Review

International Journal of Distributed Sensor Networks

International Journal of Distributed Sensor Networks
2019, Vol. 15(8)

DOI: 10.1177/1550147719867072
journals.sagepub.com/home/dsn
SAGE

Deep learning in the fog

Andrzej Sobecki[1], Julian Szymański[1], David Gil[2] and Higinio Mora[2]

Abstract

In the era of a ubiquitous Internet of Things and fast artificial intelligence advance, especially thanks to deep learning networks and hardware acceleration, we face rapid growth of highly decentralized and intelligent solutions that offer functionality of data processing closer to the end user. Internet of Things usually produces a huge amount of data that to be effectively analyzed, especially with neural networks, demands high computing capabilities. Processing all the data in the cloud may not be sufficient in cases when we need privacy and low latency, and when we have limited Internet bandwidth, or it is simply too expensive. It poses a challenge for creating a new generation of fog computing that supports artificial intelligence and selects the architecture appropriate for an intelligent solution. In this article, we show from four perspectives, namely, hardware, software libraries, platforms, and current applications, the landscape of components used for developing intelligent Internet of Things solutions located near where the data are generated. This way, we pinpoint the odds and risks of artificial intelligence fog computing and help in the process of selecting suitable architecture and components that will satisfy all requirements defined by the complex Internet of Things systems.

Keywords

Internet of Things, fog computing, edge computing, deep neural networks

Date received: 10 May 2019; accepted: 8 July 2019

Handling Editor: Wei Wei

Introduction

The Internet of Things (IoT)[1] is a paradigm of interrelated small devices such as radio-frequency identification (RFID) tags, sensors, actuators, and smartphones which are pervasive, and through the use of unique identifiers they have the ability to automatically connect with each other to achieve a common goal. A thing in the IoT can be a heart monitor implant, a farm animal with a biochip transponder, a vehicle which transmits information about traffic jams, an automobile that has built-in sensors to alert the driver when the tire pressure is low, or any other object that can be assigned an IP address and is able to transfer data over a network (https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT).

The role of the IoT device depends on the purpose and type of processed data. Often, the character of the data generated by devices does not allow effective processing in the cloud[2] (due to requirements on latency, privacy, or due to their amount) so it needs to be analyzed automatically where it is produced. Moreover, even if we transfer data to the cloud, we may have to face additional requirements concerning, for example, privacy, security, budget, or time.

One of the fast developing trends in intelligent data analysis is deep neural networks (DNNs). They offer a wide range of machine learning (ML) methods that often outperforms the state-of-the-art models. To be effectively used, deep models have special requirements such as on high computing resources or a large amount of data examples. Thus, in this article, we focus on this

[1]Gdańsk University of Technology, Gdańsk, Poland
[2]University of Alicante, Alicante, Spain

Corresponding author:
Julian Szymański, Gdańsk University of Technology, Narutowicza 11/12 Street, Gdańsk 80233, Poland.
Email: julian.szymanski@eti.pg.edu.pl

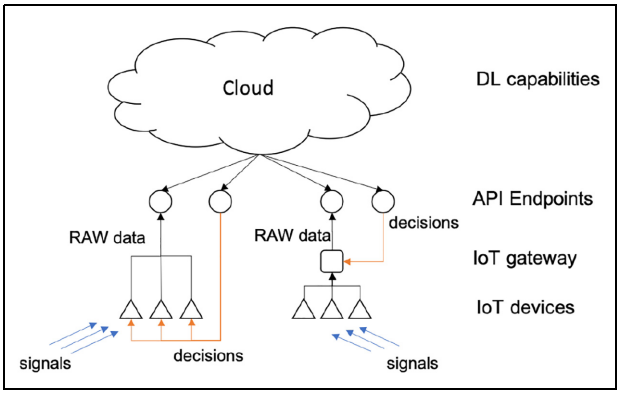

Figure 1. The current model of usage deep learning models in the IoT—example of thin edge devices.

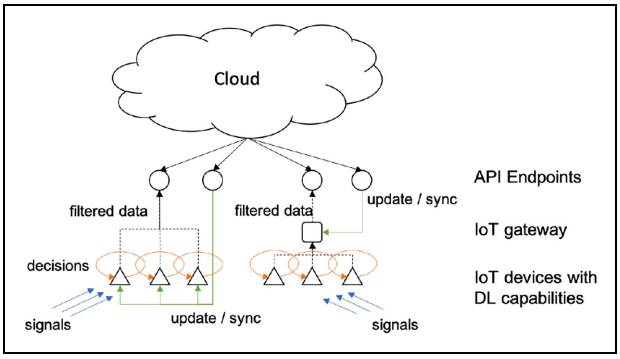

Figure 2. Model of deep learning on the edge—example of fat edge devices.

type of ML algorithms that are embedded in the IoT environment.

Usually, most computationally intensive tasks are performed outside the edge devices, for example, in the cloud (see Figure 1) or on dedicated servers. Edge devices were created originally to filter and transfer data to the cloud. Platforms like Google Cloud or AWS offer tools to accelerate models for data analysis fed with data from edge devices. The main problems with the cloud are the high latency of the decision-making process, costs of transmission, and limited privacy of transmitted data.

In order to solve the problems mentioned above, there was proposed a fog computing paradigm in which we create some proxy layer between edge devices and the cloud. Fog computing (https://internetofthingsagenda.techtarget.com/definition/fog-computing-fogging), also known as fog networking or fogging, is a decentralized computing infrastructure in which data, computation, storage, and applications are distributed in the most logical, efficient place between the data source and the cloud. This layer is created by a set of computing nodes that are placed near the source of data. The computing nodes are usually provided by the company which is the owner of the data. This architecture increases privacy and security, and reduces costs of transmitting data from the edge. Moreover, the data owner may achieve lower processing latency as a result of reducing the distance between computing nodes and data sources.

The current trends point to the next-generation fog computing paradigm which is based on the edge devices (see Figure 2). To this day, edge devices perform usually only some preprocessing, for example, compression of data or sending them to the cloud or to the fog for wider analyses. The most power-consuming tasks cannot be efficiently run on the low-power and low-performance edge devices. This highly handicaps the ability to analyze the incoming information and make decisions in real time. Currently, we can see a breakthrough in edge computing, as those devices become more and more efficient and capable, especially for usage of sophisticated data analysis models in the edge. In some cases, the edge device should process data without an Internet connection which may be crucial in dynamic environments such as vehicles and airplanes. Generally, efficient edge devices are dedicated to the environments where we need high responsiveness of algorithms based on huge data streams created in real time. Thanks to progress in hardware for mobile applications, we are already able to efficiently use deep models for inference on the edge devices. This provides us the possibility to create a new generation of fog and eliminate the proxy layer through using edge devices as computing nodes. Nevertheless, training still takes place in the cloud, and management and maintenance of multiple devices with weak Internet connections are still difficult tasks. Fog computing models, especially those designed for the ML computing, require dedicated devices and approaches for creating cohorts of those devices.

In this article, we describe the advantages and disadvantages of available devices, software, and platforms. Moreover, we present some use cases and configurations which could be selected by the reader. Based on that, we may use the presented components and configure architecture for a new application according to the requirements and limitations imposed by the fog and hybrid environment. This article is organized as follows: In section "Architecture of IoT solutions," we propose to organize IoT devices into two main groups: thin and fat devices. To better chose devices appropriate for the new application, we propose also a partition of IoT application architectures based on two factors: the size of data to take the decision and required responsiveness. Examples of fat IoT devices are described in section "Hardware." The software which we could use to delegate ML tasks to IoT devices is presented in section "Software libraries." We decide to

distinguish software and platforms because the first one is installed on the IoT devices and provides the possibility to deploy models and the core functionality. On the other side, we propose to separate platforms as a set of tools which are on the top of the IoT devices. Platforms enable the possibility of creating hybrid fogs and automating the maintenance tasks. We describe them in detail in section "IoT platforms." Finally, in section "Applications," we present examples of applications that employ ML for data from IoT devices. In section "Conclusion," we summarize the state-of-the-art solutions and identify future directions of development and usage of ML in IoT environments.

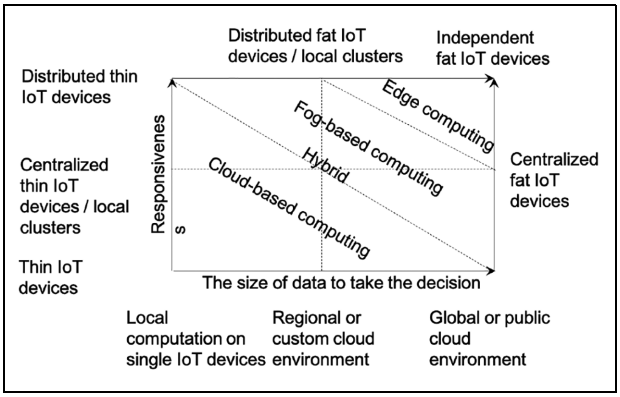

Figure 3. The proposed division of the cloud and fog architectures.

Architecture of IoT solutions

The decision about which architecture is appropriate for our application may depend on many factors. Basically, the costliness of the solution and the technology used may have the greatest impact on the choice of architecture,[3] but however the costs of creating and maintaining architecture can be a surety of other, much more important factors. In order to introduce the problem of selecting architecture, we propose two basic factors:

- Size of data—volume of data which we should collect to prepare a decision with a desired accuracy level;
- Responsiveness—how much time we need to produce a decision with a desired accuracy level.

For example, let it be assumed that we have a home with a set of basic sensors for measuring the humidity, temperature, and light intensity, and we wish to prepare the decision about whether the window should be open or closed. If we will analyze signals from the last minute, then the size of data needed to perform the decision will be very low. The accepted responsiveness is also very low which means that we accept high latency because this type of decision is not crucial. In the worst case, the house will be ventilated slightly later.

A completely different example is the autonomous vehicle in which, despite a set of sensors, we should consider data from cameras. This creates a large volume of data which we have to analyze in order to prepare multiple decisions at once. In this situation, we also expect high responsiveness because delays are unacceptable and may threaten the lives of passengers.

The presented examples are contrasting and it is clear that they use different architectures to make decisions. But we could point out many applications which are between these two. The development of the IoT technology has resulted in the fact that many devices with different architectures and capabilities are available on the market today. In order to better organize this set of devices, we propose to divide them into two main groups:

Thin devices. Basically, these are created to collect data from sensors without the capability to perform computation in an efficient way. The main advantages of this type are the small size of the device, low energy consumption, simple CPU, and low onboard memory. Examples of thin devices are Raspberry Pi, Raspberry Zero, and Arduino.
Fat devices. These are specialist equipment dedicated to performing calculations in specific applications, for example, neural networks and analyzing video stream. This is achieved by incorporating special components such as GPU and TPU (tensor processing units). Examples from this class are devices based on Google TPU or NVIDIA Xavier platforms.

In Figure 3, we propose partition of IoT architectures using the abovementioned factors.

In many basic solutions in which data are not so large and we can wait for decisions, we should think about the usage of the basic thin IoT devices as computational nodes. In that model, we assume that most computations are performed locally and data may be additionally stored in the cloud, but it is not required. When we need some coordination and to process larger volumes of data we should consider the usage of a local (custom) or regional cloud environment. Finally, we could use infrastructure from global cloud providers for computation and to store the collected data. The decision about which architecture is better in many cases will depend on the scale in which we deploy our application.

On the contrary, we may wish to use multiple IoT devices to collect data and perform some computation. Then we should use some local coordination point such

as gateways or hubs in which we process data from multiple sensors at once. We can imagine a more sophisticated solution, for example, when we need a low-latency decision-making process, and have to process data from multiple devices but select them depending on the current context. In that situation, we should consider creating a distributed environment of IoT devices that are context aware and have the ability to communicate with neighbor devices in an autonomous way. In most sophisticated solutions, the IoT devices may be deployed across the region, so we will usually need a platform for management of the IoT devices and collecting important data from them. The presented solutions assumed that we need cloud-based computing to support the IoT thin devices.

The right-hand side of Figure 3 presents use cases in which we have too much data to transfer, we cannot transfer data for security or economic reasons, or high latency is not acceptable in our business model. In those cases, we are unable to use the cloud for each request and should improve our local infrastructure to achieve the desired result. Basically, we can create hybrid solutions in which the core computations are prepared in the fog and the rest are computed in the cloud. In that situation, not all data are transferred from the edge to the cloud. The mentioned problems are currently solved using the fog computing paradigm.

The biggest challenge is applications that require real-time or near-real-time results and need the continuous delivery of large data streams. Moreover, if we assumed that the IoT devices are moving and have poor Internet connection, we have an example of a use case in which cloud computation is unavailable. In those situations, we must use fat IoT devices which could prepare results in a short time without external providers. For business models that assume achieving low latency and high accuracy based on large data streams, we should consider the usage of new-generation fog computing, such a set of efficient edge devices, or a single autonomous edge device.

Despite the selected architecture, we will still need support in management tasks such as upgrade or monitoring devices, especially including scenarios when devices may be visible in the Internet from time to time. The considerations so far have focused on adapting the IoT architecture in terms of performance. Another important aspect of the usage of fog computing for ML computations is improving the security of data processing which is crucial in many fields, that is, in hospital, industry, or military applications. Depending on the required values of the mentioned factors, we should choose the appropriate architecture for them. The contribution of this article is a structured collection of information for the user about how, and when, he or she can use fog computing models in their organizations and which components he or she may use to achieve that.

Hardware

In the market, there exist many devices that are ready to support processing data based on neural network models in the IoT nodes. We propose to divide them into three classes:

- Universal components to build the complete IoT node with DL capabilities, which represent devices that can be used as a core component when we wish to create a new IoT node with DL capabilities;
- Components for extending the capabilities of an existing IoT node, which include all devices that can be connected to an existing IoT node and increase the computing power needed for efficiently processing the DL models;
- Components dedicated to create an end-user IoT node, which are the devices created for the specified usage and usually projected to use as an independent fat IoT node.

In this section, we describe the main examples of devices available in the market, grouped according to the given classification.

Universal components

Movidius Myriad VPU. The Intel Company proposes a specialist processor family "Myriad" which can accelerate the inference of a DNN model. Currently, the family is created by two processors Myriad 2 and Myriad X that differ in speed. Both offer acceleration with low power consumption and have the size adequate to use them in existing devices. Myriad 2 (https://www.movidius.com/myriad2) offers for the user 12 128-bit vector processors which are optimized for machine vision. The Intel Company describes its processor as a device which offers 1 TFLOPS within a nominal 1 W power consumption. Communication with the vision processing unit (VPU) is possible via interfaces such as USB3 and 1 GB Ethernet.

The Myriad X (https://www.movidius.com/myriadx) processor offers 16 128-bit vector processors with theoretical performance of 4 TOPS (trillion operations per second). New interface (MIPI Lanes with ISP) in the Myriad X VPU supports up to eight RGB sensors with resolution up to 700 million pixels per second. The processor supports video encoders for 4K videos (H.264/H.265 for 30 Hz and M/JPEG for 60 Hz). Communication with the processor is available via interfaces such as USB 3.1 and PCI-E Gen 3. In order

to adjust the neural network model to the Myriad X VPU, we can use the SDK which includes an NN compiler. This compiler accepts as input the file neural networks created by Caffe and TensorFlow.

Google Edge TPU. Google Edge TPU (https://cloud.google.com/edge-tpu/) is a device which can act as a co-processor to accelerate the process of train models on low-power devices. Currently, the Edge TPU is ready to run classification models which are retrained on the device using the technique proposed in Qi et al.[4] In order to run the NN model in the TPU, the user has to convert the TensorFlow Lite model to TensorFlow Lite TPU. Models created in PyTorch may be converted using the ONNX library. Theoretical efficiency (https://cloud.google.com/edge-tpu/) of Google TPU may be measured as the number of multiple state-of-the-art artificial intelligence (AI) models that can be running simultaneously on the video stream with 30 fps. Google's TPUs may be supported by Cloud IoT Edge to manage AI models on the edge.

A ready-to-use device with Google Edge TPU is offered as the Coral Dev Board (https://coral.withgoogle.com/docs/dev-board/datasheet/). It integrates Quad-core Cortex-A53 as a CPU and Google Edge TPU as a co-processor. If we need an expansion set in the USB-form device, we can build, for example, Coral USB Accelerator (https://coral.withgoogle.com/docs/accelerator/get-started/).

ARM ML processor. The ARM ML processor is created to accelerate the process of training the ML models and it is an element of ARM Trillium project. A single ARM ML processor contains 16 compute engines which support 8-bit quantized integer. We can train models created in the following frameworks: TensorFlow, Caffe2, MXNet, and Android NNAPI. The ML processor is optimized for processing a convolutional and recurrent neural network (RNN). Theoretical efficiency of the ML processor is 4 TOPS at 1 GHz clock speed. The ARM Company assumed that the minimum efficiency of a processor powered by 1 W of energy should not be lower than 3 TOPS. We can use the ARM ML processor to build low-energy IoT devices, but also we can multiply the number of processors and build an efficient low-energy cluster of ML processors.

The most important feature of the new ARM processors is the functionality to translate layers of NN between major frameworks. This creates opportunities to ease creation of multidisciplinary NN based on existing neural networks, each of which is prepared to realize some dedicated task.

Battery-powered AI CPU. Typically, the software performs executed public-key encryption. However, the necessity of connecting the IoT to a number of different sensors makes this intelligent technology become quite complicated. The chip was designed using the elliptic-curve encryption technique. The curves have different properties and use different prime numbers. In addition, the chip features a general-purpose processor to save energy and handle encrypted data and a datagram transport layer security protocol. There is a high probability that it can provide better security for the IoT.

Usually, computation based on neural networks requires a lot of energy, and it creates a problem if we wish to use a neural network outside the office, where we have a limited number of energy sources. MIT researchers developed a new CPU (http://news.mit.edu/2018/chip-neural-networks-battery-powered-devices-0214) designed to reduce the power consumption of neural networks as well as a chip designed to perform public-key encryption for the IoT. This chip was created to reduce the power consumption of neural networks by up to 95%. In the chip, the input values of the node are converted into electrical voltages for the calculation of dot products for multiple nodes. The goal is to speed up calculations three to seven times more than its predecessors while reducing power.

Expansion components

Movidius Neural Compute Stick. Movidius Neural Compute Stick (NCS) is produced by the Intel Company and it can be run without any need of Internet. It is shipped with a software development kit that enables rapid prototyping, validation, and deployment of DNNs, as well as profiling, tuning, and compiling a DNN on a development computer with the tools that are provided in the Intel Movidius Neural Compute SDK. The Movidius NCS' compute capability comes from Myriad 2 VPU. The user just needs to plug it into an edge device via USB and he or she is able to run deep models fast and efficiently as he or she has an efficient graphics processing unit (GPU). Movidius allows you to optimize the operation of large models such as GoogLeNet.

Intel introduced a USB device called Movidius Neural Compute Stick 2 (NCS2) that is capable of running DNNs. This convenient USB stick allows developers to add neural network capability to existing hardware and start experimenting with AI. It has the ability to do it locally on an embedded system with low power consumption, coupled with low device cost—it shows the amazing possibilities and opportunities of edge computing.[5] When a computer vision application based on neural networks is run, the complex mathematical computation is offloaded to the special chip embedded inside the NCS2. The chip is designed to accelerate the computation which makes the execution

of neural networks much faster when compared to the standard CPU.

After a model is fully trained, it is moved into production for identifying and classifying objects, which is referred to as inference. When using Intel NCS, developers take a fully trained TensorFlow model and convert it into a format understood by the processing unit embedded within the device. Through the SDK and toolkits provided by Intel, only the graph is loaded into the VPU (Intel Movidius Myriad X VPU) chip while running the rest of the code on the local CPU.

BITMAIN SOPHON solutions. Other solutions could be used in fog computing to improve the efficiency of deep learning computation. Depending on the desired efficiency, developers can select between three tensor computing processors:

- BM1680 (2 TFLOPS with precision FP32, 32 MB internal memory, and 25 W power consumption);
- BM1682 (3 TFLOPS with precision FP32, 16 MB internal memory, and 2048 processing units);
- BM1880 (1 TOPS@INT8, processing unit dual-core Cortex-A53@1.5 GHz and single-core RISCV@1 GHz).

SOPHON Neural Network Stick (NNS) is a solution similar to the Movidius NCS. It is designed to accelerate neural network processing using the processor SOPHON BM1880 and support models created in Caffe, ONNX, TensorFlow, and PyTorch frameworks. Devices have USB3.0/USB2.0 interfaces and can extend the capabilities of a computer with architecture X86_64 with Ubuntu.

Brainium SmartEdge Agile. Brainium SmartEdge Agile (https://www.brainium.com/studio) is a hardware solution equipped with multiple sensors such as microphone, gyroscope, magnetometer, pressure control, temperature, humidity, and proximity. Data can be transferred through USB type C or Bluetooth 5.0 device. The device can communicate with the Octonion IoT Intelligent Edge software platform. Besides that, the Brainium Company offers the AI Studio solution in which the user can create AI models without programming skills. The trained models may be deployed in the SmartEdge Agile directly from the AI Studio.

Dedicated components

Apple A12 Bionic chip with Neural Engine. The Apple A12 Bionic is a system on a chip (SoC) from Apple that is found in iPhone XS and XR. It was announced late 2018 and offers six cores divided into two performance cores and four power efficiency cores. Compared to the previous A11 Bionic, A12 offers a 15% improved CPU performance for the performance cores and 50% lower power consumption for the efficiency cores (both according to Apple; https://www.notebookcheck.net/Apple-A12-Bionic-SoC.331518.0.html). The chip also includes a new GPU that is advertised as 50% faster, the M12 motion co-processor and a Neural Engine with eight cores for up to 5 TOPS.

Apple Neural Engine is built for advanced, real-time ML. That means the Apple iPhone X line can recognize patterns, make predictions, and learn from experience, similar to the way people do. It is incredibly efficient, which enables it to do all kinds of work in real time, for example, users can experience immersive AR. Apple also opened the Neural Engine up to the Core ML platform, so developers can bring powerful, real-time ML to their applications. With 6.9 billion transistors, the A12 Bionic is a big chip especially compared to Snapdragon 835 (3 billion) or a Skylake desktop quad-core SoC (1.75 billion). Compared to A11, the A12 integrates 60% more transistors.

Qualcomm Snapdragon 845. The Qualcomm Snapdragon 845 Mobile Platform (or SD845) is a high-end SoC for smartphones that was introduced in early 2018 and manufactured in 10-nm LPP (Low Power Plus) fin field-effect transistor (FinFET) at Taiwan Semiconductor Manufacturing Company. It integrates 4x Kryo 385 cores (Cortex-A75) at up to 2.8 GHz and 4x Kryo 385 at 1.8 GHz. More precisaly the processor include the graphic processor Adreno 360 with offer support for immersive XR experiences, an intelligent personal assistant, and advanced vault-like security is enabled by the Snapdragon 845 mobile platform. The Hexagon 685 DSP supports calculation on a traditional, scalar CPU and offers an additional dedicated processor for vector analysis (https://www.qualcomm.-com/products/snapdragon-845-mobile-platform).

NVIDIA Jetson. NVIDIA Jetson AGX Xavier (https://developer.nvidia.com/embedded/buy/jetson-agx-xavier) is a computer dedicated for autonomous machines and vehicles which need AI computing capabilities. It contains a 512-core Volta GPU with Tensor Cores, CPU with eight cores and 16 GB of memory. The performance of the NVIDIA AGX Xavier in AI computation was estimated to be 32 TOPS. The NVIDIA Jetson AGX Xavier is a device which can be used in cars in order to analyze signals from cameras and sensors in such applications as automatically steering the car. NVIDIA promises that a new device will allow to deliver assisted driving features on Level 2 + . Volvo (https://nvidianews.nvidia.com/news/volvo-selects-nvidia-drive-for-production-cars) has decided to use Jetson

AGX Xavier in its future vehicles in order to increase safety on the road.

Samsung Exynos. Samsung Exynos Auto V9 is a computer dedicated for in-vehicle infotainment systems and can support up to four screens with full HD resolution. Besides the multimedia capabilities, the Exynos Auto V9 contains a neural processing unit (NPU) which is a processor dedicated to accelerate AI methods. Samsung Exynos 9820 is an application processor which will be available in the mobile devices. This device will have a fourth-generation CPU, an advanced LTE modem with 2.0 Gbps downlink speed and an NPU. The NPU will be up to seven times faster than its predecessor. It may be used to improve photography, movies, security, object recognition, or voice recognition (https://news.samsung.com/global/samsung-brings-on-device-ai-processing-for-premium-mobile-devices-with-exynos-9-series-9820-processor).

Neousys Nuvo-6108GC. Edge AI GPU Computing node is created for power-consuming tasks such as analysis of data streams from multiple sensors in real time. The device is dedicated to cars and used in the Baidu Apollo program in which they create solutions for autonomous driving. The main advantage of the Nuvo is the possibility to use GPU to accelerate the process of usage of the deep learning models. The described node has sixth-generation Xeon CPU, accepts up to four SATA disk drives, and up to 32 GB of RAM. In addition, the user could add NVIDIA GPU up to 250 W thermal design power (TDP). The device is adapted to use in cars through the specialist construction which is vibration resistant and can work at temperatures between −25°C and 65°C.

Software libraries

Even if we have the most powerful device which can inference a neural network in a very short time, we still need software to use the capabilities of that device. In the last few years, we can observe the process of creating numbers of libraries dedicated to easier usage of ML capabilities. Most of the newest solutions provide us with tools for creating, training, and deploying neural network models based on predefined components. Currently, the new version of these libraries enables us to deploy or even train models on the edge devices. We can distinguish three classes of software libraries:

1. *Fundamental*. This offers functionality dedicated to advanced developers who wish to create a custom neural network and adjust the parameters of each layer manually.
2. *Exemplary*. This enables developers to create a solution based on a set of pretrained models or offers functionality for automatically finding the desired architecture of a neural network based on the provided data.
3. *Universal*. This supports the process of transforming the model between different fundamental and exemplary libraries.

It is noteworthy that the exemplary libraries usually provide the possibility to create a custom network based on the predefined set of fundamental libraries. In this section, we describe a main example of each of the mentioned classes.

Fundamental

TensorFlow Lite from Google. TensorFlow Lite is TensorFlow's lightweight solution for mobile and embedded devices. It enables on-device ML inference with low latency and a small binary size. TensorFlow Lite also supports hardware acceleration with the Android Neural Networks API. TensorFlow Lite uses many techniques for achieving low latency, such as optimizing the kernels for mobile apps, prefused activations, and quantized kernels that allow smaller and faster (fixed-point math) models. TensorFlow Lite supports a set of core operators, both quantized and float, which have been tuned for mobile platforms. They incorporate prefused activations and biases to further enhance performance and quantized accuracy. TensorFlow Lite has a new mobile-optimized interpreter, which has the key goals of keeping apps lean and fast. The interpreter uses static graph ordering and a custom (less dynamic) memory allocator to ensure minimal load, initialization, and execution latency (https://www.tensorflow.org/lite/overview). TensorIO is a portable framework for iOS that removes direct interaction with TensorFlow Lite library (https://medium.com/@_doc_ai/machine-learning-on-the-edge-380d2e90c9c5). In 2019, it is planned to offer the possibility to train models on the edge device (https://www.tensorflow.org/lite/guide/roadmap) and full support for long short-term memory (LSTM) network on the mobile device.

Caffe2Go from Facebook. Caffe2Go is based on the popular Caffe2 framework for developing deep learning models. It stems from Facebook's experience in the usage of ML models on mobile devices (https://heartbeat.fritz.ai/machine-learning-models-on-the-edge-mobile-and-iot-8a5384a370ba). Facebook wishes to create and offer pretrained neural networks which can inference with data stored in the smartphones on the fly. The interesting use cases are as follows:

- Face recognition;
- Fingerprint recognition;
- Detecting persons in the pictures.

Facebook claims that it is a high-performance solution for mobile and edge devices which enables efficient usage to train neural networks on the mobile device. They assume that the training process in the mobile device will be faster than that in the cloud if we include the time needed for the transfer of data from a smartphone to the cloud. Caffe2Go will be used in Facebook's applications to transform videos or photos using style transfer methods. The authors declare that in some smartphones Caffe2Go will be able to provide AI inference at less than 1/20th of a second. Currently this solution was not released yet.

Core ML from Apple. Core ML is Apple's solution for deploying ML models on Apple devices. It lets AI engineers design and develop ML models for Apple iOS apps and then package them into the app bundle. Core ML supports conversion from many of the popular frameworks like TensorFlow and Caffe2.6. Core ML as a domain-specific framework supports Vision for image analysis, Natural Language for natural language processing, and GameplayKit for evaluating decision trees. Core ML as a layer in libraries is on top of low-level primitives like Accelerate and basic neural network subroutines (BNNS), as well as Metal Performance Shaders (https://developer.apple.com/documentation/coreml). Core ML is optimized for devices offered by Apple, which means optimization for memory and power consumption. Ensuring that the privacy of user data is simpler, the data remain on the smartphone, which can be analyzed without a network connection. Core ML is available currently only for devices with the iOS.

Exemplary

ML Kit from Google. TensorFlow Lite is Google's framework with a predefined set of neural network models dedicated for deployment in the edge devices. ML Kit (https://developers.google.com/ml-kit/) offers a few different application program interfaces (APIs) for popular use cases like image recognition and natural language processing, and is integrated with Google's Firebase development platform. It works on both iOS and Android, which is a benefit over Apple's local solution. ML Kit assumes the minimization of cycles required to prepare and adjust AI. The library offers an intuitive developer's toolkit that leverages Google's ML expertise. Base APIs offer ready solutions for many popular use cases in the vision, speech, and text fields. For other use cases, the library accepts also the custom models created in TensorFlow Lite. Then ML Kit acts as an API layer offering an easier method to run and use the models. The created models may be run in the cloud or in the mobile or edge device.

Fritz.ai. Fritz.ai (https://heartbeat.fritz.ai/machine-learning-models-on-the-edge-mobile-and-iot-8a5384a370ba) is the commercial end-to-end platform that offers a set of tools to convert, deploy, and manage the ML models in edge and mobile devices. One model may be created for all types of supported platforms and Fritz.ai will automatically convert and deploy it on appropriate devices. Fritz.ai platform assumes that users do not have to be AI experts and all required support is offered through a set of available functions. It is similar in idea to the ML Kit from Google which means that only the user can send data to analysis, and provides results from predefined, popular models.

Chainer. Chainer[6] is a library which can be used for easily creating, training, and evaluating the deep learning models which will be accepted by the AWS Greengrass platform.[7] The library is based on the CuPy library and through that supports the CUDA GPU and multi-GPU learning process. The user can use the available functionality to quickly implement one of the known convolutional neural networks (CNNs), or recurrent network using also higher order derivatives. Chainer library proposes a define-by-run approach to create a neural network based on the computational graph. This means that Chainer is one of the dynamic graph frameworks in which the graph is allowed to change each iteration. In opposition, there are static graph frameworks like TensorFlow, Caffe2, and Microsoft CNTK which use the same computational graph for every iteration. The benchmarks (https://github.com/soumith/convnet-benchmarks; https://github.com/neulab/dynet-benchmark) show that the Chainer library is faster than PyTorch, TensorFlow, and Keras in some tasks. Chainer library may be extended by modules that offer ready to use models for computer vision, reinforcement learning, and modules for creating distributed networks of devices for using the created deep learning models. Currently, Chainer is offered by the AWS Greengrass platform as ready to use on the devices based on Intel Atom, NVIDIA Jetson TX2, and Raspberry Pi.

Universal

Open Neural Network Exchange (ONNX). Open Neural Network Exchange (ONNX) is an open-source standard for serialization of the AI models and transferring them from one to another fundamental library.

Currently, ONNX offers converters for most major software such as Caffe2, Cognitive Toolkit, MXNet, PyTorch, and MATLAB. Besides the mentioned formats, there are also converters in an early version for Core ML and TensorFlow. ONNX.js is a JavaScript library created by Microsoft dedicated for running ONNX models on browsers and on Node.js. (https://onnx.ai/getting-started). ONNX offers to the user a collection of pretrained models which are widely adopted to different domains and widely presented in the literature. Currently, we do not have the possibility to transform the ONNX models to TensorFlow Lite directly. But we can do that indirectly through TensorFlow and next TensorFlow Lite (https://bit.ly/2Hy8UCp).

IoT platforms

In order to create the fog, we need to use a platform which supports us in synchronization and management of devices that are outside the cloud. The fog computing paradigm proposes reducing the latency through using interlayers between edge devices and the cloud. To efficiently use the advantages of these interlayers, we need tools to orchestrate tasks, run complex scenarios, and distribute them depending on the current context and devices' conditions. Finally, we should have the possibility to describe known scenarios of computation in which we can decide under which conditions the decision-making process will be performed based on the edge devices, interlayers, or the cloud.

The abovementioned problems are addressed to the IoT platforms which provide us well-defined solutions appropriate to the fog paradigm. Basically, we can distinguish the available platforms into two classes:

- Vendor-agnostic platforms support communications in the north of infrastructure with different providers of cloud computing;
- Vendor-specific platforms are created by the company, which support only their services, for example, Google support GCP and Amazon support AWS Cloud.

To the first class, we can assign EdgeX Foundry which was created by the Linux Foundation. The second class represents solutions created by IT companies which also offer cloud environments. They are interested in simplifying the process of usage of the cloud computation power by the processes executed on the edge. Especially, the proposed solutions may be interesting when we wish to train and use DNN models.

EdgeX Foundry

EdgeX Foundry (https://www.edgexfoundry.org/about/) is the project of a platform supporting developers in creating, connecting, and managing edge devices. The project unifies the components that may be used to create solutions on the edge. This unification provides the creation of the marketplace with components OS agnostic and hardware agnostic. The components and devices based on the EdgeX Foundry platform use an open and interoperable API which is based on microservices.

The platform is focused on leverage of cloud-native principles such as loosely coupled services, platform independence, or scalability. The key feature of EdgeX Foundry is platform independence offered on three levels:

- CPU (e.g. x86, ARM);
- OS (e.g. Linux, Windows);
- Application environment (e.g. .NET, Python, Java, Go).

Using the proposed platform, the user may create the production environment (fog or edge computing) based on the available set of example microservices. Data from IoT nodes may be sent to analysis in the cloud or analyzed on the edge.

AWS IoT Greengrass

AWS IoT Greengrass (https://aws.amazon.com/greengrass/) is the Amazon Platform-as-a-Service (PaaS) offered in the Amazon Web Services cloud environment in order to connect edge devices which can perform data locally. AWS Greengrass is divided into three parts:

- AWS IoT Greengrass Core acts as a hub which connects devices that are running on Amazon FreeRTOS or has AWS IoT Device SDK installed. The registered devices create the AWS IoT Greengrass group within which they can communicate.
- AWS IoT Greengrass Connectors allows to connect with third-party solutions (sensors, gateways, on-premises software) without writing their own source code.
- AWS IoT Greengrass Secrets Manager stores and manages credentials, keys, and configurations of devices on the edge. If they need some credentials, we may use Greengrass Connectors for access to the Secrets Manager.

The main goal of this platform is providing services for efficient device management and collecting data in durable storage. Devices registered in the Greengrass platform can run also AWS Lambda functions and use the cloud to compute power-consuming tasks, for example, learning neural networks. AWS Lambda

functions can have access to the data available locally on the edge device such as cameras, serial ports, and GPUs. Moreover, the Greengrass platform provides sync and communicates securely with devices known in the platform including error handling such as a temporarily offline device. AWS Greengrass assumes that the learning process should be performed in the cloud, and the inference process may be performed on the edge devices. Using the platform, we can choose which data will be transmitted to the cloud and which should be processed locally.

Microsoft Azure IoT Edge

Microsoft Azure IoT Edge (https://azure.microsoft.com/en-us/services/iot-edge/) is a service for customers who wish to analyze their raw data as soon as possible without immediately sending them to the cloud. The user may configure which data are transferred to the cloud and which should be calculated on the edge side. IoT Edge is aware of potential communication problems with edge devices, which means that the platform waits until the edge device will be available in order to synchronize data. IoT Edge is interoperable and system agnostic through the support of different OSs in the edge devices such as Windows or Linux and application written in Java, .Net, Node.js, C, or Python. IoT Edge is based on the IoT Hub service and users can register and manage in one place all their sensors and edge devices across many platforms. Management of IoT devices is available through Azure IoT Hub.

IoT Edge support deploys a variety of types of applications such as AI models, computer visions, and real-time streaming of data. Microsoft also offers the service called IoT solution accelerators which are basically templates of IoT scenarios which could be used to quickly create an own IoT solution in an organization.

Moreover, the users of Microsoft Azure IoT Edge can use AI Toolkit (https://github.com/Azure/ai-toolkit-iot-edge) which offers many complex models ready to use, for example, to perform image recognition. Using the Cognitive Service platform and Computer Vision library (https://www.customvision.ai), the user also has the possibility to easily train a model concerning computer vision problems. The created model may be exported to platforms Core ML, ONNX, or TensorFlow using the protobuf format. The model may also be optimized and converted to the embedded device using the Microsoft Embedded Learning Library. Users may use Azure Machine Learning to train a model, manage it, and containerize it and using Azure IoT Edge to describe the data pipeline and the process of how the model should be deployed in the edge.

Google Cloud IoT Edge

Google Cloud IoT Edge (https://cloud.google.com/blog/products/gcp/bringing-intelligence-edge-cloud-iot) is a software that allows you to utilize cloud computation power and AI capabilities in the edge devices. Cloud IoT Edge is installed on the edge device and includes two components: Edge Connect and Edge ML. The first is dedicated to connect the device to the cloud, improve security of communication, and support the user in software and firmware updates. Moreover, through Edge Connect, the user can manage data exchange between edge devices and the cloud. Through Cloud IoT Edge, it is possible to use other Google Cloud services:

- Cloud Dataflow for streaming and batch analysis;
- Cloud Pub/Sub for ingesting event streams and delivering them to the Dataflow component;
- Cloud IoT Care for device connection and management;
- BigQuery for data warehouse and fast querying;
- Cloud ML Engine to train, deploy, and run ML models.

In the edge device, we could also use the Edge ML component to run pretrained ML models saved in TensorFlow Lite format. Edge ML also enables the usage of DL accelerator Google Edge TPU which supports efficiency of inference of the TensorFlow Lite models.

Cisco Edge Fog Fabric

Cisco Edge Fog Fabric (CEFF; https://www.cisco.com/c/en/us/products/collateral/cloud-systems-management/edge-fog-fabric/datasheet-c78-738866.html) is an IoT platform for industrial customers which need support in IoT applications for real-time monitoring and diagnostics equipment. The platform has modular microservice architecture that allows immediate and intelligent processing of the raw data in the place where it is registered, or its distribution to the fog or the cloud. The CEFF platform consists of the following key components:

- System administrator;
- Dataflow editor;
- System monitoring;
- Message broker;
- Links;
- IoT historian database.

Currently, Cisco renamed its product Kinetic IoT Platform.

Baidu OpenEdge

The Baidu Company offers an open-source edge computing platform (https://www.infoq.com/news/2019/01/baidu-edge-platform-opensource) which allows to create applications for smart home, wearable, and IoT devices. The OpenEdge is a component that can be installed in the edge devices in order to support collecting data, distributing messages, performing AI inference, and synchronizing devices with Baidu ABC (AI + Big Data + Cloud Computing). Edge devices with OpenEdge could be managed in one place through a Baidu Cloud user interface and run models created in TensorFlow or PaddlePaddle libraries.

Applications

Indoor localization

The problem of indoor localization is important for intelligent applications that select service which depends on the user context. We can find examples of those applications in smart homes, hospitals, and malls. Moreover, we may wish to use the indoor localization techniques to efficiently tracking children in a kindergarten to increase their safety.

Wang et al.[8] describe system DeepFi which uses signals from different transmitters (i.e. Wi-Fi access points, iBeacon, Bluetooth, RFID) and DL models to predict the location. The models are created in offline phase based on the predefined set of data. Previous studies[9],[10] report on the usage of DL models in combination with other learning methods to extract hidden features and estimate positions. The authors also describe the relation between the number of hidden layers in the DL model and their impact on localization accuracy. Liu et al.[11] describe how a CNN is used to predict the indoor localization based on both magnetic and visual sensing data. Additional work was presented in Mohammadi et al.[12] where the authors incorporate the CNN model to analyze an image from the user environment.

Intelligent transportation systems

Intelligent transportation systems (ITS) provide services to automatically collect data and manage the traffic and roads in some region. The example ITS was presented by Ma et al.[13] in which they use a deep restricted Boltzmann machine and RNN to create a model for prediction of traffic congestion. As the input data, they use the Global Positioning System (GPS) signals from taxi riding in the city. Using the GPU, they achieve the accuracy as high as 88% within less than 6 min.

Tian and Pan[14] propose to use LSTM RNNs to predict the short-term traffic flow. The main problem described by the authors is a predefined and static length of input historical data. They propose to use the LSTM RNN model in order to automatically determine the optimal time lags. The conducted experiments confirm the high accuracy of the proposed method for each of the tested intervals (15, 30, 45, and 60 min). The authors achieve the lowest value of mean absolute percentage error (MAPE) metric for each tested interval.

Automated vehicles

The automated vehicles[15] are also known as self-driving cars. Automation and electricity of vehicles are currently described as the main directions of automotive development. Through Zero Vision,[16] the European Union assumes that by 2050 we will totally eliminate accidents on the roads. To achieve that, we need systems which provide the possibility to quickly gather new data from sensors, cameras, and others, and process them in real time to handle all necessary decisions on the road. What is obvious is that we should find solutions that could work without an Internet connection because in many cases we can drive too fast or we can be too far from any city (poor Internet connection) but we will still need an assistant working in real time. This is an example of environments in which we need deep learning on the edge devices and why the popularity of such solutions will be growing. Autonomous vehicles enable us new types of solutions, such as automatic minimization of traffic jams in the city[17] and on-demand transport[18] with sharing car. We can also think about eliminating cars from the center of the city through automatically parking them outside the center and invoke them if they will be needed.

The solutions offered by the NVIDIA Company are highly interesting in supporting the vehicles in autonomous functionality. They offer Jetson family devices and have great achievements in creating neural networks which can be applied in autonomous vehicles.[19] For example, they prepared the CNN network which could be used to steer an automobile.

The Baidu Company offers, for the car manufacturers, solutions ready to use through devices such as Baidu Intelligent Edge AI Board and AI Box with the software Apollo 3.5 (http://apollo.auto/). This is a set of tools which can be used to build an autonomous car. In the project Apollo, more than 30 partners from the automotive industry such as Honda, Microsoft, Bosch, Ford, Jaguar, Hyundai, PSA, Volvo, and others like Intel and ZTE are involved.

Besides many deep learning models (services), the autonomous vehicles need support into the method of virtualization and orchestration of this type of services in edge devices. Morabito et al.[20] propose lightweight virtualization (LV) technologies to address the problems of managing services in the edge devices.

Sallab et al.[21] propose a deep reinforcement learning framework for autonomous driving. First, they use CNN to recognize objects in the environment, such as lane and vehicles. Next, for the prediction of the future state, they used LSTM RNN. Based on the information on recognized objects and potential future states, the reinforcement learning network (Q-network) was used to find the most promising solution.

AllGoVision

Video data can be costly and complex to gather and then transfer and analyze all data in the cloud, that is why Intel invented AllGoVision which provides an innovative analytics solution for video data. It rapidly analyzes video from surveillance cameras for designated parameters, like specified factors, behavior patterns, environmental monitoring, and motion tracking of people and objects (https://www.allgovision.com/).

It offers a deep learning–based video analytics solution that is integrated with existing infrastructure to provide automatic near-real-time alerts and insight for actionable business intelligence. It brings IoT and analytics expertise to unlock valuable insight from video data.[22]

AllGoVision uses an open platform video management software (VMS) and camera integration which is compatible with existing security systems and highly accurate in tasks such as object tracking in crowds. AllGoVision is also customizable and cost effective by reducing hardware needed to provide the same functionality per server. With the AllGoVision and Intel solution, analytics can be conducted at

- The edge and cloud;
- Single server;
- A distributed architecture.

For server analytics, the solution can reside in the VMS or on a separate machine. Open network video interface forum (ONVIF) streaming is supported, along with leading VMS. Alarms and alerts are sent to the VMS viewer (smart client) or to AllGoVision's Alarm Center.[22]

As it comes to the edge and cloud, analytics at the edge and cloud are available on IP cameras. Alarms and alerts are sent to the VMS viewer. Features such as detection of intrusion or suspicious incidents and counting are supported simultaneously. Also, it can be run locally (at the edge or machine) or on the cloud, with the ability to send alarms and alerts via wireless area network (WAN). Alarms and alerts can be hosted on the cloud.

The solution powered by robust Intel architecture can be used by a wide spectrum of industries and vertical segments. It has already been deployed in more than 100 installations in 35 countries.

Agriculture

To effectively planning the agriculture production process, we should have services that help us solve the problems such as detecting disease in plants, predicting the amount and proportion of fertilizers, or automatically managing the irrigation system. The volume of data needed in the process of preparing the decision may extend the capability of usually poor Internet connection. In this case, we should be able to run the decision-making process on the edge devices.

Sladojevic et al.[23] reported how deep CNNs could be used to detect, recognize, and classify diseases based on leaf images. In the classification process, they define 13 categories and achieve an accuracy of about 96%. The application with the prepared model may be provided by a regional server or by a smartphone with ML co-processor. In the long-term observation, we could also perform data fusion in order to find dependency between weather condition, used pesticides, and recognized disease.

Support of a large-scale agriculture requires a broader set of data, for example, from satellite. Kussul et al.[24] propose using data from the Landsat-8 and Sentinel-1A satellites and analyze them with the CNNs. In the result, they achieve 85% accuracy for all major crops such as wheat, maize, sunflower, soybeans, and sugar beet. Similar works in which DL was used for land and crop detection were described in previous papers.[25,26] The proposed solutions may help in creating the automated monitoring and management services in large-scale agriculture lands.

Bargoti and Underwood[27] propose to use fast R-CNN to create bounding boxes around the fruit and detect fruits. They create custom solutions based on ZF and VGG16 networks with five convolutional layers and a VGG16 network, with 13 convolutional layers. They analyzed F1 metric for three types of fruits: Apples (0.904), Mango (0.908), and Almond (0.775). They processed images with a resolution of 500×500 pixels for which the time of prediction was about 0.1 s using the GPU NVIDIA 980 Ti.

Patient-generated health data

Currently, in hospitals there are thousands of documents being generated containing the important data but usually those documents are only stored in databases. Medics do not have time to analyze all data about the patient but use only the last few records. Data can also be generated outside the hospital by IoT devices such as wearable sensors, smartphones, pacemakers, and glucose meters. Users may wish to have access to the actual recommendations, predictions, or alerts that are generated from the collected medical data.

Edge computing applications have the potential to solve this data problem. Using them, we can analyze information in real time and filter the collected data before they will be sent over the Internet.

The new Apple Watch 4 (https://support.apple.com/en-us/HT208944) can analyze a person's current condition and send alerts the moment anomalies are detected, allowing for rapid response times that may well save their life. In the meantime, the device can continue to feed non-critical data gathered over time to be sorted and processed by the network's more powerful central servers or data centers operated by edge computing companies.

Muhammad et al.[28] propose a system for voice pathology detection based on signals registered by IoT devices. All signals are sent to the cloud for analytics using an extreme learning machine trained by voice signals to diagnose the pathology.

Heart diseases: medical imaging

The popularity of the personal assistants embedded in the edge devices will be increased with creating new useful applications. For example, Pereira et al.[29] proposed to use a smart pen with sensors to measure handwritten dynamics in order to help in identifying Parkinson's disease in its early stages.

A patient with heart disease requires continuous monitoring in order to quickly predict the heart arrhythmias from an electrocardiogram (ECG) in an outstanding way. This application could be installed in smartphone, smartwatch, or dedicated device and use methods proposed by the Stanford team (http://safebeat.org/newsroom/headlines/the_machines_are_getting_ready_to_play_doctor/). They proposed a deep learning algorithm to identify different types of irregular heartbeats based on ECG data.

In addition, there are a lot of applications that work along with ECG or electroencephalogram (EEG) devices and turn alarms on when something worrying happens. Also, we can watch how websites like www.arterys.com grow instantly. A web platform for medical imaging to transform clinical care diagnostic certainty allows to collaborate with other physicians from URL and to deliver powerfully simple AI-enabled imaging solutions (like LungAI, CardioAI, LiverAI, BreastAI). The application and its owners won many meaningful awards such as MedTech Innovator 2017, SIIM Innovation challenge 2017, or FABA 2017.

Decision support system in medicine

Currently, the most medical decisions are prepared based on some subset of data and experience of a doctor. Simple applications used to manage hospital processes like electronic patient record (EPR) do not provide intelligent services to increase the efficiency of decision-making processes. Results reported in Giacomini et al.[30] point out that the combination of clinical and algorithm diagnosis can contribute to the decrease of doubtful and incorrectly classified patients and add weight to the work of the physician in the majority of cases. Although the system is in early phase, a prototype framework has already been tested in a pre-release version at the Esophageal Surgical Unit of the School of Medicine of the University of Genova to find possible qualitative correlations between gastroesophageal reflux disease (GERD) symptoms and functional results from esophageal manometry and 24-h esophageal gastric pH monitoring.

Genome

Genomes are a natural fit for AI[31] because they often consist of huge datasets, which require a lot of annotation and work. Existing software requires well-annotated genomes, and because of this much of our understanding centers around more popular organisms. Using deep learning, a software called DeepVariant can identify single-nucleotide polymorphisms by transforming genomic information into images.[32]

These images represent the different nucleotides and can be used by the model to identify polymorphisms. This method makes it possible for researchers to use it because more uncommon model organisms can be studied. This removes a significant amount of work by researchers on improving low-quality reference genomes, and the high error rate otherwise associated.

DeepVariant is an updated, open-source (GitHub) deep learning–based variant caller. It applies the Inception TensorFlow framework, which was originally developed to perform image classification. DeepVariant converts a BAM file into images similar to genome browser snapshots and then classifies the positions as variant or non-variant. Conceptually, it uses the idea that if a person can leverage a genome browser to determine if a call is real, a sufficiently smart framework should be able to make the same determination.[32]

Improving authentication methods

Secure authentication of a signal is probably one of the most challenging problems in IoT. It is related to the large-scale nature of the system and vulnerability to man-in-the-middle and eavesdropping attacks. To prevent such situations, new solutions based on deep learning and IoT are created.

The framework proposed by Ferdowsi and Saad[33] integrates a deep learning and game theory to enable computationally efficient authentication of IoT signals and devices. The deep learning algorithm used in this framework uses LSTM blocks to extract stochastic

features from the IoT signal and watermarks them inside the original signal. This allows operators in the cloud to detect attacks because the attacker cannot change watermarks. In addition, the LSTM algorithm compared to other methods (such as encryption) reduces the complexity and latency of the attack. This enables DNN to effectively complement the basic cryptographic and security solutions of an IoT.

Malware detection schemes

It is possible to use supervised learning techniques to evaluate the runtime behaviors of the apps in malware detection by IoT devices. In the malware detection scheme, an IoT device uses K-NNs and random forest classifiers to build the malware detection model.

Narudin et al.[34] report that the IoT device can filter the Transmission Control Protocol (TCP) packets and select the features among various network features including the frame number and length, label them, and store these features in the database. To perform these tasks, the algorithm uses the K-NN-based malware detection to assign the network traffic to the class with the largest number of objects among its K-NNs. The second classifier based on random forest builds the decision trees with the labeled network traffic to distinguish types of malware. The achieved results indicate that true-positive rates (TPRs) of the proposed K-NN-based malware detection and random forest-based scheme with the MalGenome data set are 99.7% and 99.9%, respectively.

Voice assistants

Voice assistants (Alexa, Cortana, Siri, Google)[35,36] currently have to send data from the user device to the cloud and wait for analysis. This may be a potential drawback if the user values his privacy. The user has no warranty which information is sent to the cloud and where exactly is the cloud. If the assistants also have access to the data on health, then in many countries sending this type of data outside the country is prohibited. ML on the edge is one solution to prevent sending sensitive data outside the user location.

Amazon Go

On 22 January 2018, Amazon opened its first convenience store, Amazon Go in Seattle, Washington. The store attracted a lot of attention because it does not have any hurdles which people usually face in traditional supermarkets. The store does not have any registers, cashiers, or checkout line. Amazon has called this "Just Walk Out" technology. Amazon's Just Walk Out technology uses ML, IoT, computer vision, and sensor fusion which ensure that customers are only charged for the product which they actually bought. Amazon has also made it clear that it does not use facial recognition technology.[37]

Before visiting Amazon Go, the shoppers must have the Amazon Go app linked to their Amazon Prime accounts. The key feature generates a QR code that lets a user enter the store. And the Receipts feature tells you about the things you have bought as soon as you walk out of the store. The payment is processed using a credit card which is linked to your Amazon account. There are a lot of cameras in the store. These cameras are the central equipment of how things function in the store. The cameras use computer vision, which tells the system when the item is taken from the shelf and who has taken it. Amazon has also revealed the use of "sensor fusion" which combines data from different sensors for perfection and efficiency. The various sensors include the weight sensors which allow the company to track individual product.

AWS DeepLens

AWS DeepLens[38] is the first camera with deep learning functionality. DL algorithms can use 100 GFLOPS of computing power on the device. This allows to process HD video in real time using deep learning models. The camera cooperates with Amazon SageMaker which supports the user in creating, training, and deploying the ML models. AWS DeepLens accepts popular deep learning frameworks such as TensorFlow and Caffe. The device is based on the Intel Atom Processor, 8 GB RAM, Intel Gen9 Graphics Engine, and Ubuntu as an OS.

Intelligent Suitcase

ForwardX Company was created in May 2016 and has started R&D project with an intelligent Suitcase (Ovis; https://www.forwardx.com). Now their product was presented in CES 2019 as a suitcase which learns face and posture of the owner, in order to automatically follow him. The suitcase uses the AI models to recognition patterns without sending data to the cloud.

Anomaly detection

In some cases, we may wish to quickly detect anomaly in order to reduce the range of security attacks or to increase efficiency of the industry processes.

Vincent et al.[39] describe the solution based on an autoencoder for anomaly detection which is created as a two-part algorithm. The first part resides in the edge device and the second one in the cloud. The edge device can detect anomaly in a fully distributed manner, without communicating with the cloud. The second part of the algorithm resides in the cloud and is dedicated to

the computation-intensive task such as inference of the model.

IBM GRAF (Global High-Resolution Atmospheric Forecasting System; https://www.ibm.com/weather/industries/cross-industry/graf) is a forecasting system that uses mobile devices (e.g. smartphones) as a set of distributed sensors which increase the density of the measurements and improves the accuracy of prediction. They are particularly valuable in places where there is a lack of sensors and the predictions have poor quality.

Conclusion

A revolution in Edge AI computing has started. In this article, we distinguished three main challenges in this revolution:

- Supply edge devices with enough computing power to run AI algorithms like DNNs;
- Create libraries which provide the transfer of pretrained models to the edge devices and use them efficiently;
- Provide a platform to ease managing, updating, and syncing the set of edge devices in a system-agnostic way.

Currently, in the market we can see two opposite directions of development of the ML on the edge which we will call: fat and thin edge devices.

The first one assumes that the edge devices should be autonomous and all potential predictions or decisions should be prepared on the device side with the reduction of time when a connection with the cloud is needed. This is a direction good for autonomous vehicles, intelligent wearable devices, and healthcare, but also for industrial companies for which the privacy of production data is crucial. The second direction is thin edge devices, which assumes tightly coupled cloud architecture, which means that the cloud will still provide the main role in processing new data.

Distinctions proposed in this article are examples of totally opposite directions. Rather than dividing the market, we should expect the development of hybrid solutions dedicated for fog computing and further for edge computing. The process of creation of the smart city architecture, increasing the number of intelligent vehicles, clothes, and other objects, may cause an increasing number of small data centers located close to the data sources.

Intel, Google, NVIDIA, Apple, and Samsung lead in hardware development and try to create rather the fat edge devices dedicated to fast analysis of video streams. On the other side, Facebook and Google create great software for high-performance deep learning inference and training. Currently, there are great tools for fast development and prototyping like TensorFlow and PyTorch, but deployment capabilities are limited. For instance, developers report many problems with running TensorFlow Lite on Raspberry Pi, but also other devices. At the same time, we are waiting for a solution from Facebook, their Caffe2Go. Finally, managing edge infrastructure is not easy, but platforms like EdgeX, AWS IoT Greengrass, or Google Cloud Edge help with that. Currently, many of them seem to support only thin edge devices by requiring the data to be sent from the edge device to the cloud in order to analyze and store them.

We should still observe changes in the market because many solutions (software and hardware) are still in beta or early development stage. We need to wait for the maturity of a whole edge computing ecosystem which will enable the deployment of AI models in business use cases such as Industry 4.0 and indoor localization, through ITS, finally to agriculture and healthcare applications.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work has been partially supported by funds from the Faculty of Electronics, Telecommunications and Informatics of Gdańsk University of Technology.

ORCID iD

Julian Szymański https://orcid.org/0000-0001-5029-6768

References

1. Atzori L, Iera A and Morabito G. The internet of things: a survey. *Comput Netw* 2010; 54(15): 2787–2805.
2. Hummen R, Henze M, Catrein D, et al. A cloud design for user-controlled storage and processing of sensor data. In: *Proceedings of the 4th IEEE international conference on cloud computing technology and science*, Taipei, 3–6 December 2012, pp.232–240. New York: IEEE.
3. Moullec ML, Jankovic M and Eckert C. Selecting system architecture: what a single industrial experiment can tell us about the traps to avoid when choosing selection criteria. *AI Edam* 2016; 30(3): 250–262.
4. Qi H, Brown M and Lowe DG. Low-shot learning with imprinted weights. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, Salt Lake City, UT, 18–23 June 2018, pp.5822–5830. New York: IEEE.
5. Kurková V, Manolopoulos Y, Hammer B, et al. Artificial neural networks and machine learning—ICANN

2018. In: *Proceedings of 27th international conference on artificial neural networks (part II)*, vol. 11141, Rhodes, Greece, October 4–7, 2018. Cham: Springer, 2018.
6. Tokui S, Oono K, Hido S, et al. Chainer: a next-generation open source framework for deep learning. In: *Proceedings of workshop on machine learning systems (LearningSys) in the twenty-ninth annual conference on neural information processing systems (NIPS)*, vol. 5, 2015, pp.1–6, http://learningsys.org/papers/LearningSys_2015_paper_33.pdf
7. Xu X, Huang S, Feagan L, et al. EAaaS: edge analytics as a service. In: *Proceedings of the 2017 IEEE international conference on web services (ICWS)*, Honolulu, HI, 25–30 June 2017, pp.349–356. New York: IEEE.
8. Wang X, Gao L, Mao S, et al. DeepFi: deep learning for indoor fingerprinting using channel state information. In: *Proceedings of the 2015 IEEE wireless communications and networking conference (WCNC)*, New Orleans, LA, 9–12 March 2015, pp.1666–1671. New York: IEEE.
9. Gu Y, Chen Y, Liu J, et al. Semi-supervised deep extreme learning machine for Wi-Fi based localization. *Neurocomputing* 2015; 166: 282–293.
10. Zhang W, Liu K, Zhang W, et al. Deep neural networks for wireless localization in indoor and outdoor environments. *Neurocomputing* 2016; 194: 279–287.
11. Liu Z, Zhang L, Liu Q, et al. Fusion of magnetic and visual sensors for indoor localization: infrastructure-free and more effective. *IEEE T Multimedia* 2017; 19(4): 874–888.
12. Mohammadi M, Al-Fuqaha A, Sorour S, et al. Deep learning for IoT big data and streaming analytics: a survey. *IEEE Commun Surv Tut* 2018; 20: 2923–2960.
13. Ma X, Yu H, Wang Y, et al. Large-scale transportation network congestion evolution prediction using deep learning theory. *PLoS ONE* 2015; 10(3): e0119044.
14. Tian Y and Pan L. Predicting short-term traffic flow by long short-term memory recurrent neural network. In: *Proceedings of the 2015 IEEE international conference on smart city/SocialCom/SustainCom (SmartCity)*, Chengdu, China, 19–21 December 2015, pp.153–158. New York: IEEE.
15. Millard-Ball A. Pedestrians, autonomous vehicles, and cities. *J Plan Educ Res* 2018; 38(1): 6–12.
16. Johansson R. Vision zero—implementing a policy for traffic safety. *Safety Sci* 2009; 47(6): 826–831.
17. Stern RE, Cui S, Delle Monache ML, et al. Dissipation of stop-and-go waves via control of autonomous vehicles: field experiments. *Transport Res C: Emer* 2018; 89: 205–221.
18. Fagnant DJ and Kockelman KM. Dynamic ride-sharing and fleet sizing for a system of shared autonomous vehicles in Austin, Texas. *Transportation* 2018; 45(1): 143–158.
19. Bojarski M, Del Testa D, Dworakowski D, et al. End to end learning for self-driving cars. *Arxiv*, 2016, https://images.nvidia.com/content/tegra/automotive/images/2016/solutions/pdf/end-to-end-dl-using-px.pdf
20. Morabito R, Cozzolino V, Ding AY, et al. Consolidate IoT edge computing with lightweight virtualization. *IEEE Netw* 2018; 32(1): 102–111.
21. Sallab AE, Abdou M, Perot E, et al. Deep reinforcement learning framework for autonomous driving. *Electron Imag* 2017; 2017(19): 70–76.
22. Tripathi RK, Jalal AS and Agrawal SC. Suspicious human activity recognition: a review. *Artif Intell Rev* 2018; 50: 283–339.
23. Sladojevic S, Arsenovic M, Anderla A, et al. Deep neural networks based recognition of plant diseases by leaf image classification. *Comput Intel Neurosc* 2016; 2016: 3289801.
24. Kussul N, Lavreniuk M, Skakun S, et al. Deep learning classification of land cover and crop types using remote sensing data. *IEEE Geosci Remote S* 2017; 14(5): 778–782.
25. Kuwata K and Shibasaki R. Estimating crop yields with deep learning and remotely sensed data. In: *Proceedings of the 2015 IEEE international geoscience and remote sensing symposium (IGARSS)*, Milan, 26–31 July 2015, pp.858–861. New York: IEEE.
26. Scott GJ, England MR, Starms WA, et al. Training deep convolutional neural networks for land-cover classification of high-resolution imagery. *IEEE Geosci Remote S* 2017; 14(4): 549–553.
27. Bargoti S and Underwood J. Deep fruit detection in orchards. In: *Proceedings of the 2017 IEEE international conference on robotics and automation (ICRA)*, Singapore, 29 May–3 June 2017. New York: IEEE.
28. Muhammad G, Rahman SMM, Alelaiwi A, et al. Smart health solution integrating IoT and cloud: a case study of voice pathology monitoring. *IEEE Commun Mag* 2017; 55(1): 69–73.
29. Pereira CR, Pereira DR, Papa JP, et al. Convolutional neural networks applied for Parkinson's disease identification. In: Holzinger A (ed.) *Machine learning for health informatics*. Cham: Springer, 2016, pp.377–390.
30. Giacomini M, Michelini R, Deantoni F, et al. A visual tool for a user-friendly artificial neural network based decision support system in medicine. In: *Proceedings of the conference on artificial intelligence in medicine in Europe: artificial intelligence medicine*, Cascais, 1–4 July 2001, pp.148–151. Berlin; Heidelberg: Springer.
31. Alipanahi B, Delong A, Weirauch MT, et al. Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. *Nat Biotechnol* 2015; 33(8): 831–838.
32. Zou J, Huss M, Abid A, et al. A primer on deep learning in genomics. *Nat Genet* 2019; 51: 12–18.
33. Ferdowsi A and Saad W. Deep learning for signal authentication and security in massive internet of things systems (arXiv preprint arXiv:1803009162018), https://arxiv.org/pdf/1803.00916.pdf
34. Narudin FA, Feizollah A, Anuar NB, et al. Evaluation of machine learning classifiers for mobile malware detection. *Soft Comput* 2016; 20(1): 343–357.
35. Hoy MB. Alexa, Siri, Cortana, and more: an introduction to voice assistants. *Med Ref Serv Q* 2018; 37(1): 81–88.
36. López G, Quesada L and Guerrero LA. Alexa vs. Siri vs. Cortana vs. Google assistant: a comparison of speech-based natural user interfaces. In: Nunes I (ed.) *Advances in human factors and systems interaction (international*

conference on applied human factors and ergonomics, 2017). Cham: Springer, 2017, pp.241–250.
37. Polacco A and Backes K. The Amazon Go concept: implications, applications, and sustainability. *J Bus Manag* 2018; 24(1): 79–92.
38. Amazon. AWS DeepLens, https://aws.amazon.com/deeplens/ (2018, accessed 19 January 2019).
39. Vincent P, Larochelle H, Lajoie I, et al. Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. *J Mach Learn Res* 2010; 11: 3371–3408.