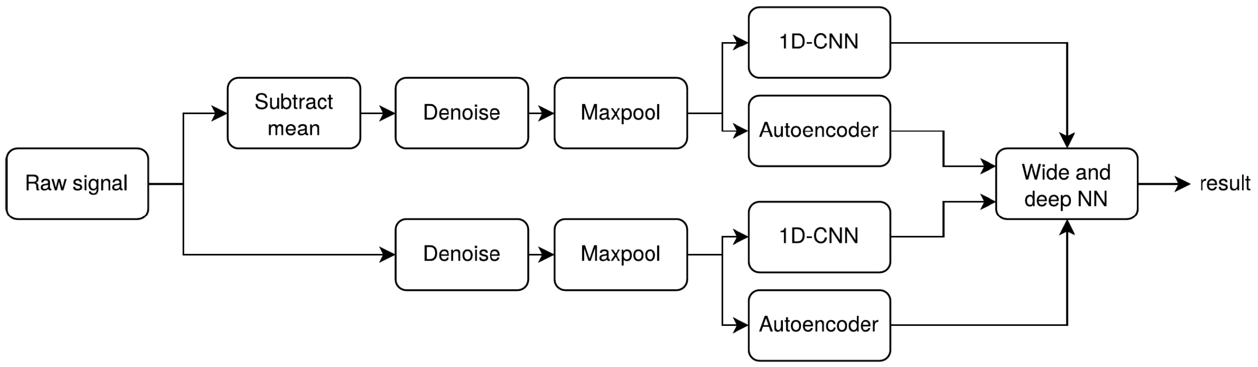


FIGURE 5. Overview of the 1D Convolutional Neural Network stacking ensemble using autoencoders and 1D-CNN networks.

frequency, and the intensity of the signal is represented by the color or intensity of the pixels on the graph. In other words, the spectrogram provides a convenient visual representation of how the frequency content of a signal changes over time.

After the conversion of the signal to a 2D spectrogram, we used a stacking ensemble of 2D CNN-based networks and the ResNet network to analyze it. The stacking ensemble was designed specifically for this task while the second model represents a well-known and widely-used deep neural network architecture. ResNet was chosen because of its past good results in other applications, wide availability on a number of edge devices, and best performance in initial trials. It achieved similar results but lower complexity than EfficientNet, better accuracy than MobileNet, and overall better performance than VGG (all networks were tested on inputs of 320×240 RGB pixels).

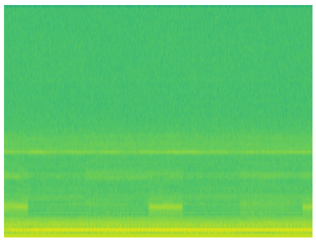


FIGURE 6. Visualization of a spectrogram of a single captured signal from the antenna. The visual representation very well illustrates at what time and for what frequencies varies the intensity of the signal most (pixel colors are different) and at what time and for what frequencies is the intensity uniform (pixel colors are similar or the same).

1) STACKING ENSEMBLE OF 2D CNNs

In this work, we propose a stacking ensemble of 2D CNNs and a top-level neural meta-learner. Each CNN in the ensemble was trained separately and their outputs were passed to

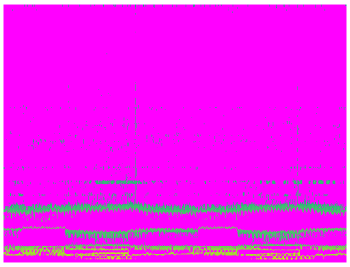


FIGURE 7. An example of a false color difference between two spectrograms from a contactless antenna with and without partial discharges using a simple RGB comparison (pixel diff). The differences may be subtle and difficult to discern visually due to background noise from sources such as TV broadcasts, radio, and other sources of noise. The colors also clearly identify which parts of the spectrogram that are important for the classification problem.

a meta-learner in the form of a Wide & Deep network. The architecture is inspired by [46] and the relative success of the stacking ensemble in 1D time series analysis. It is assumed that each CNN might base its decisions on a slightly different set of features because they are trained independently. The top-level meta-learner processes the outputs of all members of the ensemble and provides the final decision on whether the spectrogram contains partial discharges or not. Moreover, an architecture comprising multiple lightweight CNNs makes this architecture suitable for acceleration and concurrent execution of the model for better performance. An overview of the ensemble is shown in fig. 8.

The ensemble makes use of five 2D CNNs with various depths (number of 2D blocks in the center of the network). For the depths, we chose 1 2D block and then selected the prime numbers (3, 5, 7, 11). A depth of more than 11 blocks did not improve the ensemble results. 2D CNN networks were the most simple CNN networks with convolution, max-pooling, batch normalization, and ReLu activation. This was done to simplify the network so that it can be used on all edge devices. This design also enables the utilization of the available width of the Edge TPU computing unit [41].

To optimize the ensemble, a number of hyperparameters (the number of layers, filters, kernel sizes, and activation functions) was adjusted in a series of trial-and-error experiments. The hyperparameter values were not selected by an exhaustive search but were taken from a pool of values that performed well according to the literature and past experience of the researchers.

2) ResNet V2

ResNet is a well-known deep neural network proposed in 2015 [47]. To solve the problem of disappearing gradients, ResNet introduced a concept called residual blocks. It uses skip connections that link layers to subsequent ones by skipping some others. This forms a residual block and the ResNets model is created by stacking the residual blocks together.

ResNet V2 [17] is an evolution of the original ResNet architecture with two main changes: it uses the stack of batch normalization, ReLU activation function, and 2D convolution and removes the activation function after an addition (skip connection).

In this work, we use two particular ResNet V2 models to study the changes in performance in relation to network depth. The first one was a 50-layer deep network that was the best-performing ResNet V2 architecture on these data and the second one consisted of 152 layers. The models were created in Keras with a fully connected layer with dropout and sigmoid activation as the top layer and re-trained for the PD detection problem.

3) VGG-19

VGG-19 [48] is a convolutional neural network model trained originally on the ImageNet dataset. This model and its variants are widely used for image classification tasks. The network has 19 layers (hence the ‘‘19’’ in the name of the model) and is characterized by the use of small convolutional filters, with filter sizes of 3×3, and a rather deep architecture.

The VGG-19 architecture consists of 19 layers, including 16 convolutional layers and 3 fully-connected layers. The convolutional layers are arranged in groups, with the first group using 64 filters, the second using 128 filters, the third using 256 filters, and the fourth using 512 filters. The fully-connected layers have 4096 nodes each and the weights are initialized randomly. In this work, we preliminary evaluated this architecture but it was found inferior to the evaluated methods in terms of both, accuracy and performance. Because of that, further experiments focused on the proposed approaches only.

4) MobileNetV2

MobileNetV2 [49] is a convolutional neural network architecture designed for efficient on-device image classification. It was developed by Google and introduced as an improved version of MobileNetV1, which was designed to be small and efficient for mobile and embedded devices.

MobileNetV2 uses depthwise separable convolutions, which leads to a significant scaling down of the number of parameters and computations in the network, making it more efficient. It also uses a linear bottleneck design which further reduces the number of operations in the network while maintaining a high level of accuracy. MobileNetV2 has been frequently used for a variety of applications, including object detection and classification, image segmentation, and facial recognition. We used random weight initialization and kept the default Keras settings. In our work we also preliminary tested MobileNetV2 but it achieved worse results than ResNet V2.

5) XGBoost

XGBoost [50] is a powerful machine learning algorithm that has been successfully applied to tackle a variety of problems including image classification. It is an implementation of gradient boosting, which involves training multiple weak models in a sequential manner and combining their predictions to form a strong model.

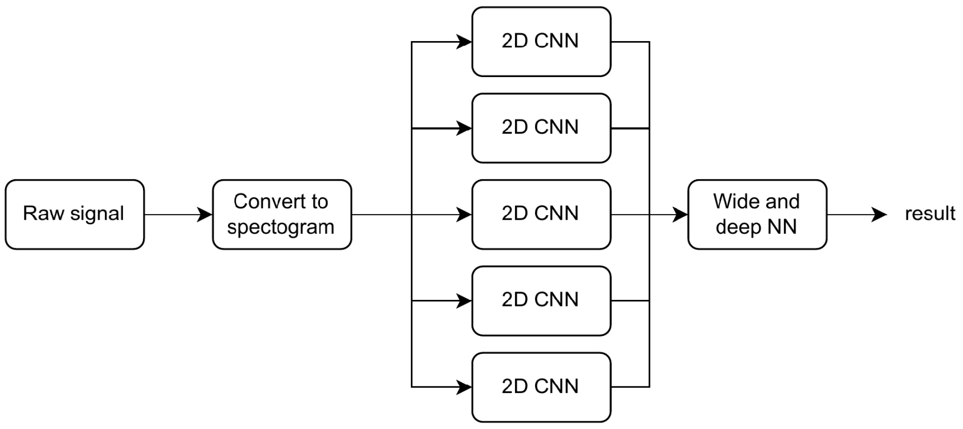


FIGURE 8. **Overview of the 2D Convolutional Stacking Neural Network Ensemble.**

One of the key advantages of XGBoost is its ability to handle large datasets and high-dimensional features efficiently, making it well-suited for image classification tasks. It also has a number of advanced features that allow for fine-tuning of model parameters and handling imbalanced datasets. This algorithm allows GPU acceleration and was included in preliminary testing. However, the rather complex nature of this model made it unsuitable for use on the target category of edge devices.

C. NETWORK TRAINING

All networks used in this work were defined and trained using the Keras framework [51]. they were trained on a cluster of four NVIDIA GTX 1070 GPUs. The PD data was divided into two sets: training and test. The test data set contained 20% data and 15% of the training data set was used for validation. All data sets were stratified.

The training used early stopping set to track validation loss with 15 epochs patience and learning rate reduction by 50% on a plateau with 15 epoch patience. This allowed training the networks until they started to overfit. To select the model with the best generalization ability, the weights of the network with the best validation loss were saved and later evaluated over the test data set.

V. TEST DEVICES

The proposed PD detection models were evaluated on several commercially available edge devices capable of hardware-accelerated neural network inference and suitable for edge applications.

A. GOOGLE EDGE TPU

Edge TPU is a Google-made application-specific integrated circuit (ASIC) for deep neural network applications. It is capable of inference only and has a restricted set of operations. In exchange, it features excellent energy efficiency and achieves a very good inference performance [52]. Edge TPU can perform 4 trillion operations per second (TOPS), using 0.5 Watts for each TOPS (2 TOPS per Watt). In this work, it was used together with the Dev Board Mini with MediaTek 8167s SoC (Quad-core Arm Cortex-A35).

Edge TPU accelerators use a template-based design. Template accelerators are organized into 2D arrays of processing elements arranged as templates and accelerators. Each processing element performs a set of arithmetic operations in a single instruction multiple data (SIMD) manner [41]. Edge TPUs support the quantized TensorFlow lite model, which is mapped to operations of the device. The operations are highly limited, and, for example, deconvolution is not available at all. It means that the autoencoder used for 1D time series analysis cannot be easily run by the Edge TPU platform. A workaround would be the use of split convolution [53]. However, this approach would require after each split deconvolution the transfer of the entire memory to the host, making the entire process extremely slow. Also, a bug in the Edge TPU compiler prevented the compilation of the 1D CNN network on this platform.

B. NVIDIA JETSON FAMILY OF SoCs

NVIDIA Jetson is a family of SoCs designed for high-performance applications requiring low power consumption. Jetson modules are more general and have better performance than TPUs. To deploy and run the neural networks, the Open Neural Network Exchange (ONNX) framework was used. The ONNX runtime used TensorRT as the backend and was optimized for each Jetson device separately. If a network, e.g., the autoencoder, uses operations not available in TensorRT, the ONNX compiler automatically falls back to the lower-level CUDA architecture. Three different Jetson modules were used in the experiments: Jetson Nano, Jetson Xavier NX, and Jetson Xavier AGX. The properties of the modules are shown in table 1.

For a broader context, the models were also executed on a desktop GPU, in particular the NVIDIA GTX 1070. It is a high-end desktop GPU introduced by NVIDIA in 2016 based on the Pascal architecture. The same GPUs were used for training of all models.

C. PERFORMANCE EVALUATION AND POWER CONSUMPTION

The performance of all models was benchmarked on every device capable their execution. Each model was executed on every device 1000 times and the execution times were recorded. In the benchmark loop, a new input was provided in every iteration and inference was performed (model initialization was already done). The Edge TPU used models compiled and optimized for its architecture. The Jetson devices used the ONNX runtime [54] with the TensorRT backend [55].

During the benchmarking, the power consumption of the devices was measured. To track power consumption, the open source jtop[1] library was used on the Jetson devices. It performs a software-based estimate of power consumption but can even provide information about the power consumption of particular parts of the device [56].

To measure the power consumption of the Edge TPU, a hardware USB power meter was used. Because it reports only peak power consumption, average power consumption could not be determined for this device.

VI. RESULTS

A series of computational experiments was performed to study the properties of the PD detection models presented in this work. They focused on three principal areas: accuracy, speed, and energy costs of online PD detection (inference) of different models on the target edge devices.

A. PREDICTION ACCURACY

First, the ability of the models to predict PDs was investigated. Prediction accuracy was evaluated by the Matthews correlation coefficient (MCC) [57] because the data set is highly unbalanced. The MCC is a special case of Pearson's correlation coefficient better suited for unbalanced data. It takes values from the range -1 to 1. The value of -1 indicates a complete disagreement between the prediction and the observation. The value of 0 suggests that the prediction is no better than a random guess, and in the case of MCC value 1, the predictions are in perfect agreement with the observations.

In our preliminary experiments, we evaluated the performance of several machine learning algorithms for MCC on different devices. These algorithms included Ada Boost [58], Random Forest Classifier, C-Support Vector Classification (SVC) with optimized parameters using grid search (optimized parameters: C - coef0, gamma and kernel), and XGBoost. We used the scikit-learn library [59] for Ada Boost, Random Forest Classifier and to optimize the parameters for the SVC algorithm. We also tried the XGBoost algorithm [50], which can be easily implemented on GPU-focused platforms like the Jetson family.

[1]https://github.com/rbonghi/jetson_stats

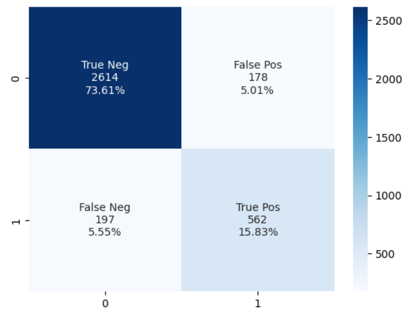


FIGURE 9. Confusion matrix for ResNet50V2.

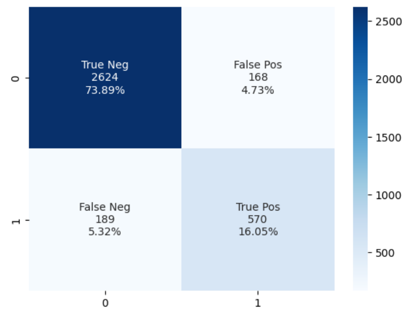


FIGURE 10. Confusion matrix for ResNet152V2.

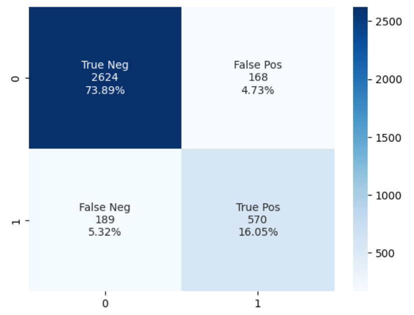


FIGURE 11. Confusion matrix for 2D CNN ensemble.

Table 2 shows the values of the MCC between predictions and ground truth obtained by all models. The models were trained and evaluated 100 times using random cross validation. The best accuracy (highest MCC) was obtained by the 2D CNN ensemble. The second-best accuracy was obtained by the 50-layer ResNetV2 (ResNet50V2) and the third-best by the 152-layer version of the same model (ResNet152V2). The accuracy of both ResNetV2-based models is illustrated by confusion matrices shown in fig. 9 and fig. 10, respectively.

The worst prediction was obtained by the 1D ensemble of neural networks applied to raw data.

The other evaluated algorithms had worse performance in terms of MCC and were too complex to be further considered. For the rest of the results, we only used ResNet V2 with

TABLE 1. NVIDIA Jetson devices.

Device	Jetson Nano	Jetson Xavier NX	Jetson AGX Xavier
AI Performance	472 GFLOPs	21 TOPs	32 TOPs
GPU	128-core Maxwell™	384-core Volta™	512-core Volta™
CPU	Cortex®-A57	6-core Carmel Arm®v8.2	8-core Carmel Arm®v8.2
Memory	4 GB 64-bit	8 GB 128-bit	32 GB 256-bit

TABLE 2. MCC averaged over 100 cross-validated results.

Neural network	MCC
Ada Boost	0.141
Random Forest Classifier	0.211
XGBoost	0.224
SVC(C=1000, gamma=0.001)	0.288
VGG19	0.495
MobileNetV2	0.512
1D CNN ensemble	0.523
ResNet152V2	0.542
Single 2D CNN	0.571
ResNet50V2	0.683
2D CNN ensemble	**0.698**

152 layers, Single 2D CNN, ResNet V2 with 50 layers, and the proposed 2D CNN ensemble. For 1D input-based algorithms, we only used the best published algorithm, which outperformed all other algorithms that we tested in our previous work [16].

The 2D CNN ensemble was more accurate than the single 2D CNN network. A single 2D CNN had the worst MCC of 0.416 and the average MCC of 0.571. Even the best single 2D CNN (0.648) was worse than the ensemble of 2D CNNs with the Wide & Deep meta-learner with the MCC of 0.698. The confusion matrix of classification by the 2D CNN ensemble is shown in fig. 11.

B. INFERENCE SPEED

Next, the time needed for inference on different target devices by different models was evaluated. Table 3 shows the average time needed for the classification of a single spectrogram as an average of 1000 independent runs. When a model could fit into the memory of the Edge TPU, then the device was the second fastest (the case of the single 2D CNN). In other cases, the Edge TPU was only slightly slower than Jetson Nano. The other results were largely as expected: the desktop GPU, GeForce GTX 1070, was the fastest device, followed by Jetson AGX Xavier, Jetson Xavier NX, and Jetson Nano. With regard to the models, the most time-consuming inference was that of the largest network, ResNet152V2. The Edge TPU was unable to successfully finish the inference of ResNet152V2 and froze at every attempt.

The results obtained for the 1D CNN ensemble were rather surprising; the sum of latency of each part of the ensemble was much greater than a single inference from the ensemble as is showed in table 4. Also, as can be seen from 3, there was only a modest improvement in latency at Jetson Xavier NX when compared with Jetson Nano. This is different from the spectrogram-based approach where the use of Jetson Xavier

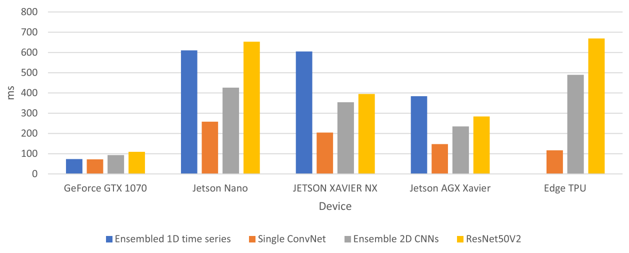


FIGURE 12. Inference latency.

NX clearly improved inference latency in comparison with Jetson Nano.

An interesting outcome of the experiments (see table 3) was the very small difference in inference time between a single 2D CNN and the 2D CNN ensemble on a GeForce 1070 GPU. However, the difference in the inference times on edge devices such as the Jetson Nano was more significant, with the ensemble resulting in a 30% increase in average inference time compared to the single 2D CNN's 65% increase.

Remarkably, the difference was smaller on more powerful devices such as the Jetson AGX Xavier. It was also observed that the 1D CNN ensemble was much faster (relative to the 2D CNN) on a classical GPU than on edge devices. On GeForce 1070, it achieved similar inference time as the single 2D CNN, but it was much slower on edge devices.

C. SPACE REQUIREMENTS

The target devices use several different formats for model representation. The Keras format is fully based on TensorFlow and used by the popular Keras framework, the ONNX is an open and portable format for neural model representation developed, among others, to leverage the interoperability of machine learning frameworks, and the Edge TPU uses its own native network format. The memory required for each model under different representation (i.e., on different devices) is shown in table 5. The ONNX format greatly reduces the space requirements of the models. The Edge TPU uses an optimized and special format from TensorFlow Lite, and as such had the smallest models. A single 2D CNN for the Edge TPU can fit in its modest (4 GiB) memory so that it does not need to transfer data between host and device, which is an expensive operation.

D. POWER CONSUMPTION AND ENERGY COSTS

The power consumption associated with PD detection was studied on all devices, too. Table 6 shows the average power consumption (energy) per 1,000 inferences on the NVIDIA

TABLE 3. Average inference time (ms).

Device	GTX 1070	Jet. Nano	Jet. Xavier NX	Jet. AGX Xavier	Edge TPU
Single 2D CNN	71.67	257.87	204.35	146.92	**116.86**
2D CNN ensemble	93.43	426.00	353.81	234.53	489.35
ResNet50V2	109.43	653.45	394.82	283.52	669.25
ResNet152V2	155.69	810.18	423.66	313.53	N/A
1D CNN ensemble	73.09	610.35	604.94	383.42	N/A

TABLE 4. Average inference time (ms) for parts of Ensembled 1D.

Device	Autoencoder	1D CNN	Wide & Deep	Ensembled 1D
GeForce GTX 1070	62.52	64.33	53.90	**73.09**
Jetson Nano	161.32	467.03	236.86	**610.35**
Jetson Xavier NX	166.11	459.97	247.44	**604.94**
Jetson AGX Xavier	104.92	294.33	158.92	**383.42**

TABLE 5. Model sizes in different representations (MiB).

Neural network	Keras	ONNX	Edge TPU
1D CNN ensemble	1.1	0.2	N/A
Single 2D CNN (averaged)	29	9.5	2.5
2D CNN ensemble	48	48	12.3
ResNet50V2	271	90	23.4
ResNet152V2	670	223	58.1

TABLE 6. Average power consumption (mW).

	Jetson model		
Neural network	Nano	Xavier NX	AGX Xavier
1D CNN ensemble	2826	3982	9088
Single 2D CNN	3639	4907	10662
2D CNN ensemble	3013	4157	9288
ResNet50V2	3651	5083	10747
ResNet152V2	3534	4798	10331

TABLE 7. Peak power consumption (mW).

	Device			
Neural network	Jetson Nano	Jetson Xavier NX	Jetson AGX Xavier	Edge TPU
1D CNN ensemble	3332	4205	9455	N/A
Single 2D CNN	4712	5316	10032	2024
2D CNN ensemble	3256	6420	10418	2384
ResNet50V2	5237	8125	14858	2415
ResNet152V2	5111	8533	13967	N/A

Jetson modules. The power consumption was measured by onboard sensors using the popular jtop application. Although this software-based measurement is not as precise as a hardware-based measurement would be, it provides a solid basis for the comparison of the energy costs of the inference on different devices.

The peak power consumption per 1000 inferences is shown in table 7. On the Jetson modules, it was again measured using jtop. On the Edge TPU, we used a USB power meter which recorded only the current power consumption so only the peak power consumption could be reliably determined. Clearly, the Edge TPU had the lowest peak power consumption while Jetson AGX Xavier had the highest peak power consumption.

In addition, the energy needed to perform the inference on every device by each model was compared. Table 8 provides the approximate values of energy (in mJ) consumed by a single inference. For the Jetson modules, it was computed on the basis of the average power consumption. For the Edge TPU, where average power consumption was not available, the peak power consumption was used instead to estimate the needed energy in the worst-case scenario. However, it should be noted that the peak and average consumption of the Edge TPU are not expected to be much different. The lowest energy per inference was consumed by the Edge TPU using a single 2D CNN model. The highest energy consumption was observed on Jetson AGX Xavier with 4379 mJ per inference, on average.

TABLE 8. Approximated energy consumed per inference (mJ).

	Device			
Neural network	Jetson Nano	Jetson Xavier NX	Jetson AGX Xavier	Edge TPU
1D CNN ensemble	2033	2543	3625	N/A
Single 2D CNN	1215	1086	1473	**236**
2D CNN ensemble	1387	2271	2443	1166
ResNet50V2	3422	3207	4212	1616
ResNet152V2	4140	3615	4379	N/A

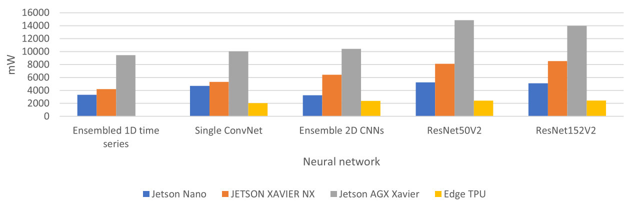


FIGURE 13. Peak power consumption.

VII. DISCUSSION

The results, presented in section VI, can be discussed in a broader context.

A. TIME SERIES-BASED VS. SPECTROGRAM-BASED PD DETECTION

The two high-level approaches to online PD detection evaluated in this work are the analysis of raw data (1D time series analysis) and spectrogram analysis. Several neural models for both types of detections were studied from different points of view.

The comparison between the 1D time series approach and the spectrogram-based approach was rather complicated due to the specific properties of some models and the limitations of the test devices. Moreover, both approaches require different data pre-processing. The time complexity of the straightforward data preparation for the 1D time series analysis is linear, with $O(n)$, while the second approach requires more complex steps including spectrogram creation that involves expensive steps, e.g., the Fast Fourier Transformation. In addition, max-pooling can be easily included in neural network models. On the other hand, models for the 1D time-series analysis featured generally lower accuracy, illustrated by lower MCC.

The spectrogram-based approach was better in terms of accuracy and energy efficiency, overall. It outperformed the time-series-based approach on all devices, even when only a single CNN was used for PD detection. Data pre-processing (spectrogram construction) can be done by the accelerator on NVIDIA Jetson SoCs but requires computation on the host when using the Edge TPU.

Interestingly, the time series-based approach achieved a better average inference time on a classical GPU (GeForce 1070) than on edge devices compared to spectrogram-based PD detection, where Single 2D CNN is 2x faster than the 1D CNN ensemble and on Geforce 1070 the average inference time is very similar. This could theoretically be attributed to the fact that the operations used in the time-series-based approach are better supported on classical GPUs than on edge devices.

The power consumption and energy requirements were approximated using different kinds of power consumption measurements (HW and SW-based) so the obtained results ought to be assessed with caution.

B. NEURAL MODELS FOR SPECTROGRAM-BASED PD DETECTION

A number of different neural models for spectrogram-based PD detection was designed and evaluated. Single networks achieved reasonable results at a lower space complexity. That makes them suitable for the most constrained devices such as the Edge TPU where they can operate without extensive host-to-device data transfers.

To execute more complex models, for example, CNN ensembles, on Edge TPU, distributed computation (e.g., TPU pipelining) is possible to allow the use of models with the best accuracy on architectures with TPUs. On the other hand, the approximate power consumption of CNNs on 5 Edge TPUs is comparable to the consumption of a single 2D CNN (5×236 vs 1166), excluding communication overhead. The ResNet model achieved worse results than the CNN ensemble in terms of MCC and power consumption. On the other hand, the increased depth of the network by 102 layers did not radically affect inference speed.

Only the Edge TPU used quantization. This could affect the performance of the models but several studies confirmed that this fact should not degrade the performance of other devices and the comparison is sufficiently fair and robust [60], [61].

C. NEURAL NETWORK ENSEMBLING FOR PD DETECTION

The experiments confirmed the validity of the basic assumptions behind model ensembling in the PD detection use case. They have clearly shown that stacking ensembles of CNNs allow a more accurate PD detection than individual networks (weak learners) and standalone neural models. It also illustrated the good potential of such models for accelerated execution on suitable edge devices. For example, the inference of the 1D CNN ensemble was more than 2 times faster than the sequential execution of the networks in the ensemble on Jetson Nano (610 vs 1494 ms).

This observation applies to the spectrogram-based approach as well. The 2D CNNs ensemble achieved better accuracy than the underlying networks alone and ran much faster (1290 ms if run sequentially vs 426 ms in ensemble). The accuracy of the ensemble was also better than the accuracy of the underlying models.

VIII. CONCLUSION

This work provides an extensive study on the accuracy and performance of different deep neural models for partial discharge detection on edge devices. Several models consisting of standalone and ensembled deep neural networks were designed, implemented, and thoroughly evaluated on a handful of commercially available edge devices capable of neural network inference acceleration. The comparison was based on a data set of signals obtained from covered conductors in several locations of the Czech Republic by a contactless antenna. The detection was done either from the raw signal or from spectrograms extracted from the noisy raw data. The accuracy, latency, and approximate power consumption per inference were evaluated. The results highlight the importance of the selection of suitable PD detection models. They show that stacking ensembles of 2D CNNs are efficient in terms of both, accuracy and energy consumption. The overall best accuracy was achieved by the stacking ensemble of CNNs with the Wide & Deep network as meta-learner used for the spectrogram-based PD detection (2D CNN ensemble). Second to best results were obtained by the widely used ResNetV2 neural network with 50 layers. Both models were also able to run on all test devices, including the highly restricted Edge TPU. This is in contrast with the larger ResNet152V2 that was not able to finish on the TPU the inference and the 1D CNN ensemble that included an autoencoder, a network requiring deconvolution, an operation not available in the limited environment of the TPU. The experiments demonstrated that the stacking ensemble of lightweight neural networks (2D CNN ensemble) was on all devices executed faster than the more complex model (ResNet50V2) and underpin the importance of experimental development of custom models tailored for particular applications.

REFERENCES

[1] B. Chen, J. Wan, A. Celesti, D. Li, H. Abbas, and Q. Zhang, "Edge computing in IoT-based manufacturing," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 103–109, Sep. 2018.

[2] I. Sitton-Candanedo, R. S. Alonso, S. Rodriguez-Gonzalez, J. A. G. Coria, and F. D. L. Prieta, "Edge computing architectures in Industry 4.0: A general survey and comparison," in *Proc. Int. Workshop Soft Comput. Models Ind. Environ. Appl.* (Advances in Intelligent Systems and Computing). Cham, Switzerland: Springer, May 2019, pp. 121–131.

[3] V. K. Prasad, M. D. Bhavsar, and S. Tanwar, "Influence of montoring: Fog and edge computing," *Scalable Comput., Pract. Exper.*, vol. 20, no. 2, pp. 365–376, May 2019.

[4] S. Cass, "Nvidia makes it easy to embed AI: The Jetson Nano packs a lot of machine-learning power into DIY projects—[Hands on]," *IEEE Spectr.*, vol. 57, no. 7, pp. 14–16, Jul. 2020.

[5] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, pp. 637–646, 2016.

[6] J. Mocnej, M. Miskuf, P. Papcun, and I. Zolotová, "Impact of edge computing paradigm on energy consumption in IoT," *IFAC-PapersOnLine*, vol. 51, no. 6, pp. 162–167, 2018.

[7] L. Kljucaric, A. Johnson, and A. D. George, "Architectural analysis of deep learning on edge accelerators," in *Proc. IEEE High Perform. Extreme Comput. Conf. (HPEC)*, Sep. 2020, pp. 1–7.

[8] A. A. A. Jainuddin, Y. C. Hou, M. Z. Baharuddin, and S. Yussof, "Performance analysis of deep neural networks for object classification with edge TPU," in *Proc. 8th Int. Conf. Inf. Technol. Multimedia (ICIMU)*, Aug. 2020, pp. 323–328.

[9] S. Valladares, M. Toscano, R. Tufino, P. Morillo, and D. Vallejo-Huanga, "Performance evaluation of the Nvidia Jetson Nano through a real-time machine learning application," in *Intelligent Human Systems Integration*, D. Russo, T. Ahram, W. Karwowski, G. Di Bucchianico, and R. Taiar, Eds. Cham, Switzerland: Springer, 2021, pp. 343–349.

[10] S. Mittal, "A survey on optimized implementation of deep learning models on the Nvidia Jetson platform," *J. Syst. Archit.*, vol. 97, pp. 428–442, Jan. 2019.

[11] S. Sadiq, M. Zmieva, M.-L. Shyu, and S.-C. Chen, "Reduced residual nets (Red-Nets): Low powered adversarial outlier detectors," in *Proc. IEEE Int. Conf. Inf. Reuse Integr. (IRI)*, Jul. 2018, pp. 436–443.

[12] A. A. Suzen, B. Duman, and B. Sen, "Benchmark analysis of Jetson TX2, Jetson Nano and raspberry PI using deep-CNN," in *Proc. Int. Congr. Human-Comput. Interact., Optim. Robotic Appl. (HORA)*, Jun. 2020, pp. 1–5.

[13] P. Puchtler and R. Peinl, "Evaluation of deep learning accelerators for object detection at the edge," in *Proc. German Conf. Artif. Intell.* (Lecture Notes in Computer Science). Cham, Switzerland: Springer, 2020, pp. 320–326.

[14] P. Kang and J. Jo, "Benchmarking modern edge devices for AI applications," *IEICE Trans. Inf. Syst.*, vol. 104, no. 3, pp. 394–403, Mar. 2021.

[15] S. Misak, J. Fulnecek, T. Jezowicz, T. Vantuch, and T. Burianek, "Usage of antenna for detection of tree falls on overhead lines with covered conductors," *Adv. Electr. Electron. Eng.*, vol. 15, no. 1, pp. 21–27, Mar. 2017.

[16] L. Klein, D. Seidl, J. Fulnecek, L. Prokop, S. Misák, and J. Dvorský, "Antenna contactless partial discharges detection in covered conductors using ensemble stacking neural networks," *Exp. Syst. Appl.*, vol. 213, Mar. 2023, Art. no. 118910.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Computer Vision—ECCV*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham, Switzerland: Springer, 2016, pp. 630–645.

[18] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah, "Wide & deep learning for recommender systems," in *Proc. 1st Workshop Deep Learn. Recommender Syst.*, New York, NY, USA, 2016, pp. 7–10.

[19] L. Voldhaug, "MV overhead lines using XLPE covered conductors. Scandinavian experience and NORWEB developments," in *Proc. IEE Colloq. Rev. Options Overhead Rural Distrib.*, 1995, pp. 52–60.

[20] P. Pakonen, *Detection of Incipient Tree Faults on High Voltage Covered Conductor Lines*. Tampere, Finland: Tampere Univ. Technology, 2007.

[21] T. Leskinen, "Finnish and Slovene experience of covered conductor overhead lines," in *Proc. CIGRÉ*, 2004, pp. 54–60.

[22] R. Bartnikas, "Partial discharges. Their mechanism, detection and measurement," *IEEE Trans. Dielectr. Electr. Insul.*, vol. 9, no. 5, pp. 763–808, Oct. 2002.

[23] O. Kabot, J. Fulnecek, S. Misak, L. Prokop, and J. Vaculik, "Partial discharges pattern analysis of various covered conductors," in *Proc. 21st Int. Sci. Conf. Electric Power Eng. (EPE)*, Oct. 2020, pp. 1–5.

[24] G. C. Stone, "Partial discharge diagnostics and electrical equipment insulation condition assessment," *IEEE Trans. Dielectr. Electr. Insul.*, vol. 12, no. 5, pp. 891–903, Oct. 2005.

[25] S. Misák, S. Hamacek, and M. Bartlomiejczyk, "Verification of a novel method of detecting faults in medium-voltage systems with covered conductors," *Metrology Meas. Syst.*, vol. 24, no. 2, pp. 277–288, Jun. 2017.

[26] K. Chen, T. Vantuch, Y. Zhang, J. Hu, and J. He, "Fault detection for covered conductors with high-frequency voltage signals: From local patterns to global features," *IEEE Trans. Smart Grid*, vol. 12, no. 2, pp. 1602–1614, Mar. 2021.

[27] D. Ahmad, S. Wang, and M. Alam, "Long short term memory based deep learning method for fault power line detection in a MV overhead lines with covered conductors," in *Proc. 21st Nat. Power Syst. Conf. (NPSC)*, Dec. 2020, pp. 1–4.

[28] T. Vantuch, M. Prílepok, J. Fulnecek, R. Hrbác, and S. Misák, "Towards the text compression based feature extraction in high impedance fault detection," *Energies*, vol. 12, no. 11, p. 2148, Jun. 2019.

[29] M. Kratky, S. Misak, P. Gajdos, P. Lukas, R. Baca, and P. Chovanec, "A novel method for detection of covered conductor faults in medium voltage overhead line systems," *IEEE Trans. Ind. Electron.*, vol. 65, no. 1, pp. 543–552, Jan. 2018.

[30] K. Akyol, "Stacking ensemble based deep neural networks modeling for effective epileptic seizure detection," *Exp. Syst. Appl.*, vol. 148, Jun. 2020, Art. no. 113239.

[31] M. Shorfuzzaman, "IoT-enabled stacked ensemble of deep neural networks for the diagnosis of COVID-19 using chest CT scans," *Computing*, vol. 105, no. 4, pp. 887–908, Jun. 2021.

[32] F. Perez, S. Avila, and E. Valle, "Solo or ensemble? Choosing a CNN architecture for melanoma classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2019, pp. 2775–2783.

[33] M. Kim, M. Lee, M. An, and H. Lee, "Effective automatic defect classification process based on CNN with stacking ensemble model for TFT-LCD panel," *J. Intell. Manuf.*, vol. 31, no. 5, pp. 1165–1174, Oct. 2019.

[34] M. Mohammed, H. Mwambi, I. B. Mboya, M. K. Elbashir, and B. Omolo, "A stacking ensemble deep learning approach to cancer type classification based on TCGA data," *Sci. Rep.*, vol. 11, no. 1, pp. 1–22, Aug. 2021.

[35] R. Gupta, "Stacking ensemble of convolutional neural networks for sign language recognition," in *Proc. Int. Conf. Comput. Commun. Informat. (ICCCI)*, Jan. 2022, pp. 1–5.

[36] M. Middlehurst, J. Large, M. Flynn, J. Lines, A. Bostrom, and A. Bagnall, "HIVE-COTE 2.0: A new meta ensemble for time series classification," *Mach. Learn.*, vol. 110, pp. 3211–3243, Dec. 2021.

[37] D. Vasan, M. Alazab, S. Wassan, B. Safaei, and Q. Zheng, "Image-based malware classification using ensemble of CNN architectures (IMCEC)," *Comput. Secur.*, vol. 92, May 2020, Art. no. 101748.

[38] I. H. Sarker, "Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions," *Social Netw. Comput. Sci.*, vol. 2, no. 6, p. 420, Aug. 2021.

[39] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proc. IEEE*, vol. 107, no. 8, pp. 1655–1674, Aug. 2019.

[40] Y. Deng, "Deep learning on mobile devices: A review," in *Proc. SPIE*, vol. 10993, pp. 52–66, May 2019.

[41] A. Yazdanbakhsh, K. Seshadri, B. Akin, J. Laudon, and R. Narayanaswami, "An evaluation of edge TPU accelerators for convolutional neural networks," 2021, *arXiv:2102.10423*.

[42] M. P. Véstias, R. P. Duarte, J. T. de Sousa, and H. C. Neto, "Moving deep learning to the edge," *Algorithms*, vol. 13, no. 5, p. 125, May 2020.

[43] M. Verhelst and B. Moons, "Embedded deep neural network processing: Algorithmic and processor techniques bring deep learning to IoT and edge devices," *IEEE Solid-State Circuits Mag.*, vol. 9, no. 4, pp. 55–65, Nov. 2017.

[44] Y. Chen, B. Zheng, Z. Zhang, Q. Wang, C. Shen, and Q. Zhang, "Deep learning on mobile and embedded devices: State-of-the-art, challenges, and future directions," *ACM Comput. Surveys*, vol. 53, no. 4, pp. 1–37, Aug. 2020.

[45] J. Fulnecek and S. Misak, "A simple method for tree fall detection on medium voltage overhead lines with covered conductors," *IEEE Trans. Power Del.*, vol. 36, no. 3, pp. 1411–1417, Jun. 2021.

[46] W. Tang, G. Long, L. Liu, T. Zhou, J. Jiang, and M. Blumenstein, "Rethinking 1d-cnn for time series classification: A stronger baseline," 2020, *arXiv:2002.10061*.

[47] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[48] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[49] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," 2018, *arXiv:1801.04381*.

[50] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794.

[51] F. Chollet. (2015). *Keras*. [Online]. Available: https://keras.io

[52] S. P. Baller, A. Jindal, M. Chadha, and M. Gerndt, "DeepEdgeBench: Benchmarking deep neural networks on edge devices," in *Proc. IEEE Int. Conf. Cloud Eng. (ICE)*, Oct. 2021, pp. 20–30.

[53] D. Xu, C. Liu, Y. Wang, K. Tu, B. He, and L. Zhang, "Accelerating generative neural networks on unmodified deep learning processors—A software approach," *IEEE Trans. Comput.*, vol. 69, no. 8, pp. 1172–1184, Aug. 2020.

[54] J. Bai. (2019). *ONNX: Open Neural Network Exchange*. [Online]. Available: https://github.com/onnx/onnx

[55] M. Sever and S. Ogut, "A performance study depending on execution times of various frameworks in machine learning inference," in *Proc. 15th Turkish Nat. Softw. Eng. Symp. (UYMS)*, Nov. 2021, pp. 1–5.

[56] P. Hurni, B. Nyffenegger, T. Braun, and A. Hergenroeder, "On the accuracy of software-based energy estimation techniques," in *Wireless Sensor Networks* (Lecture Notes in Computer Science). Berlin, Germany: Springer, 2011, pp. 49–64.

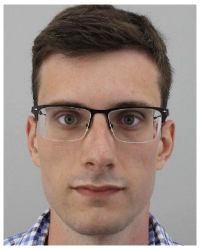
[57] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, no. 1, pp. 1–13, Jan. 2020.

[58] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997.

[59] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, A. Müller, J. Nothman, G. Louppe, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2012.

[60] Z. Shi, "Optimized YOLOv3 deployment on Jetson TX2 with pruning and quantization," in *Proc. IEEE 3rd Int. Conf. Frontiers Technol. Inf. Comput. (ICFTIC)*, Nov. 2021, pp. 62–65.

[61] G. Plastiras, S. Siddiqui, C. Kyrkou, and T. Theocharides, "Efficient embedded deep neural-network-based object detection via joint quantization and tiling," in *Proc. 2nd IEEE Int. Conf. Artif. Intell. Circuits Syst. (AICAS)*, Aug. 2020, pp. 6–10.



LUKÁŠ KLEIN received the B.Sc. and M.Sc. degrees in computer science from the VSB—Technical University of Ostrava, Czech Republic, in 2019 and 2021, respectively, where he is currently pursuing the Ph.D. degree. His main interest is focused on the classification of partial discharges in cooperation with the ENET Centre, where he also works. His research interests include time series classification, machine learning, deep learning, and embedded development.

PETR ŽMIJ received the degree from the Technical College Kopřivnice, Czech Republic, in 1996. Since 1996, he has been working in various expert and management positions in production companies in the automotive industry sector. He has professional experience in manufacturing, production technology, and industrial engineering. His research interests include production and technology in the automotive industry supply chain.

PAVEL KRÖMER (Senior Member, IEEE) received the Ph.D. degree from the VSB—Technical University of Ostrava, Czech Republic, in 2010. He was a Postdoctoral Fellow with the University of Alberta, Edmonton, Canada, in 2014. He was a Researcher with the IT4Innovations National Supercomputing Center, VSB—Technical University of Ostrava, from 2011 to 2016, where he is currently an Associate Professor. His research interests include artificial intelligence, evolutionary computation, machine learning, and real-world applications of intelligent methods.