On-Device Split Inference for Edge Devices

A literature review

Bora Kozan

Supervisors: Qing Wang, Mingkun Yang, Ran Zhu

EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Bora Kozan
Final project course: CSE3000 Research Project
Thesis committee: Qing Wang, Mingkun Yang, Ran Zhu, Johan Pouwelse

Abstract

Nowadays, the popularity of machine learning and artificial intelligence algorithms is very high. A new research direction has emerged where the machine learning algorithms are executed on resource-constrained embedded devices. With the development of the Internet of Things paradigm, these edge devices are deployed in a lot of places. Due to the limited resources of embedded devices, it is difficult to bring machine learning algorithms to them. This is where the on-device split inference comes in. It is possible to distribute the inference between multiple edge devices and the cloud so that the edge devices can execute the inference of complex machine learning models. This paper presents a systematized literature review of papers that focus on on-device split inference for edge devices. The papers are analyzed and compared based on pre-determined questions and displayed based on several categories.

1 Introduction

Machine learning and artificial intelligence (AI) algorithms are commonly used nowadays. Usually, they are run on the cloud on powerful hardware. However, there are several benefits to running these algorithms on embedded devices such as microcontrollers. Bringing AI to Internet of Things (IoT) devices and microcontrollers, we expand the power of AI [33]. This is where on-device split inference can be utilized to run complex and demanding inference algorithms on multiple devices.

In today's world, IoT is the cornerstone of many applications. There are embedded devices all around us. That's why recently there has been research focused on edge computing. By bringing computation to the origin of data, the processing of the data can be low-latency and highly available [49]. As machine learning and artificial intelligence are quite popular, in recent years, there has been an interest in bringing these heavy algorithms from the cloud to low-power embedded devices and this subject is called Tiny Machine Learning[33].

When edge computing is mentioned, Tiny Machine Learning (TinyML) comes to mind. The idea of TinyML became popular around 2018 and can be defined as "a paradigm that facilitates running ML on the edge devices with minimal processor and memory requirements; hence, the power consumption of such systems is expected to be within a few milliwatts or less." [9] According to Warden [56], many people in the technology industry and academia suggest that it can be quite handy to run a neural network model on a device with low energy consumption. That is because such a device can be made to be very small and run for a significantly long time which would be very beneficial in many different industries.

This paper is a literature review of split machine learning inference on edge devices including embedded systems. "The term "edge" refers to the location where data is collected and processed". Edge devices are the client devices in a system. Some examples can be a smartphone, camera, drone, or self-driving car [41]. This paper aims to give an overview of the existing algorithms and methods that enable on-device split inference. It compares the existing systems by presenting the general categories they fall into and compares the papers based on several features. Lastly, it talks about the real-world applications of on-device split inference.

Research Question

The research focus is to survey the key split inference technologies on edge devices.

Sub-questions:

1. *Which machine learning models and algorithms are used for distributed inference on edge devices?*
   There are different ways on-device split inference can be achieved. There are different machine learning models and different ways to distribute inference to multiple devices.

2. *What are the benefits and the limitations of the currently used on-device inference methods?*
   The distributed inference methods can have different performances depending on a couple of parameters such as the number of nodes. By knowing their strong and weak points, the right one can be chosen for a given application.

3. *Are there any distributed computing and parallelization algorithms that are used for or can be applied to split inference on edge devices?*
   The focus of this survey is the distributed inference of artificial intelligence and machine learning algorithms on edge devices. However, there might be some distributed computing and parallelization algorithms that support and enable on-device split inference.

4. *What are the applications of on-device split inference?*
   There are many different areas in real life where on-device split inference can play a role. Knowing about real world applications can increase the understanding of the topic.

This paper is organized in the following way: section 2 talks about similar work done in the field. section 3 presents how the papers were found, selected, and analyzed. The survey results are displayed in section 4. section 5 talks about the ethical risks present in the research and the precautions taken against them. Some key points about research are discussed in section 6. Lastly, section 7 concludes the paper and talks about possible future work.

2 Related Work

There are several survey papers that look at similar topics as this survey paper. There are quite some papers that review literature around TinyML such as [9], [33], [48], and [1]. In addition to TinyML, there are papers talking about edge devices and IoT such as [43]. Even though these papers contain valuable information about various trends and use cases of TinyML and IoT, they do not focus on distributed/collaborative inference. During the research process, one recent paper which is [45] was found which focuses on distributed deep neural network inference. This survey is similar to that one and in this paper, runtime flexibility categories and the metric categories were inspired by that paper. In addition to that paper, this paper contains some discussion about potential use cases of on-device split inference.

3 Methodology

The literature study presented in this paper is a systematized review. It is not a full systematic review due to time constraints. The goal is to find papers that are the most relevant to the research topic in the given time frame and extract common trends from them.

To increase the review process of papers, a set of inclusion and exclusion criteria was created. A paper needs to meet all the inclusion criteria and not satisfy any of the exclusion criteria to be included in the review.

Inclusion Criteria

- The paper talks about distributed inference on edge devices. (*The focus of this paper*)
- The hardware used includes multiple edge devices, embedded processing units, or microprocessors.(*The scope of this paper*)
- The edge devices are used for inference (*The focus of this paper*)

Exclusion Criteria

- The paper talks about distributed inference but does not focus on edge devices or embedded systems.
- The paper talks about inference but it is not distributed inference.

3.1 Finding papers

At the start of the study, to get a general idea of the topic, Google Scholar was used. After that, three databases were used to create a more systematic review. Those databases are ACM Digital Library, IEEE Xplore, and Scopus.

At this point, it should be mentioned that the initial goal of this survey was to find papers that focused on embedded systems and microcontrollers which is a subset of edge devices. Edge devices can be quite powerful compared to microcontrollers which have a few kilobytes of RAM, and a few megabytes of flash memory, run with a clock speed of a couple tens of megahertz, and consume around 1mW power [56]. However, it turned out that not a lot of papers focus on split inference on microcontrollers and embedded systems. There was quite some focus on tiny machine learning inference on single microcontrollers, so not split inference. In addition, split inference on edge devices was a popular topic. That's why the scope of this paper was extended to split inference on edge devices including embedded systems.

After selecting databases, comes the task of creating queries to find the papers that are the most relevant for the survey. The first step in this process was to find keywords that could be in the papers. The main research question can be divided into three main parts which are distributed part, the inference part, and the edge devices part. Therefore, two sets of keywords were created for those parts. The idea of the keywords came from the papers found in the initial searches at the early stages of the research.

Distributed: split, distributed, collaborative
Inference: inference, artificial intelligence, machine learning, neural network
Embedded systems: TinyML, tiny machine learning, embedded, microcontroller, resource constraint, edge

Using the aforementioned keywords, queries were formed using the query languages of the databases. In order to narrow down the search, only the abstracts or the metadata of the papers were considered in the queries. If the number of retrieved papers exceeded 100, only the first 100 papers were scanned for review due to time constraints. The retrieved papers were first scanned to check if they met the inclusion criteria and did not satisfy the exclusion criteria. The scan consisted of reading the title and the abstract and a quick look at the whole paper. Based on the inclusion and exclusion criteria, the papers were either selected for the review or ignored. The following list shows how many papers were retrieved, scanned, and reviewed from each database. The item Other represents papers found during the initial search or papers found as citations on the other reviewed papers. In total, 50 papers were reviewed and included in this survey.

ACM Digital Library
Number of papers retrieved with the query: 59
Number of papers scanned: 59
Number of papers reviewed: 13

IEEE Xplore
Number of papers retrieved with the query: 264
Number of papers scanned: 100
Number of papers reviewed: 15

Scopus
Number of papers retrieved with the query: 1485
Number of papers scanned: 100
Number of papers reviewed: 15

Other
Number of papers reviewed: 7

3.2 Analysis of the selected papers

To streamline the survey and make it easy to compare the papers, a set of questions was created. All selected papers were read with the goal of finding answers to the created questions.

What data/information to extract?

- What device is used? (Related sub-questions: 1,2)
- What machine learning or artificial algorithm is the paper based on? (For example, if the paper is talking about splitting CNN, the answer for this question is CNN) (Related sub-questions: 1, 3)
- What optimization and preprocessing methods are they using? (For example: pruning, quantization, etc.) (Related sub-questions: 3)

- How are they achieving the distribution of the inference task? (For example, by splitting a neural network) (Related sub-questions: 1)
- Are there any performance gains? (Related sub-questions: 2)
- What are the benefits of the method? (For example: faster inference) (Related sub-questions: 2)
- What are the shortcomings of the method? (For example: reduced accuracy) (Related sub-questions: 2)
- What is the purpose or the use case of the method? (Related sub-questions: 4)

After reviewing some papers and looking at [45], based on common trends observed, additional questions were created to further analyze and compare the papers. The inference distribution question was inspired by [40] and the adaptivity question was inspired by [45].

Additional Questions

- What is the inference distribution of the proposed method? (horizontal, vertical, or both)
- Which programming libraries are they using?
- Is the method proposed static or adaptive during runtime?

4 Survey Results

4.1 Inference Distribution

As Murshed suggests, there are two main ways that a neural network can be distributed: horizontally and vertically. Horizontal distribution means that the inference task is split between devices that are at the same level in an edge-cloud architecture.[40] Since the focus of this paper is split inference on edge devices, only horizontal distribution on edge devices is considered in this paper. On the other hand, vertical distribution is when the inference is distributed among various layers of an edge-cloud architecture.[40] Again, because of the scope of this paper, the vertical inference distribution methods described in this paper are selected because there is some inference task being executed on an edge device.

According to the proposed methods in the reviewed papers, the inference distribution seems to be the main factor that affects how the system architecture looks. It makes sense to make it the main characteristic to categorize different on-device split inference methods. That's why in this section, they are discussed separately. The first two subsections are about distributing the inference horizontally and vertically, respectively. The last subsection talks about methods that incorporate both inference distributions. Table 1 list presents the papers that fall into each category. Figure 1 illustrates both of the inference distribution types.

Horizontally Distributed Inference

As an example of horizontal inference distribution, the Network-of-Neural Networks (NoNN) method can be presented. [6] It is a CNN-based algorithm that is designed with network science in mind. The inference task is handled by memory and communication-aware students. The students are trained with one section of the teacher's knowledge that does not overlap with other students' sections. That's why there is very little communication until the final layer of the CNN. Low communication cost seems to be the main benefit of this method. In addition, the student modules can be sized based on the students' hardware constraints. NoNN looks promising in the experiments. NoNN using two Raspberry Pi's was tested against the teacher network on a single Raspberry Pi's. NoNN led to 12.2 reduced latency and 14.3 times reduced energy consumption.[6]

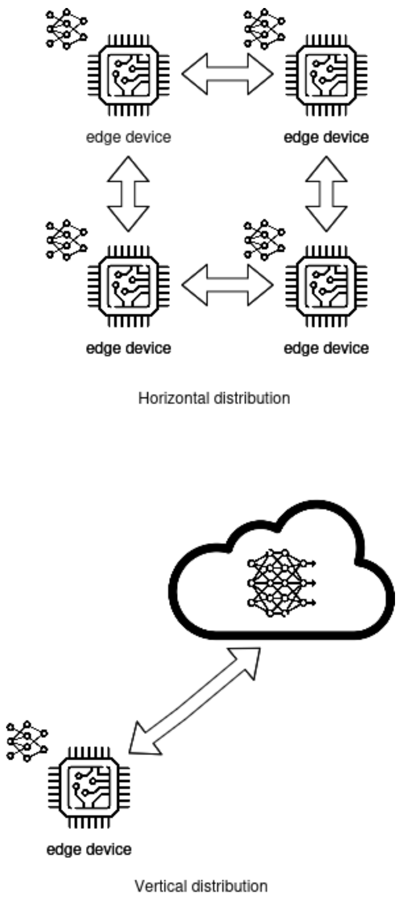

Figure 1: Horizontal distribution vs vertical distribution [16]

In horizontal inference distribution, there are two main ways two split a neural network: semantically and layer splitting [54]. The NoNN method mentioned previously is an example of semantic split. In the semantic case, the neural network weights are grouped into logical sets that recognize certain features. That's why the created sub-networks don't have to communicate with each other until the later stages of execution [6, 54] which can lead to systems with low communication overhead. Furthermore, the generated models can run independently since they don't have to communicate with each other, so they can run in parallel which greatly reduced latency [54]. On the other hand, Tuli et al. suggests that semantic splitting results in decrease in accuracy because of the reduced communication between the neural network sections [54]. In addition, semantic splitting requires training for the

Table 1: The inference distribution types of the methods in the reviewed papers

Inference Distribution Type	Number of papers	Papers
Horizontal	25	[2, 7, 10–14, 19, 20, 23, 26, 29, 39, 42, 51, 54, 55, 58, 60, 63–68]
Vertical	21	[3, 5, 8, 15, 17, 18, 24, 27, 30–32, 35, 36, 38, 44, 47, 52, 57, 59, 61, 62]
Both	4	[4, 37, 46, 53]

created sub-models [54] since they are essentially neural networks themselves and not just simple subsections of a neural network. The papers [60] and [10] also utilize semantic splitting in their proposed methods.

The other common method of neural network splitting in the case of horizontal inference distribution is layer-wise splitting [54]. It is when a pre-trained neural network is split in between the layers [54]. However, in some cases such as [51], the individual layers might also be split. The sections of the neural network is distributed over multiple devices. This structure means that the intermediate results must be transmitted between the devices which suggests high communication costs [54]. Moreover, Tuli et al. argues that layer-wise splitting leads to better accuracy [54]. The proposed methods in [64] and [39] can be given as examples for horizontal inference distribution with layer-wise splitting.

The SplitCNN method [54] stands out here as it utilizes both layer and semantic splitting. It intelligently decided between those two splitting techniques based on latency requirements. Overall, horizontal inference distribution on edge devices are very beneficial as it allows complex neural networks to be employed on the edge, removing the need for a cloud server, and making edge devices more intelligent and useful.

Vertically Distributed Inference

The PArtNNer [18] method can be presented as an example for vertically distributed inference. PArtNNer [18] is a "platform-agnostic adaptive system" that is designed to automatically partition DNNs. The method doesn't split layer by layer but rather uses high-level blocks for partitioning that is tailored for DAG topology DNNs. The advantage of PArtNNer is that it doesn't have to look at the hardware used for inference beforehand. It only performs run-time measurements of end-to-end inference latency solely on the edge device and based on the custom heuristics created for itself, the PArtNNer algorithm can dynamically select the optimal DNN partition point. So the partition point is selected not just for the particular edge-cloud platforms but also for the current operating conditions. However, one significant drawback of this method is that currently, it requires exactly on e edge device and one cloud server. In addition, it can be very communication heavy between the edge and the cloud because it has to transmit intermediate feature maps with full precision. However, the experiments suggest that it can provide up to 21.1 times improvement in end-to-end inference latency compared to running a DNN only on the edge and up to 6.7 times improvement compared to running a DNN only on the cloud.[18]

One of the prominent techniques used for distributing inference tasks is to split the neural network model and putting one portion of it to the edge device and the other portion to the cloud server. The papers that employ this scheme are [3, 17, 18, 24, 27, 30–32, 35, 38, 44, 47, 52, 59, 62]. The edge collects the data and runs the local portion. Then it transmits the intermediate result to the cloud and the cloud executes the rest of the model. This has several benefits. It is possible that this scheme increases the throughput [3, 47] and decrease latency [47]. Since part of the model is executed on the edge device, the edge device doesn't have to send raw data to the server. It only sends the intermediate output which can be less costly in terms of communication and energy [47]. This is also beneficial for privacy sensitive systems because the raw data stays on the edge device and not sent to the server over the network [52]. In the split computing method proposed in [52], the partition point can be adjusted to how much computation the edge and the server has to do, size of the intermediate output to be transmitted, or for privacy.

Splitting a neural network vertically can be advantageous for privacy as well. If the partition point is close to the initial layers of a neural network, the intermediate output is similar to the actual data [52]. However, if the neural network is split at a later stage, the intermediate output is resembles the actual data a lot less since it only contains information that is necessary for the inference [52]. [24] stands out here because it has two split points. The head of the model is executed on the edge devices and then the middle section of the model is offloaded to the server. After that, the tail of the model also run on the edge so that the output of the inference is only known to the edge [24]. In addition, [38] also takes extra steps for privacy. Based on a random value, the edge rearranges the output of the DNN. Again, the DNN is split, and the edge executes the head and the server executes the tail. However, the server cannot know the meaning of the output without the random number generated on the edge. So, the generated output is sent back to the edge and using the random number generated earlier, the edge interprets the output [38].

Another practice that is commonly used in vertical inference distribution is have two models: One lightweight model for the edge device and one heavyweight model for the cloud server. The papers that adopt this strategy are [5, 8, 15, 36, 57, 61]. In this practice, the inference is done fully on the edge or fully on the server. Either the input is analyzed as in Appeal-Net [36], or the lightweight model on the edge is executed and it outputs a confidence value as in the "Use of Classifiers after TinyML" section in [5]. According to the input analysis or the outputted confidence value, the input is marked as simple or complex. For simple inputs, the local lightweight model is used and the cloud is not used. For complex inputs, the inference is offloaded to the heavyweight model in the cloud.

Lastly, early exit strategies frequently come up in vertical inference distribution. The idea of an early exit is that some inputs might be simple and they might not need to be processed using all the layers of a neural network to get an inference with good accuracy [37]. With early exits, a neural network model can make classifications using the early layers where some features might already be learned well enough [59]. C. Luo et al. present that models that incorporate early exits have two issues [8]. One of the issues is that the early exit points have low accuracy because they don't utilize the full model and can only work with low-level features [8]. The other drawback is that placing an early exit to the model may decrease the accuracy of the final exit (the full model) [8]. That's because when updating the weights of the model, the accuracy of the early exits will be given importance as well and this may come at the cost of reducing the accuracy of the final exit of the model [8]. However, early exits require less computation power, and that makes it possible to execute a model until the early exit locally and get an inference result on an edge device [37]. The papers that use this method are [8, 32, 37, 59].

Methods That Include Both Vertical and Horizontal Inference Distribution

Some of the papers proposed systems that include two types of inference distribution at the same time. One example is the Multi-Compression Scale DNN Inference Acceleration (MCIA) system [46]. In this paper, Qi et al. present a whole architecture consisting of a cloud server, an edge server, a base station, and a group of heterogeneous end devices. The cloud is first responsible for, training and learning with deep reinforcement learning and also configuring and compressing the DNN. The cloud then creates an optimizer for resource allocation and computation offloading. This optimizer is then provided to the edge server. The edge server utilizes this optimizer for decision-making for DNN inference tasks. The end server receives an inference task request from end devices and the end server optimizer makes a decision based on the status of the environment (network bandwidth, computational resource status, end device status). The optimizer decides the model version, partitioning point, and bandwidth and computational resource allocations. The end server talks to the end devices and they all complete a DNN inference task [46].

The papers that propose a method that incorporates both vertical and horizontal inference distribution are [4, 37, 46, 53]. In these methods, the inference tasks are divided between multiple edge devices and also multiple servers.

4.2 Which artificial intelligence or machine learning technique is the base of the methods?

All of the papers that were reviewed talk about distributing inference tasks. They start with a model and then use various techniques mentioned in the previous section. The majority of the papers focused on generic DNNs while another significant portion of the papers focused specifically on convolutional neural networks (CNN) which are a type of neural network that excels in "computer vision, speech recognition, virtual/augmented reality, and other fields"[64].

Table 2 displays the algorithms the papers are based on. The other category contains papers that do not focus on DNNs or CNNs. [15] utilizes a shallow feed-forward neural network and also a CNN. [68] doesn't focus on machine learning specifically and proposes a method for distributed tasks in general. [4] employs a simple statistical model based on linear regression. [5] uses both tinyML models and simple machine learning classifiers. Lastly, [2] talks about a system that is based on recurrent neural networks.

Table 2: The algorithms the reviewed papers are based on

Algorithm type	Number of papers	Papers
DNN	34	[7, 8, 11, 13, 14, 17, 18, 20, 23, 24, 27, 30–32, 35–39, 42, 44, 46, 47, 51–55, 57, 59–63]
CNN	11	[3, 10, 12, 19, 26, 29, 58, 64–67]
Other	5	[2, 4, 5, 15, 68]

4.3 What are the optimization and preprocessing methods that stand out?

All the reviewed papers are talking about a system that aims to bring machine learning inference to the edge which is an ambitious goal. While doing so they utilize some preprocessing and optimization techniques. Two main methods stood out during this review: the DAG representation and pruning. Note that here, the methods that are related to neural networks and inference distribution are given importance. The application-specific techniques that are just about manipulating the input are not considered here. First, there is converting a neural network to a directed acyclic graph (DAG) representation. This makes it possible to run various algorithms on the neural network model. The papers that employ this technique are [18, 30, 47, 62]. For example, in [47], after converting the DNN to a DAG, the Max-flow Min-cut algorithm is applied to the DAG to partition the DNN.

Another technique that was encountered during the review was pruning. Pruning is removing some parts of the neural network that don't contribute much, therefore decreasing the size of the model without sacrificing much from accuracy [22]. The papers [14] and [10] use this method where they prune less important filters in a neural network to compress the model.

4.4 Are the methods static or adaptive during runtime?

The methods in the reviewed papers were designed to optimize based on several metrics such as accuracy, latency, communication overhead, memory consumption, computation costs, etc. In some papers, the system is static. The optimizations were done during the design. Once it is deployed, it takes predetermined actions. However, several other systems are adaptive. They make decisions during run-time and adjust some system parameters to further increase the performance of the system. Table 3 displays which papers introduce adaptive systems and which ones introduce static systems.

From Table 3, it can be seen that more than half of the papers are talking about a system that is adaptive. That is

Table 3: Whether or not the systems proposed in the reviewed papers are adaptive

Adaptive during run-time?	Number of papers	Papers
Static	20	[2, 5, 7, 10, 15, 27, 31, 36, 38, 39, 44, 51–53, 55, 58–60, 63, 68]
Adaptive	30	[3, 4, 8, 11–15, 17, 19, 20, 23, 24, 26, 29, 30, 32, 35, 37, 42, 46, 47, 54, 57, 61, 62, 64–67]

not surprising considering that the systems include edge devices. The edge environments are usually dynamic and it makes sense to implement some adaptability to make the system more robust. There are several parameters that adaptive systems monitor and adjust. One of the most common parameters is the network condition. One example is [24] where the model is split between the edge and the server and the split point is adjusted in real-time based on the wireless channel state to minimize latency. Another system that stands out here is RobustDice [19]. It can continue with the inference task and manages to maintain accuracy even if one or more edge devices fail [19]. Similarly, the system proposed in [26] can handle device failures. Furthermore, the system in [42] supports joining and leaving devices. Another metric is the computation power of the edge devices. For example, in [65], if an edge device is fast, it will get assigned more tasks compared to the slower edge devices. In addition, another example for this case is OfpCNN [64]. OfpCNN has an online optimization stage where it predicts the computation time of CNN layers on the devices based on real-time CPU loads of the devices and network conditions and then partitions the CNN model to minimize the latency [64]. Using these various techniques, these systems adapt to the environment conditions and execute inference tasks robustly.

4.5 Which programming libraries are being used?

Table 4: Programming libraries used in the reviewed papers

Programming library	Number of papers	Papers
PyTorch	14	[2, 3, 12–14, 17, 18, 31, 55, 61, 62, 64–66]
Tensorflow	4	[10, 51, 52, 60]
Other	7	[68]:DeepThings, [17]:grpc, [60]:CUDA, [39]:MXNet(modified), [54]:COSCO, [37]:Keras, [18]:Pillow,OpenCV

A lot of the reviewed papers didn't stop after they came up with the theoretical design of a system. They implemented their system to evaluate their performance. Some of the papers shared which programming libraries they used. Looking at what programming libraries are used might be a good way to see the common trends among the papers. Table 4 shows the used programming libraries and the papers that mentioned them. Note that some papers used multiple libraries. It is clear that PyTorch is quite popular. It makes sense to use programming libraries that make it very easy to work with machine learning models. They make it easy to focus on the important parts of the system instead of the small implementation details.

4.6 Which devices are being used?

All the systems mentioned in the reviewed papers have the goal of bringing intelligence to the edge. That's why it makes sense to see what these papers consider as edge devices. This can give a good idea about the performance of the proposed systems. In this survey, the edge devices the papers used to deploy and evaluate are reviewed. The results can be found in Table 5. The simulation section lists the papers that didn't deploy their system to actual devices but ran simulations with their system instead. Note that some papers used two or more different edge devices.

Table 5: Edge devices and simulation usage in the reviewed papers

Device or simulation	Number of papers	Papers
Raspberry Pi model [34]	21	[7, 8, 10–12, 14, 17, 18, 20, 26, 30, 32, 47, 52, 55, 60, 63–67]
NVIDIA Jetson model [25]	7	[3, 18, 19, 31, 58, 60, 61]
Microcontroller-based board	2	[4]: LPC2148 from NXP; [51]: MSP430FR5994 from TI
Smartphone	3	[17, 23, 39]
Laptop or PC	6	[8, 12, 24, 37, 47, 64]
Simulation	15	[2, 5, 13, 27, 29, 35, 36, 38, 44, 46, 54, 57, 59, 62, 68]
Other	2	[18]: Intel Neural Compute Stick 2, Coral Dev Board; [7]:Odroid-XU4S

From Table 5, it is obvious that a Raspberry Pi model is the most popular choice. A Raspberry Pi is a single-board computer that is quite small compared to a conventional laptop. It makes sense to use it on the edge due to its compactness. It is used in commercial IoT products [21, 50]. It is also a very popular product among hobbyists and makers and there for there is lots of documentation and information available online for Raspberry Pi's. That's why it is not surprising that it is commonly used in research as well. The second most common platform is simulation. This is also expected as some researchers are not interested in actually developing the systems they design. It is enough for them to evaluate their designs with simulations and experiments. After the simulation category comes NVIDIA Jetson models with 7 papers using them. NVIDIA Jetson models are similar to Raspberry Pi's in the sense that they are also compact single-board computers [25]. However, they have much more computational capabilities than Raspberry Pi's especially because they have graphical processing units [25]. This makes them ideal for machine learning tasks. The other categories that are not as popular are smartphones, laptops, PCs, and other development boards. All in all, there is some variety in the edge devices used to develop and test distributed inference.

4.7 Use Cases and Real World Applications

During the review, some papers talked about various applications of on-device split inference. Currently, there are several fields and applications that distributed inference on the edge can be quite useful. As the first example, [15] can be presented where a verticle inference distribution architecture for healthcare applications is presented. In some healthcare applications, large amounts of data need to be transferred to the cloud for processing with machine learning [15]. However, using edge devices can be very beneficial in this case since they can make it possible for important applications with strict latency requirements to make real-time decisions [15]. They also developed an example system to detect arrhythmia based on ECG signals to demonstrate their idea [15]. Thanks to the popularity of small wearable devices and IoT technologies, using collaborative on-device inference can be very promising in the healthcare industry. A similar example is given in [10]. The paper mentions that collaborative inference tasks can be employed in an environment with multiple IoT devices. A possible scenario is people in a meeting can use their smartphones to perform collaborative automatic speech recognition-based applications such as automatic transcription generation [10].

Another real-world application is presented in [4]. The paper talks about an architecture for predictive environmental sensor networks that can be deployed over large geographic areas. The architecture has multiple levels and node types. The idea is to have multiple Sensing nodes placed over an area. These nodes collect and share data between them. On the next level, there are Computation nodes. They perform statistical analysis and also prediction. In addition, other higher-level nodes provide an interface for government officials and also for the public. These high-level nodes can also perform predictions using the collected data. This way a large geographical area can be monitored. The researchers also develop a real system using this architecture to predict river flooding. The goal is to monitor several environmental parameters such as rainfall and air temperature and predict when the rivers are going to flood. That way the area around the rivers can be evacuated and other emergency protocols can be utilized early on [4].

Furthermore, there can be industrial use cases for on-device split inference. The paper [14] introduces a system called EdgeDI and talks about its potential applications. Fang et al. say that EdgeDI can be deployed on devices near a production line and run DNNs collaboratively to identify product defects or detect equipment faults in real-time [14]. They also mention that EdgeDI can be employed for vision tasks such as object detection and tracking on unmanned aerial vehicles (UAV) [14]. The computation power of UAVs can be limited so it can be valuable to run inference tasks collaboratively in a swarm. Another industrial application design is presented in [10]. They design a system where their splitCNN system is used to detect anomalies based on vibration. The system utilizes multiple devices with sensors placed in different locations on the object of interest. These devices use splitCNN to collaboratively run the CNN inference task [10]. This system can be beneficial since it can reduce the latency and remove the need for energy-intensive data transmission even though it may require advanced signal processing on the data [10].

Lastly, video processing is another potential application of on-device distributed inference. DNNs are commonly used for real-time object detection [31]. There are deep learning applications such as object detection and tracking, face recognition, and activity recognition where the images and videos taken by IoT devices are used [27]. However, it is difficult to process the data and use DNNs on IoT devices as they have limited resources [27, 28]. The on-device collaborative inference can be very effective in these scenarios. The papers [31] and [28] propose methods for video processing where they spit the inference task between the edge and the cloud to reduce the computational load and latency.

5 Responsible Research

Of course, there is a risk of bias in this review as is in all research. This study was conducted by a Bachelor of Computer Science and Engineering student who was the only reviewer. Therefore there was only one person responsible for searching, selecting, and reviewing papers. The first opportunity for bias to affect the review is the queries. In order to properly choose keywords for the queries, an brief initial study was conducted. In this study, a couple of surveys on tiny machine learning were read to get an idea about how the surveys in the field look like. In addition, a couple of papers that were related to the topic were skimmed over and the important keywords were noted. Based on the keywords found, the queries for the search engines were created. The full queries can be found in Appendix A.

Furthermore, all papers are unique and different. In order to compare them, the same questions presented in subsection 3.2 were asked. Based on the answers, the papers were categorized and analyzed. Of course, again only one person was responsible for understanding and recording the information presented in the papers which might be a source of bias. However, in order to validate the study, anyone can easily reproduce it since the entire methodology is clearly presented in section section 3 and all of the reviewed papers are listed in the references.

6 Discussion

Overall, it can be said that this review sheds some light on common trends of on-device split inference. However, it should be pointed out that this review was done with 50 papers which might not be considered high in the scope of literature reviews. This was due to time constraints since this was a ten-week Bachelor project. That's why this is a systematized review and not a systematic one. Furthermore, subquestions 2 and 3 were useful when reviewing a paper and trying to understand the contents but they were not very handy for comparing the papers. Most papers present results that are positive and don't mention a lot of negative points about their work. In addition, due to the short time, it is difficult to learn enough about the complex algorithms to compare them. That's why in this paper, I talked about the general architectures and overall structures of the methods without going into details about very specific algorithms that are utilized in some parts of the systems.

Another point to mention here is that the initial goal of this literature review was to focus on on-device split inference on microcontrollers. However, finding papers that focused on distributed inference for microcontrollers turned out to be quite difficult. That's why the scope of this paper was extended to include edge devices. In the end, only two papers were found that used microcontrollers as it can be seen in Table 5. This is a bit surprising since there is a lot of research revolving around TinyML as seen in [9], [33], [48], and [1]. However, it is understandable that developing systems for microcontrollers is a lot more difficult compared to edge devices. For example, edge devices such as Raspberry Pi's or smartphones run operating systems that provide useful tools. For example, operating systems provide interfaces for wired and wireless communication and also support more libraries. However, the options are very limited for microcontrollers, and the developers (in this case researchers) have to do a lot more work to get a system working on microcontrollers. Furthermore, as the technology evolves, computers are getting smaller and more power-efficient. Thus they are getting easier to use them on the edge as embedded devices. For example, Jetson AGX Orin Series single-board computers have a footprint of 100 millimeters by 87 millimeters and come with a powerful GPU [25]. That's why focusing on edge devices that are a bit more powerful than microcontrollers makes sense as they are used a lot in the real world. All in all, the reason for the low number of papers that focus on distributed inference on microcontrollers is likely to be that they are difficult to work with compared to other more powerful edge devices, and the other edge devices are commonly used in many applications.

7 Conclusion and Future Work

In conclusion, this paper presents the results of a literature survey about on-device split inference on edge devices. First, the papers were categorized based on the distribution style of the inference task. This is the main parameter that shapes the architecture of an on-device split inference system. Furthermore, the papers were analyzed according to the machine learning models they are based on, the adaptability of the proposed systems, the programming libraries, and the devices they used. In addition, several use cases of on-device distributed inference were presented. In the future, a systematic review that aims to find all the related papers that focus on this topic can be done. This will result in the review of more papers so it can give a better view of the overall trends in this research area. Moreover, since distributed inference on microcontrollers is not a heavily focused topic, research can be conducted based on findings from distributed inference on edge devices to design a system for distributed inference on microcontrollers and similar embedded systems.

References

[1] Youssef Abadade et al. "A Comprehensive Survey on TinyML". In: *IEEE Access* 11 (2023). Conference Name: IEEE Access, pp. 96892–96922. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2023.3294111. URL: https://ieeexplore.ieee.org/document/10177729 (visited on 05/02/2024).

[2] P. Abudu and A. Markham. "Distributed communicating neural network architecture for smart environments". In: Proceedings - 2019 IEEE International Conference on Smart Computing, SMARTCOMP 2019. 2019, pp. 233–240. DOI: 10.1109/SMARTCOMP.2019.00058. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85070893625&doi=10.1109%2fSMARTCOMP.2019.00058&partnerID=40&md5=69015d1f1afdad96b662a070a229b758.

[3] Mario Almeida et al. "DynO: Dynamic Onloading of Deep Neural Networks from Cloud to Device". In: *ACM Trans. Embed. Comput. Syst.* 21.6 (Oct. 2022). Place: New York, NY, USA Publisher: Association for Computing Machinery. ISSN: 1539-9087. DOI: 10.1145/3510831. URL: https://doi.org/10.1145/3510831.

[4] Elizabeth A. Basha, Sai Ravela, and Daniela Rus. "Model-based monitoring for early warning flood detection". In: *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*. SenSys '08. event-place: Raleigh, NC, USA. New York, NY, USA: Association for Computing Machinery, 2008, pp. 295–308. ISBN: 978-1-59593-990-6. DOI: 10.1145/1460412.1460442. URL: https://doi.org/10.1145/1460412.1460442.

[5] Adarsh Prasad Behera et al. "Improved Decision Module Selection for Hierarchical Inference in Resource-Constrained Edge Devices". In: *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. 125. New York, NY, USA: Association for Computing Machinery, Oct. 2, 2023, pp. 1–3. ISBN: 978-1-4503-9990-6. URL: https://doi.org/10.1145/3570361.3615732 (visited on 05/06/2024).

[6] Kartikeya Bhardwaj, Wei Chen, and Radu Marculescu. "New directions in distributed deep learning: bringing the network at forefront of IoT design: invited". In: *Proceedings of the 57th ACM/EDAC/IEEE Design Automation Conference*. DAC '20. Place: Virtual Event, USA. IEEE Press, 2020. ISBN: 978-1-4503-6725-7.

[7] Kartikeya Bhardwaj et al. "Memory- and Communication-Aware Model Compression for Distributed Deep Learning Inference on IoT". In: *ACM Transactions on Embedded Computing Systems* 18.5 (Oct. 8, 2019), 82:1–82:22. ISSN: 1539-9087. DOI: 10.1145/3358205. URL: https://dl.acm.org/doi/10.1145/3358205 (visited on 05/28/2024).

[8] C. Luo et al. "BSL: Sustainable Collaborative Inference in Intelligent Transportation Systems". In: *IEEE Transactions on Intelligent Transportation Systems* 24.12 (Dec. 2023), pp. 15995–16005. ISSN: 1558-0016. DOI: 10.1109/TITS.2023.3308370.

[9] Luigi Capogrosso et al. "A Machine Learning-Oriented Survey on Tiny Machine Learning". In: *IEEE Access* 12 (2024). Conference Name: IEEE Access, pp. 23406–23426. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2024.3365349. URL: https://ieeexplore.ieee.org/document/10433185 (visited on 05/02/2024).

[10] J. Chen et al. "Split Convolutional Neural Networks for Distributed Inference on Concurrent IoT Sensors". In: Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS. Vol. 2021-December. 2021, pp. 66–73. DOI: 10.1109/ICPADS53394.2021.00014. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85129871643&doi=10.1109%2fICPADS53394.2021.00014&partnerID=40&md5=2169ba50df8554b8f672a1443c313b15.

[11] Y. Chen et al. "Self-aware Collaborative Edge Inference with Embedded Devices for Task-oriented IIoT". In: IEEE Vehicular Technology Conference. 2023. DOI: 10.1109/VTC2023-Fall60731.2023.10333778. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85181172811&doi=10.1109%2fVTC2023-Fall60731.2023.10333778&partnerID=40&md5=a671aba791015db4079f5bb656aaa6d4.

[12] Yanming Chen et al. "EdgeCI: Distributed Workload Assignment and Model Partitioning for CNN Inference on Edge Clusters". In: *ACM Transactions on Internet Technology* 24.2 (May 6, 2024), 10:1–10:24. ISSN: 1533-5399. DOI: 10.1145/3656041. URL: https://dl.acm.org/doi/10.1145/3656041 (visited on 05/13/2024).

[13] F. Xue et al. "EdgeLD: Locally Distributed Deep Learning Inference on Edge Device Clusters". In: *2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. 2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS). Journal Abbreviation: 2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS). Dec. 14, 2020, pp. 613–619. DOI: 10.1109/HPCC-SmartCity-DSS50907.2020.00078.

[14] Weiwei Fang et al. "Joint Architecture Design and Workload Partitioning for DNN Inference on Industrial IoT Clusters". In: *ACM Transactions on Internet Technology* 23.1 (Feb. 23, 2023), 7:1–7:21. ISSN: 1533-

5399. DOI: 10.1145/3551638. URL: https://dl.acm.org/doi/10.1145/3551638 (visited on 05/13/2024).

[15] Bahar Farahani, Mojtaba Barzegari, and Fereidoon Shams Aliee. “Towards Collaborative Machine Learning Driven Healthcare Internet of Things”. In: *Proceedings of the International Conference on Omni-Layer Intelligent Systems*. COINS ’19. event-place: Crete, Greece. New York, NY, USA: Association for Computing Machinery, 2019, pp. 134–140. ISBN: 978-1-4503-6640-3. DOI: 10.1145/3312614.3312644. URL: https://doi.org/10.1145/3312614.3312644.

[16] *Free Icons and Stickers - Millions of images to download*. Flaticon. URL: https://www.flaticon.com/ (visited on 06/23/2024).

[17] Kaihua Fu et al. “QoS-aware irregular collaborative inference for improving throughput of DNN services”. In: *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. SC ’22. Place: Dallas, Texas. IEEE Press, 2022. ISBN: 9784665454445.

[18] Soumendu Kumar Ghosh et al. “PArtNNer: Platform-Agnostic Adaptive Edge-Cloud DNN Partitioning for Minimizing End-to-End Latency”. In: *ACM Trans. Embed. Comput. Syst.* 23.1 (Jan. 2024). Place: New York, NY, USA Publisher: Association for Computing Machinery. ISSN: 1539-9087. DOI: 10.1145/3630266. URL: https://doi.org/10.1145/3630266.

[19] Xiaotian Guo et al. “RobustDiCE: Robust and Distributed CNN Inference at the Edge”. In: *Proceedings of the 29th Asia and South Pacific Design Automation Conference*. ASPDAC ’24. Place: <conf-loc>, <city>Incheon</city>, <country>Republic of Korea</country>, </conf-loc>. IEEE Press, 2024, pp. 26–31. ISBN: 9798350393545. DOI: 10.1109/ASP-DAC58780.2024.10473970. URL: https://doi.org/10.1109/ASP-DAC58780.2024.10473970.

[20] Ramyad Hadidi et al. “Robustly Executing DNNs in IoT Systems Using Coded Distributed Computing”. In: *Proceedings of the 56th Annual Design Automation Conference 2019*. DAC ’19. New York, NY, USA: Association for Computing Machinery, June 2, 2019, pp. 1–2. ISBN: 978-1-4503-6725-7. DOI: 10.1145/3316781.3322474. URL: https://dl.acm.org/doi/10.1145/3316781.3322474 (visited on 06/12/2024).

[21] *Homey Pro smart home hub*. Raspberry Pi. URL: https://www.raspberrypi.com/success-stories/homey-pro-smart-home-hub/ (visited on 06/23/2024).

[22] Hengyuan Hu et al. *Network Trimming: A Data-Driven Neuron Pruning Approach towards Efficient Deep Architectures*. July 12, 2016. DOI: 10.48550/arXiv.1607.03250. arXiv: 1607.03250[cs]. URL: http://arxiv.org/abs/1607.03250 (visited on 06/23/2024).

[23] Zhiming Hu et al. “DeepHome: Distributed Inference with Heterogeneous Devices in the Edge”. In: *The 3rd International Workshop on Deep Learning for Mobile Systems and Applications*. EMDL ’19. event-place: Seoul, Republic of Korea. New York, NY, USA: Association for Computing Machinery, 2019, pp. 13–18. ISBN: 978-1-4503-6771-4. DOI: 10.1145/3325413.3329787. URL: https://doi.org/10.1145/3325413.3329787.

[24] J. Lee, H. Lee, and W. Choi. “Wireless Channel Adaptive DNN Split Inference for Resource-Constrained Edge Devices”. In: *IEEE Communications Letters* 27.6 (June 2023), pp. 1520–1524. ISSN: 1558-2558. DOI: 10.1109/LCOMM.2023.3269769.

[25] *Jetson Modules, Support, Ecosystem, and Lineup*. NVIDIA Developer. URL: https://developer.nvidia.com/embedded/jetson-modules (visited on 06/23/2024).

[26] Nihel Kaboubi et al. “Hybrid Partitioning for Embedded and Distributed CNNs Inference on Edge Devices”. In: *Advanced Network Technologies and Intelligent Computing*. Ed. by Isaac Woungang et al. Cham: Springer Nature Switzerland, 2023, pp. 164–187. ISBN: 978-3-031-28180-8. DOI: 10.1007/978-3-031-28180-8_12.

[27] MA Khan et al. “Distributed Inference in Resource-Constrained IoT for Real-Time Video Surveillance”. In: *IEEE SYSTEMS JOURNAL* 17.1 (Mar. 2023), pp. 1512–1523. ISSN: 1932-8184. DOI: 10.1109/JSYST.2022.3198711.

[28] U.A. Khan, S. Kar, and J.M.F. Moura. “Distributed Algorithms in Sensor Networks”. In: *Handbook on Array Processing and Sensor Networks*. 2010, pp. 533–557. DOI: 10.1002/9780470487068.ch17. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84885535496&doi=10.1002%2f9780470487068.ch17&partnerID=40&md5=1832efc59d7a467363c2921d495b943c.

[29] E. Kilcioglu et al. “An Energy-Efficient Fine-Grained Deep Neural Network Partitioning Scheme for Wireless Collaborative Fog Computing”. In: *IEEE Access* 9 (2021), pp. 79611–79627. DOI: 10.1109/ACCESS.2021.3084689. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85107221217&doi=10.1109%2fACCESS.2021.3084689&partnerID=40&md5=e156074912770b7686b11b2dc1e94510.

[30] Z. Kou et al. “Adaptive Deep Inference Framework for Cloud-Edge Collaboration”. In: Proceedings - 2023 International Conference on Computers, Information Processing and Advanced Education, CIPAE 2023. 2023, pp. 46–49. DOI: 10.1109/CIPAE60493.2023.00014. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85182313864&doi=10.1109%2fCIPAE60493.2023.00014&partnerID=40&md5=92f03cee6f555ca1fa8ef4f46aba0998.

[31] JC Lee et al. “A Splittable DNN-Based Object Detector for Edge-Cloud Collaborative Real-Time Video Inference”. In: 2021 17TH IEEE INTERNATIONAL CONFERENCE ON ADVANCED VIDEO AND SIGNAL BASED SURVEILLANCE (AVSS 2021). 2021. ISBN: 978-1-66543-396-9. DOI: 10.1109/AVSS52988.2021.9663806.

[32] E. Li, Z. Zhou, and X. Chen. "Edge intelligence: On-demand deep learning model co-inference with device-edge synergy". In: MECOMM 2018 - Proceedings of the 2018 Workshop on Mobile Edge Communications, Part of SIGCOMM 2018. 2018, pp. 31–36. DOI: 10.1145/3229556.3229562. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85056854955&doi=10.1145%2f3229556.3229562&partnerID=40&md5=76c07b7cd92ecee9709abb52256c8092.

[33] Ji Lin et al. "Tiny Machine Learning: Progress and Futures [Feature]". In: *IEEE Circuits and Systems Magazine* 23.3 (2023). Conference Name: IEEE Circuits and Systems Magazine, pp. 8–34. ISSN: 1558-0830. DOI: 10.1109/MCAS.2023.3302182. URL: https://ieeexplore.ieee.org/document/10284551 (visited on 04/28/2024).

[34] Raspberry Pi Ltd. *Buy a Raspberry Pi*. Raspberry Pi. URL: https://www.raspberrypi.com/products/ (visited on 06/23/2024).

[35] M. Krouka et al. "Energy-Efficient Model Compression and Splitting for Collaborative Inference Over Time-Varying Channels". In: *2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. 2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC). Journal Abbreviation: 2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC). Sept. 13, 2021, pp. 1173–1178. ISBN: 2166-9589. DOI: 10.1109/PIMRC50174.2021.9569707.

[36] M. Li et al. "AppealNet: An Efficient and Highly-Accurate Edge/Cloud Collaborative Architecture for DNN Inference". In: *2021 58th ACM/IEEE Design Automation Conference (DAC)*. 2021 58th ACM/IEEE Design Automation Conference (DAC). Journal Abbreviation: 2021 58th ACM/IEEE Design Automation Conference (DAC). Dec. 5, 2021, pp. 409–414. ISBN: 0738-100X. DOI: 10.1109/DAC18074.2021.9586176.

[37] M. Sbai, N. Trigoni, and A. Markham. "Multiple Early-Exits Strategy for Distributed Deep Neural Network Inference". In: *2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*. 2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT). Journal Abbreviation: 2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT). June 19, 2023, pp. 34–38. ISBN: 2325-2944. DOI: 10.1109/DCOSS-IoT58021.2023.00014.

[38] Mohammad Malekzadeh and Fahim Kawsar. "Salted Inference: Enhancing Privacy while Maintaining Efficiency of Split Inference in Mobile Computing". In: *Proceedings of the 25th International Workshop on Mobile Computing Systems and Applications*. HOTMOBILE '24. New York, NY, USA: Association for Computing Machinery, Feb. 28, 2024, pp. 14–20. ISBN: 9798400704970. DOI: 10.1145/3638550.3641131. URL: https://dl.acm.org/doi/10.1145/3638550.3641131 (visited on 05/13/2024).

[39] Jiachen Mao et al. "MoDNN: Local distributed mobile computing system for Deep Neural Network". In: *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017. ISSN: 1558-1101. Mar. 2017, pp. 1396–1401. DOI: 10.23919/DATE.2017.7927211. URL: https://ieeexplore.ieee.org/document/7927211 (visited on 05/06/2024).

[40] M. G. Sarwar Murshed et al. "Machine Learning at the Network Edge: A Survey". In: *ACM Comput. Surv.* 54.8 (Oct. 2021). Place: New York, NY, USA Publisher: Association for Computing Machinery. ISSN: 0360-0300. DOI: 10.1145/3469029. URL: https://doi.org/10.1145/3469029.

[41] Hasnae Nouhas, Abdessamad Belangour, and Mahmoud Nassar. "Cloud and Edge Computing Architectures: A Survey". In: *2023 IEEE 11th Conference on Systems, Process & Control (ICSPC)*. 2023 IEEE 11th Conference on Systems, Process & Control (ICSPC). Dec. 2023, pp. 210–215. DOI: 10.1109/ICSPC59664.2023.10420123. URL: https://ieeexplore.ieee.org/document/10420123 (visited on 05/28/2024).

[42] Torsten Ohlenforst et al. "Enabling Distributed Inference of Large Neural Networks on Resource Constrained Edge Devices using Ad Hoc Networks". In: *Distributed Computing and Artificial Intelligence, 20th International Conference*. Ed. by Sascha Ossowski et al. Cham: Springer Nature Switzerland, 2023, pp. 145–154. ISBN: 978-3-031-38333-5. DOI: 10.1007/978-3-031-38333-5_15.

[43] Franklin Oliveira et al. "Internet of Intelligent Things: A convergence of embedded systems, edge computing and machine learning". In: *Internet of Things* 26 (July 1, 2024), p. 101153. ISSN: 2542-6605. DOI: 10.1016/j.iot.2024.101153. URL: https://www.sciencedirect.com/science/article/pii/S2542660524000945 (visited on 05/06/2024).

[44] P. Yang et al. "Communication-Efficient Vertically Split Inference via Over-the-Air Computation". In: *2023 IEEE 24th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. 2023 IEEE 24th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC). Journal Abbreviation: 2023 IEEE 24th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC). Sept. 25, 2023, pp. 1–5. ISBN: 1948-3252. DOI: 10.1109/SPAWC53906.2023.10304428.

[45] Federico Nicolás Peccia and Oliver Bringmann. *Embedded Distributed Inference of Deep Neural Networks: A Systematic Review*. May 6, 2024. DOI: 10.48550/arXiv.2405.03360. arXiv: 2405.03360[cs]. URL: http://arxiv.org/abs/2405.03360 (visited on 06/19/2024).

[46] Huamei Qi et al. “Multi-Compression Scale DNN Inference Acceleration based on Cloud-Edge-End Collaboration”. In: *ACM Trans. Embed. Comput. Syst.* 23.1 (Jan. 2024). Place: New York, NY, USA Publisher: Association for Computing Machinery. ISSN: 1539-9087. DOI: 10.1145/3634704. URL: https://doi.org/10.1145/3634704.

[47] R. Yang et al. “DNN Real-Time Collaborative Inference Acceleration with Mobile Edge Computing”. In: *2022 International Joint Conference on Neural Networks (IJCNN)*. 2022 International Joint Conference on Neural Networks (IJCNN). Journal Abbreviation: 2022 International Joint Conference on Neural Networks (IJCNN). July 18, 2022, pp. 01–08. ISBN: 2161-4407. DOI: 10.1109/IJCNN55064.2022.9892582.

[48] Visal Rajapakse, Ishan Karunanayake, and Nadeem Ahmed. “Intelligence at the Extreme Edge: A Survey on Reformable TinyML”. In: *ACM Computing Surveys* 55.13 (July 13, 2023), 282:1–282:30. ISSN: 0360-0300. DOI: 10.1145/3583683. URL: https://dl.acm.org/doi/10.1145/3583683 (visited on 05/02/2024).

[49] Partha Pratim Ray. “A review on TinyML: State-of-the-art and prospects”. In: *Journal of King Saud University - Computer and Information Sciences* 34.4 (Apr. 1, 2022), pp. 1595–1623. ISSN: 1319-1578. DOI: 10.1016/j.jksuci.2021.11.019. URL: https://www.sciencedirect.com/science/article/pii/S1319157821003335 (visited on 05/02/2024).

[50] *Revolution Pi industrial computers*. Raspberry Pi. URL: https://www.raspberrypi.com/success-stories/revolution-pi-industrial-computers/ (visited on 06/23/2024).

[51] R Sahu et al. “DENNI: Distributed Neural Network Inference on Severely Resource Constrained Edge Devices”. In: 2021 IEEE INTERNATIONAL PERFORMANCE, COMPUTING, AND COMMUNICATIONS CONFERENCE (IPCCC). 2021. ISBN: 1097-2641. DOI: 10.1109/IPCCC51483.2021.9679448.

[52] T. Nishio et al. “Demo: Split Computing-Based Privacy-Preserving Image Classification and Object Detection”. In: *2024 IEEE 21st Consumer Communications & Networking Conference (CCNC)*. 2024 IEEE 21st Consumer Communications & Networking Conference (CCNC). Journal Abbreviation: 2024 IEEE 21st Consumer Communications & Networking Conference (CCNC). Jan. 6, 2024, pp. 1092–1093. ISBN: 2331-9860. DOI: 10.1109/CCNC51664.2024.10454718.

[53] Surat Teerapittayanon, Bradley McDanel, and H.T. Kung. “Distributed Deep Neural Networks Over the Cloud, the Edge and End Devices”. In: *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS). ISSN: 1063-6927. June 2017, pp. 328–339. DOI: 10.1109/ICDCS.2017.226. URL: https://ieeexplore.ieee.org/document/7979979 (visited on 06/12/2024).

[54] S. Tuli, G. Casale, and N.R. Jennings. “SplitPlace: AI Augmented Splitting and Placement of Large-Scale Neural Networks in Mobile Edge Environments”. In: *IEEE Transactions on Mobile Computing* 22.9 (2023), pp. 5539–5554. DOI: 10.1109/TMC.2022.3177569. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85130816028&doi=10.1109%2fTMC.2022.3177569&partnerID=40&md5=abf6eaa550fe9c7ae86882c7ce7caebe.

[55] Meng Wang et al. “Model Parallelism Optimization for Distributed DNN Inference on Edge Devices”. In: *2023 IEEE 14th International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)*. 2023 IEEE 14th International Symposium on Parallel Architectures, Algorithms and Programming (PAAP). ISSN: 2168-3042. Nov. 2023, pp. 1–6. DOI: 10.1109/PAAP60200.2023.10391646. URL: https://ieeexplore.ieee.org/abstract/document/10391646 (visited on 05/06/2024).

[56] Pete Warden and Daniel Situnayake. *TinyML : machine learning with TensorFlow Lite on Arduino and ultra-low-power microcontrollers*. First edition. 1 online resource (1 volume) : illustrations vols. Sebastopol, CA: O’Reilly Media, 2019. URL: https://proquest.safaribooksonline.com/9781492052036.

[57] Wen Wu et al. “Accuracy-Guaranteed Collaborative DNN Inference in Industrial IoT via Deep Reinforcement Learning”. In: *IEEE Transactions on Industrial Informatics* 17.7 (July 2021). Conference Name: IEEE Transactions on Industrial Informatics, pp. 4988–4998. ISSN: 1941-0050. DOI: 10.1109/TII.2020.3017573. URL: https://ieeexplore.ieee.org/document/9170818 (visited on 05/27/2024).

[58] X. Guo, A. D. Pimentel, and T. Stefanov. “Automated Exploration and Implementation of Distributed CNN Inference at the Edge”. In: *IEEE Internet of Things Journal* 10.7 (Apr. 1, 2023), pp. 5843–5858. ISSN: 2327-4662. DOI: 10.1109/JIOT.2023.3237572.

[59] X. Xu et al. “Learning-Based Edge-Device Collaborative DNN Inference in IoVT Networks”. In: *IEEE Internet of Things Journal* 11.5 (Mar. 1, 2024), pp. 7989–8004. ISSN: 2327-4662. DOI: 10.1109/JIOT.2023.3317785.

[60] Y. Fang, Z. Jin, and R. Zheng. “TeamNet: A Collaborative Inference Framework on the Edge”. In: *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). Journal Abbreviation: 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). July 7, 2019, pp. 1487–1496. ISBN: 2575-8411. DOI: 10.1109/ICDCS.2019.00148.

[61] Y. Hu et al. “Content-Aware Adaptive Device–Cloud Collaborative Inference for Object Detection”. In: *IEEE Internet of Things Journal* 10.21 (Nov. 1, 2023), pp. 19087–19101. ISSN: 2327-4662. DOI: 10.1109/JIOT.2023.3279579.

[62] Y. Xiao et al. "NAIR: An Efficient Distributed Deep Learning Architecture for Resource Constrained IoT System". In: *IEEE Internet of Things Journal* (2024), pp. 1–1. ISSN: 2327-4662. DOI: 10.1109/JIOT.2024.3375924.

[63] CY Yang et al. "Cooperative Distributed Deep Neural Network Deployment with Edge Computing". In: IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS (ICC 2021). 2021. ISBN: 1550-3607. DOI: 10.1109/ICC42927.2021.9500668.

[64] L. Yang et al. "OfpCNN: On-Demand Fine-Grained Partitioning for CNN Inference Acceleration in Heterogeneous Devices". In: *IEEE Transactions on Parallel and Distributed Systems* 34.12 (2023), pp. 3090–3103. DOI: 10.1109/TPDS.2023.3321755. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85174808130&doi=10.1109%2fTPDS.2023.3321755&partnerID=40&md5=7e0523033bacbcf8e077e122f933b300.

[65] S.Q. Zhang, J. Lin, and Q. Zhang. "Adaptive Distributed Convolutional Neural Network Inference at the Network Edge with ADCNN". In: ACM International Conference Proceeding Series. 2020. DOI: 10.1145/3404397.3404473. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85090595123&doi=10.1145%2f3404397.3404473&partnerID=40&md5=b1cd813d81b522ed38c392f1292ddbcb.

[66] Shuai Zhang et al. "DeepSlicing: Collaborative and Adaptive CNN Inference With Low Latency". In: *IEEE Transactions on Parallel and Distributed Systems* 32.9 (Sept. 2021). Conference Name: IEEE Transactions on Parallel and Distributed Systems, pp. 2175–2187. ISSN: 1558-2183. DOI: 10.1109/TPDS.2021.3058532. URL: https://ieeexplore.ieee.org/document/9353250 (visited on 05/06/2024).

[67] Z. Zhao, K.M. Barijough, and A. Gerstlauer. "DeepThings: Distributed adaptive deep learning inference on resource-constrained IoT edge clusters". In: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. Vol. 37. Issue: 11. 2018, pp. 2348–2359. DOI: 10.1109/TCAD.2018.2858384. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85055264118&doi=10.1109%2fTCAD.2018.2858384&partnerID=40&md5=b87efa304f67ffe0df863b7fe7301f82.

[68] Zhuoran Zhao, Kamyar Mirzazad Barijough, and Andreas Gerstlauer. "Network-level Design Space Exploration of Resource-constrained Networks-of-Systems". In: *ACM Trans. Embed. Comput. Syst.* 19.4 (June 2020). Place: New York, NY, USA Publisher: Association for Computing Machinery. ISSN: 1539-9087. DOI: 10.1145/3387918. URL: https://doi.org/10.1145/3387918.

A Constructed Queries

A.1 ACM Digital Library

((Abstract: "split inference") OR (Abstract: "distributed inference") OR (Abstract: "collaborative inference") OR (Abstract: "split artificial intelligence") OR (Abstract: "distributed artificial intelligence") OR (Abstract: "collaborative artificial intelligence") OR (Abstract: "split machine learning") OR (Abstract: "distributed machine learning") OR (Abstract: "collaborative machine learning") OR (Abstract: "split neural network") OR (Abstract: "distributed neural network") OR (Abstract: "collaborative neural network")) AND ((Abstract: "tinyml") OR (Abstract: "tiny machine learning") OR (Abstract: "embedded") OR (Abstract: "microcontroller") OR (Abstract: "resource constrain*") OR (Abstract: "edge"))

A.2 IEEE Xplore

("All Metadata":"distributed" NEAR/5 "All Metadata":"inference" OR "All Metadata":"split" NEAR/5 "All Metadata":"inference" OR "All Metadata":"collaborative" NEAR/5 "All Metadata":"inference" OR "All Metadata":"distributed" NEAR/5 "All Metadata":"artificial intelligence" OR "All Metadata":"split" NEAR/5 "All Metadata":"artificial intelligence" OR "All Metadata":"collaborative" NEAR/5 "All Metadata":"artificial intelligence" OR "All Metadata":"distributed" NEAR/5 "All Metadata":"machine learning" OR "All Metadata":"split" NEAR/5 "All Metadata":"machine learning" OR "All Metadata":"collaborative" NEAR/5 "All Metadata":"machine learning" OR "All Metadata":"distributed" NEAR/5 "All Metadata":"neural network" OR "All Metadata":"split" NEAR/5 "All Metadata":"neural network" OR "All Metadata":"collaborative" NEAR/5 "All Metadata":"neural network") AND ("All Metadata":"TinyML" OR "All Metadata":"tiny machine learning" OR "All Metadata":"embedded" OR "All Metadata":"microcontroller" OR "All Metadata":"resource constrain*" OR "All Metadata":"edge")

A.3 Scopus

(ABS(distributed W/5 inference) OR ABS(split W/5 inference) OR ABS(collaborative W/5 inference) OR ABS(distributed W/5 artificial intelligence) OR ABS(split W/5 "artificial intelligence") OR ABS(collaborative W/5 "artificial intelligence") OR ABS(distributed W/5 "machine learning") OR ABS(split W/5 "machine learning") OR ABS(collaborative W/5 "machine learning") OR ABS(distributed W/5 "neural network") OR ABS(split W/5 "neural network") OR ABS(collaborative W/5 "neural network")) AND (ABS("TinyML" OR "tiny machine learning" OR "embedded" OR "microcontroller" OR "resource constrain*" OR "edge"))