Deep Learning on the mobile edge

Georg Eickelpasch, Marton Kajo*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email:georg.eickelpasch@tum.de, kajo@net.in.tum.de

***Abstract*—Applying Deep Learning on mobile devices proves to be a difficult challenge due to limited resources. However, it is still very much a required methodology, especially in the context of IoT devices, and therefore the computation is offloaded to the cloud or mobile edge. The offloading can be done dynamically were the device, edge and cloud share the work depending on different factors.**

***Index Terms*—Deep Learning, mobile edge computing, Internet of Things**

1. Introduction

This paper tries to give a comprehensive look at the questions of what applications benefit from using Deep Learning in a mobile context and how it can be applied. Because the data cannot be processed on the device, it has to be offloaded. This paper will take a look at different offloading strategies and when to use them.

Currently, Deep Learning is already used a lot on mobile devices, for example for speech recognition and image detection. However, a common approach is to offload the entire workload to the cloud. This brings many disadvantages, e.g. a strong dependency on a good bandwidth which can be a bottleneck. There are promising approaches, such as shown by Yiping Kang et al. in [1] to use the mobile edge as well as the cloud or completely replace the cloud by the edge like shown by En Li et al. in [2]. These algorithms offer a great improvement over cloud offloading but are still improvable. In this paper, we will show the strength and weaknesses of the algorithms as well as their common use cases.

Edge Computing is the idea to use locally existing hardware in the network instead of the cloud which is thought of a far away, more or less unlimited computing power. When talking about the edge, there are two different ways of thinking of the edge.

1) The first one is to think of the edge as the end-user mobile device with limited computing power and therefore tries to offload as much as possible to the cloud.
2) The second idea is that the edge is a powerful local server that offers the advantage of local caching and a low round trip time (RTT) for communication.

Even though, the ideas seem very different they can be used similarly. In both cases, there is one device with weak computation power and one powerful device, which can be reached through offloading. Scheduling algorithms that offload between one computational strong and one computational weak component can be used in both cases, however, one has to be aware whether the edge is the strong component or the weak component.

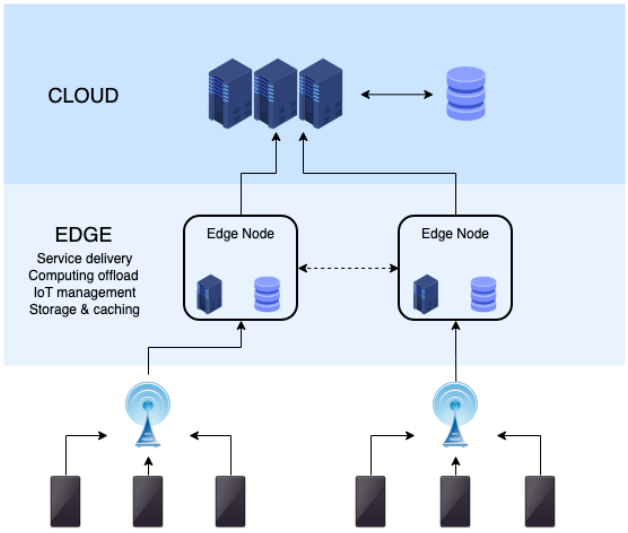


Figure 1: Different layers of Edge Computing Model [3]

2. Background

The offloading strategies discussed in this paper are based on the NeuroSurgeon paper [1] as well as the Edgent paper [2]. But while these two works focus on the details of the individual algorithm this work focuses on the high-level advantages and disadvantages of each. Furthermore, this paper shows the problems of Deep Learning on the mobile edge using the specific case of an IoT system as shown by Hakima Khelif et al. in [4] and He Li et al. in [5].
The usage of Deep Neural Networks (DNNs) is rapidly increasing in everyday life. Nowadays not only powerful computers but even mobile devices, e.g. smartphones use DNNs for various applications. Until recently, running a DNN on a mobile device seemed unfeasible. This is because mobile devices usually face the problem of limited resources. Restricted processing power and memory, as well as limited energy availability, meet the high computational effort of a DNN. But since the benefits and use cases of DNNs are more and more desirable the

doi: 10.2313/NET-2020-04-1_01

scientific community is working on ways to execute DNNs on mobile devices. The easiest approach to offload the actual computing into the cloud, which works well but is bottlenecked by the data transfer between the mobile device and the cloud and therefore require high bandwidth for good performances. To increase this performance a hybrid model between the cloud and the mobile device with strategically and dynamically chosen offloading points is a viable attempt. But since the processing power on the device is highly limited it still leaves much room for improvement. The current state-of-the-art solution is another processing layer, between the mobile device and the cloud, called the Mobile Edge. Instead of offloading all heavy computational tasks to the cloud they are offloaded to strong processing devices nearby. This does not only improve the performance drastically and decreases the dependency on a strong internet connection, but also improves security, since personal data is not required to be offloaded to the cloud.

3. Use cases

In this section, we want to show which mobile applications require DNNs that run on the device or the edge.

3.1. Speech

The first thing that comes to mind is probably speech recognition. That means converting language spoken by a human into a machine-readable format. This problem used to be incredibly hard to solve and if solved rudimentary, is only able to detect very clearly spoken words or limited vocabulary which is not a sufficient solution for everyday applications like simple dictating tools for speech to text, since manually correcting errors can be very tedious. DNNs are suited well for speech recognition because they are more robust than alternatives and perform comparably well with noisy data. In combination with speech recognition, a second major field is often used because converting the speech to text is not always enough the machine also needs to understand what the user is speaking. For this Natural Language Processing (NLP) is required. Due to the complexity of natural languages, DNNs are a viable solution approach for NLP. However, NLP applications like intelligent personal assistants, e.g. Amazon Echo or Google Home require real-time processing since it is supposed to feel natural, like speaking to a human.

3.2. Computer Vision

Another problem commonly faced by mobile devices that requires DNNs is object identification on pictures. This could be a smart fridge checking what groceries need to be bought or a phone identifying the user via FaceID to unlock a smartphone. While the smartphone again faces the challenge of real-time results it has the new problem of a highly mobile context which leads to inconsistent bandwidth in some cases, so offloading to the cloud is unreliable. The smart fridge might not require real-time results but even though it is expected to be connected to a wifi network, bandwidth is a limited resource. This is due to the expected growth of the IoT and connected home devices. If all of these devices use bandwidth recklessly, it will lead to congestion in the network. Pairing Edge Computing with IoT devices can greatly reduce required bandwidth and therefore is a desirable methodology. A more advanced field in computer vision that requires DNNs is Continuous Vision. Tracking objects in a video in real-time is very useful on the one hand, but on the other hand, it is very challenging to do. Currently mostly used in robotics and surveillance it is expected to expand more into the private sector for example for video games. Being one of the computationally most expensive subtopics due to the sheer size of video files offloading everything to the cloud is not possible for real-time applications. Therefore preprocessing on the edge is currently the most promising approach.

3.3. Bioinformatics

Lastly, bioinformatics and especially E-Health is a fast-growing sector that requires DNNs due to noisy real-world data. To get good results with noisy data DNNs are a common approach. Also, E-Health requires very robust solutions since human lives might depend on it. Therefore, DNNs are the first choice. In the context of IoT devices offloading to the edge can also provide the advantage of anonymizing the data which is also especially important for human health data.

4. Deep Learning on the mobile Edge in an IoT context

In the previous section, many use cases for DNNs on the mobile edge were shown. Many of them were related to or applicable in IoT devices. In this section, we want to take an in-depth look at the problems IoT devices face and why DNNs are a good way to solve them. After that, we will add in Edge Computing as a solution to apply DNNs to IoT devices and see what other benefits arise.

4.1. DNNs in IoT

The IoT is a growing phenomenon where more and more devices are interconnected with each other and the Internet. Not only smart home solutions for private people but more importantly in factories are more and more robots and machines part of the IoT and it is expected to change, or already changing the way how pipelines work, in Germany under the synonym Industry 4.0 [6]. All of these devices use various sensors and communication to adapt in real-time to changing conditions. However, real-world data are always noisy or hard to predict. To work with this data DNNs are a powerful solution due to their robustness and the availability of a large amount of data. Of course not every IoT device has powerful computing capabilities so the actual DNN has to be computed externally.

4.2. Edge Computing for DNNs in IoT

Offloading the computation process traditionally means to upload the raw data into the cloud where everything will be fully computed and the result returned

doi: 10.2313/NET-2020-04-1_01

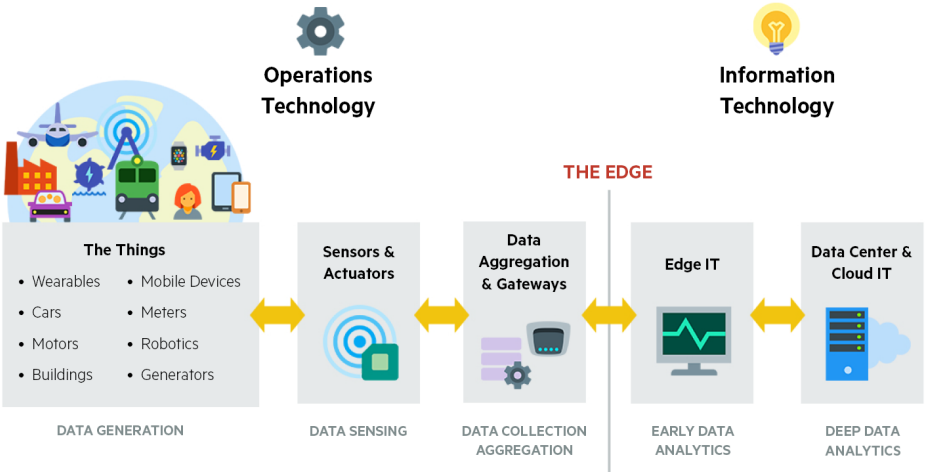


Figure 2: Workflow of an IoT-Edge solution [7]

to the IoT device. However, with more and more devices being part of the IoT uploading all the raw data themselves can already be too expensive or not be achieved within a required timeframe. Upgrading bandwidth and computational infrastructure are possible, but it also has limitations and can be very expensive, therefore a method to relieve the bottleneck which is uploading is very useful. This method is Edge Computing. Edge Computing can be done on the IoT device itself or a designated physically close edge server. The basic idea is to exploit the property of many DNNs that the input data are bigger than the intermediate data [2]. This means that the relevant data are smaller than the raw data, after processing the first layers. So the beginning of the processing is done on the edge device even though it computes slower than the cloud but the lost time is saved later when smaller data are uploaded to the cloud instead. By doing this the bottleneck is relieved of some workload. The question remains when exactly to offload to the cloud service. We will take a look at this question in the next section. But there are more benefits in using Edge Computing than just compressing the data by preprocessing them. Another typical feature of data acquired by IoT devices is the very high heterogeneity due to many different sensors and different use cases or environments. If everything is uploaded to the cloud this heterogeneity requires the cloud to be able to process many kinds of data. If Edge Computing is used the edge device can preprocess the data and since each edge device usually faces the same data from the same sensors it can be optimized to specifically preprocess this data and upload a more generalized dataset to the cloud. This localization of data preprocessing to the edge also enables strong caching mechanisms since the data of each sensor are processed locally every time - on the cloud, this would not be possible due to different sensor environments. By generalizing the data during the preprocessing on the edge it is not only faster computable by the cloud but it might also be very hard to reconstruct the original data because backtracking the layers of a DNN is a difficult problem. This is another advantage because it can be used to protect sensitive user data and the cloud never has access to the original data. Finally, by distributing the system onto many edge devices a certain layer of redundancy is added. In case of failure of an edge device only a small part of the network will not work properly, which might even be covered by other edge devices. This makes the system more resistant to typical attacks like DDoS because DDoSing an edge device would be useless and the cloud is less important to the entire system. Strengthening this single point of failure is especially an important feature for companies.

5. Scheduling of Offloading

In this section, we want to take a look at two recent scheduling strategies for edge assisted computing of DNNs. We want to clarify what variables exist in the offloading process and how they affect the offloading decision.

5.1. Preprocessing Strategy

The general idea of this strategy is that the intermediate data will be smaller than the raw data. Therefore the data are first processed on the edge device and at a strategically chosen point uploaded to the cloud which does all the remaining computation. The offloading point is chosen based on the following factors. The first variable to consider is the architecture of the DNN. There is an in-depth analysis of layers in state-of-the-art DNNs that sets computational effort and size of output data into relation [1]. For the offloading decision, it is important that as much computational effort as possible is after the offloading as well as offloading after a layer where the output data is small. The next variable is the edge hardware. To accurately predict when to offload it is important to know how much computing power is available on the edge device. If an edge server is used this power may vary too based on current workload. The more computing power is available on the edge, the faster the edge device can compute deeper layers of the DNN where a smaller data output size might reduce uploading time and save bandwidth workload. The next variable is the network bandwidth. Especially on mobile devices like smartphones, the available network can vary strongly between wifi or mobile data. The slower the upload speed of the edge device is the more priority

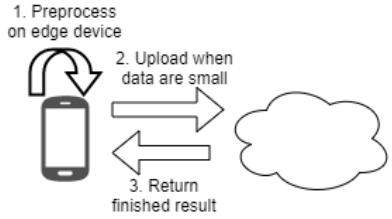
has to be put on uploading at a point where the data are small, while a fast upload might upload at a bigger point but therefore has to execute less computation on the edge device. Finally, the computational power of the cloud also impacts the decision. While the cloud is expected to have a much faster computation speed than the edge device, the speed of the cloud might be limited due to workload at peak times. After considering all these factors the decision when to offload can be made quite accurately and improve the execution speed of DNNs on mobile devices drastically. Furthermore, the same algorithm can be used to optimize the energy usage of the phone by adding consumed energy as a variable that can be traded for speed. If multiple devices run this scheduler are interconnected they could also balance the workload of the cloud intelligently.

5.2. Deadline Strategy

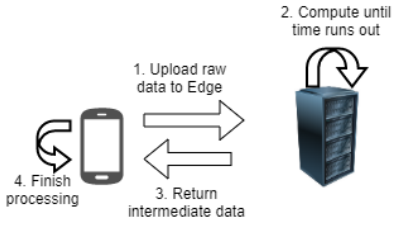
This general idea of this strategy is to trade off accuracy for time. While the previous strategy tried to optimize the time needed for the execution, this strategy expects to have a deadline until when it has to be finished. This can be useful in real-time contexts where the results have to be delivered on time, e.g. for autonomous driving danger has to be spotted at a time when the car is still able to react. If the calculation would be completed later, the result would be of no use. Therefore accuracy is a new variable. To have a variable accuracy a DNN with multiple exit points is used. Unlike the first strategy, the deadline strategy always uploads the raw data but it does not receive the final result instead receives an intermediate value and will finish the computation on the device. By early-exiting the DNN it can be guaranteed that a deadline can be met while the accuracy is as high as possible for the given timeframe. However, to be able to accurately predict deadlines other variables have to be more stable or even fixed. The Edgent algorithm [2]] assumes a model where the data is not processed on the cloud but instead on a very powerful edge device. That offers the benefit of strong and consistent bandwidth. While DNN architecture and the used hardware are still variable for the application they are fixed after the first setup, leaving as little variables as possible to guarantee performance.

6. Conclusion and future work

Deep Learning is an important state-of-the-art methodology with many use cases. Especially with the growth and integration of IoT into everyday life, it is expected to stay very relevant in the future. To make this methodology available for all devices and real-time applications it has to be offloaded since processing DNNs on small or weak devices would take too long or trades off accuracy [8]. Instead of loading everything into the cloud, using Edge Computing offers great advantages. Optimizing the offloading process is important due to the expensiveness of DNNs and there is a lot of potential for research in that area. Future works could be about optimizing Edgent with the NeuroSurgeon baseline since Edgent currently is not optimized in that regard. In the context of IoT, a new algorithm with two offloading stages could be considered, where one stage is computed on the IoT device, one stage on an edge server and one stage in the cloud. This would offer very dynamic computing capabilities, high speed for regular workload due to a strong edge server and strong robustness in case of peak workload due to the cloud.



(a) Neurosurgeon's basic workflow



(b) Edgents basic workflow

Figure 3: Comparison of the two workflows

References

[1] Y. Kang, "Neurosurgeon: Collaborative Intelligence Between the Cloud and Mobile Edge," 2017.

[2] E. Li, "Edge Intelligence: On-Demand Deep Learning Model Co-Inference with Device-Edge Synergy," 2018.

[3] "Edge Computing Model," https://upload.wikimedia.org/wikipedia/commons/b/bf/Edge_computing_infrastructure.png, [Online; accessed 29-September-2019].

[4] H. Khelifi, "Bringing Deep Learning at the Edge of Information-Centric Internet of Things," 2019.

[5] H. Li, "Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing," 2018.

[6] C. Towers-Clark, "Big Data, AI & IoT Part Two: Driving Industry 4.0 One Step At A Time," 2019, [Online; accessed 29-September-2019].

[7] E. Signorini, "HPE and IoT Compute at the Edge," 2016.

[8] Y. Li and T. Strohmer, "What Happens on the Edge, Stays on the Edge: Toward Compressive Deep Learning," 2019.

doi: 10.2313/NET-2020-04-1_01