POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Dimensions of AI-enabled Mobile and Embedded Devices: an Integration Design Guide

Tesi di Laurea Magistrale in
Computer Science Engineering - Ingegneria Informatica

Author: **Marco Fasanella**

Student ID: 10617541
Advisor: Prof. Cinzia Cappiello
Co-advisors: Prof. Giacomo Boracchi
Academic Year: 2022-23

Abstract

The increasing prevalence of AI-enabled devices is revolutionizing society by harnessing the power of artificial intelligence to enhance a wide range of applications.

However, when dealing with resource-constrained devices like smartphones and embedded systems, limitations and challenges in implementing AI can be faced.
To gain a comprehensive understanding of AI implementation in resource-constrained devices, a deep investigation of each aspect is crucial.
This work seeks to develop strategies and techniques that address the challenges specific to these devices in both their local and offloading execution.

With the correct optimization of these design dimensions, such as resources, timeliness, availability, privacy, and accuracy, the proposed solutions aim to enhance the performance and usability of AI on resource-constrained devices.

This involves exploring techniques like model compression, lightweight algorithms, efficient data representation, and intelligent resource allocation strategies. The goal is to enable resource-constrained devices to leverage AI capabilities effectively without compromising on performance, energy efficiency, or user experience.

The proposed solution aims to empower these devices with intelligent functionalities, enabling them to handle a wide range of applications while operating within their limitations, contributing to the advancement of AI-enabled technologies that can be seamlessly integrated into everyday life. The methodology proposed in this thesis uses a graphical tool to exploit the 5 dimensions of AI and have the best recommendation of the most efficient and practical deployment typology to use.

Keywords: AI, Software Engineering, Design Dimensions, Resource-Constrained Devices

Abstract in lingua italiana

La crescente diffusione di dispositivi con IA integrata sta rivoluzionando la società, sfruttandone le potenzialità per migliorare le sue applicazioni nel settore industriale e informatico. Tuttavia possono presentarsi alcune sfide e limitazioni nell'implementazione efficace dell'IA quando si tratta di dispositivi con risorse limitate, come gli smartphone e i sistemi embedded.

Per identificare le best-practise da attuare quando si tratta di integrare l'Intelligenza Artificiale è fondamentale indagare in modo approfondito ogni suo aspetto.

Attraverso la comprensione della complessità di ciascuna dimensione, questo lavoro cerca di sviluppare strategie e tecniche che affrontino le sfide specifiche di questi dispositivi sia nell'esecuzione in locale che attraverso offloading. Ottimizzando attentamente queste dimensioni, come resources, timeliness, availability, privacy e accuracy, viene proposto come migliorare le prestazioni e la fruibilità dell'IA su dispositivi con risorse limitate.

Ciò comporta l'esplorazione di tecniche come model compression, lightweight algorithms, efficient data representation, e intelligent resource allocation strategies. L'obiettivo è quindi consentire ai dispositivi di sfruttare efficacemente le capacità dell'IA senza comprometterne le prestazioni, l'efficienza energetica o l'esperienza dell'utente.

Pur operando all'interno dei loro limiti, le soluzioni proposte mirano a dotare questi dispositivi di funzionalità intelligenti, consentendo loro di gestire un'ampia gamma di applicazioni. Ciò per contribuire al progresso delle tecnologie con IA che possono essere integrate nella vita di tutti i giorni, dando strumenti efficaci agli utenti e consentendo applicazioni innovative in vari settori.

Parole chiave: IA, Ingegneria del Software, Dimensioni, Dispositivi con Risorse Limitate

Contents

1 | Introduction

The COVID-19 pandemic led to a rise in demand for escapism, entertainment, and mobile experiences, which in turn resulted in unprecedented mobile growth between 2020 and 2022, with a remarkable increase in app installs and record-breaking sales.

The global mobile application market is anticipated to grow at a Compound Annual Growth Rate (CAGR) of 8.6% between 2023 and 2033. The market is anticipated to cross a market share of US$ 170.2 billion by 2033, while it currently holds a revenue of US$ 74.3 billion in 2023 [15].
In 2022 alone, 142.6 billion apps and games were downloaded, showcasing a positive trend since the first quarter of 2015 [24].

This emphasizes the ongoing growth and popularity of the app market. Indeed, these numbers highlight the pervasive nature of the mobile application sector in our daily lives. The increasing usage and reliance on mobile applications have become an integral part of our routines, shaping how we communicate, work, entertain ourselves, and access information.

Mobile applications have seamlessly integrated into various aspects of our lives, providing convenience, efficiency, and entertainment at our fingertips.
From social networking, messaging, and productivity apps to gaming, streaming, and e-commerce platforms, these applications have become indispensable tools that accompany us throughout our daily activities.
The continuous growth of the mobile application sector reflects its significance and impact on society. As the market expands and new technologies emerge, we can expect further innovation and integration of mobile applications into various domains of our lives. The sector's ability to adapt and evolve to new user demands underscores its importance and its ability to enhance our daily experiences.

For this reason, this thesis aims to establish a set of guidelines to facilitate the engineering process of MDLA, and give a complete outlook of such a wide field like AI.

The tool proposed, exploiting all 5 dimensions of AI on resource-constrained devices through a graphical representation, will guide the user to a better understanding of the most effective deployment strategy to use, highlighting the strengths and weaknesses of the model concerning a real-world scenario.

The thesis is structured as follows: the content of Chapter 2 will describe the problem and its formalization. Then in Chapter 3, the current related work, frameworks and methodologies will be analyzed. Once described the problem and the actual state of art solutions and limitations, in Chapter 4 the methodology dimensions will be deeply investigated, to differentiate the three design solutions proposed in this thesis. In the end, in Chapter 5 a real use case for a banking app will be implemented with the proposed methodology, leading to final considerations in Chapter 6.

2 | Problem Formulation

Nowadays, the current limitations of mobile and embedded devices ask for a solid methodology to pragmatically implement robust and efficient models. An initial approach consists of a deep dive into the most important aspects that AI holds, considered in this work as dimensions, and then gives a complete insight into the drawbacks of the current state of the art, proposing new solutions that leverage the various framework and technologies. To overcome these problems, this work proposes a graphical tool to practically visualize and understand the dimensions of AI, giving an easy-to-interpret solution to their implementation on resource-constrained devices.

Let D be the set of dimensions, where $D = \{d_1, d_2, d_3, d_4, d_5\}$, and each dimension d_i represents an aspect of AI integration design.
Let us denote the dimensions:

- d_1: Resources
- d_2: Availability
- d_3: Accuracy
- d_4: Timeliness
- d_5: Privacy

These five dimensions represent a Deep Learning application within a resource-constrained device. They encompass all the necessary aspects and requirements to describe the limitations and the strength associated with each specific aspect of it.
We can plot our representation of a sample point:

$$p = (d_1, d_2, d_3, d_4, d_5)$$

in the D-Multidimensional space using a radar chart, where each d_i is a score, from 0 to 10, assigned by the architectural designer to enhance each dimension from the point of view of what is necessary to the model to fulfill all their purposes.

Radar Chart A radar chart shows multivariate data of three or more quantitative variables mapped onto an axis [53]: it is mostly used to display the strengths and weaknesses aspects of a certain application field.
The radar chart consists of a sequence of equiangular spokes, called radii, with each spoke representing one of the variables. The data length of a spoke is proportional to the magnitude of the variable for the data point relative to the maximum magnitude of the variable across all data points. A line is then drawn, connecting the data values for each spoke.

The application of the 5 dimensions to this particular type of chart is possible since all the dimensions are orthogonal to each other and so show no dependencies.
Nevertheless, it is important to underline that this is a pure abstraction: in a real-world scenario, we will always face the intrinsic relationship that some of these dimensions have.

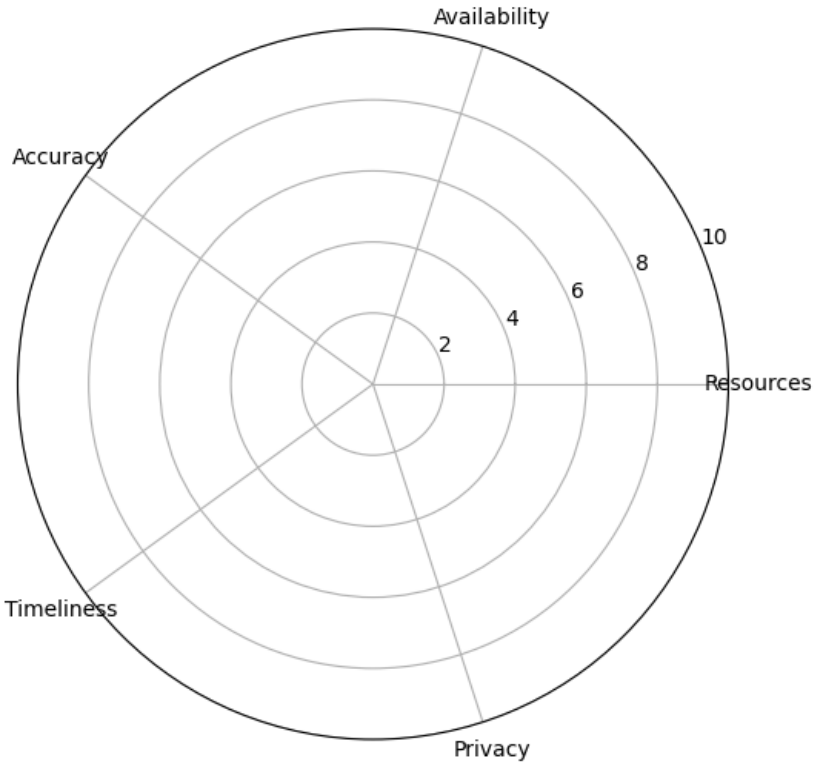

Figure 2.1: Radar Chart representation

The order of representation is aimed to bring together correlated dimensions to achieve a clear visual distinction.
Furthermore, an effort has been made to position dimensions with conflicting real-world applications in diametrically opposite positions. Examples of this include timeliness and resources, or accuracy and resources. In the following chapters, it will be shown more clearly why these dimensions can compete with each other.

In order to exploit the graph to be a useful tool to have a hint on which kind of design solution should be used, we can represent its center of gravity (centroid).

Centroid The centroid is the center of mass of a two-dimensional planar lamina [62]. A lamina is a two-dimensional planar closed surface L which has a mass M and a surface density $\sigma(x, y)$ (in units of mass per area squared) such that:

$$M = \int_L \sigma(x, y) dx dy$$

The centroid of a lamina is the point on which it would balance when placed on a needle. The centroid of a solid is the point on which the solid would "balance". The resulting point will be:

$$\overline{x} = \frac{\int\int x\sigma(x,y)dA}{M}$$

$$\overline{y} = \frac{\int\int y\sigma(x,y)dA}{M}$$

where A is the polygon area.

While it may not hold mathematical significance in this particular application, its only purpose is to provide a graphical understanding of the dominant dimensions and recommend the best implementation to use, depending on the resulting centroid point.

A representation of an example point:

$$p = (6, 6, 6, 6, 6)$$

is shown in Figure 2.2-2.3. The resulting polygon is highlighted in blue, while its corresponding centroid is highlighted in green.

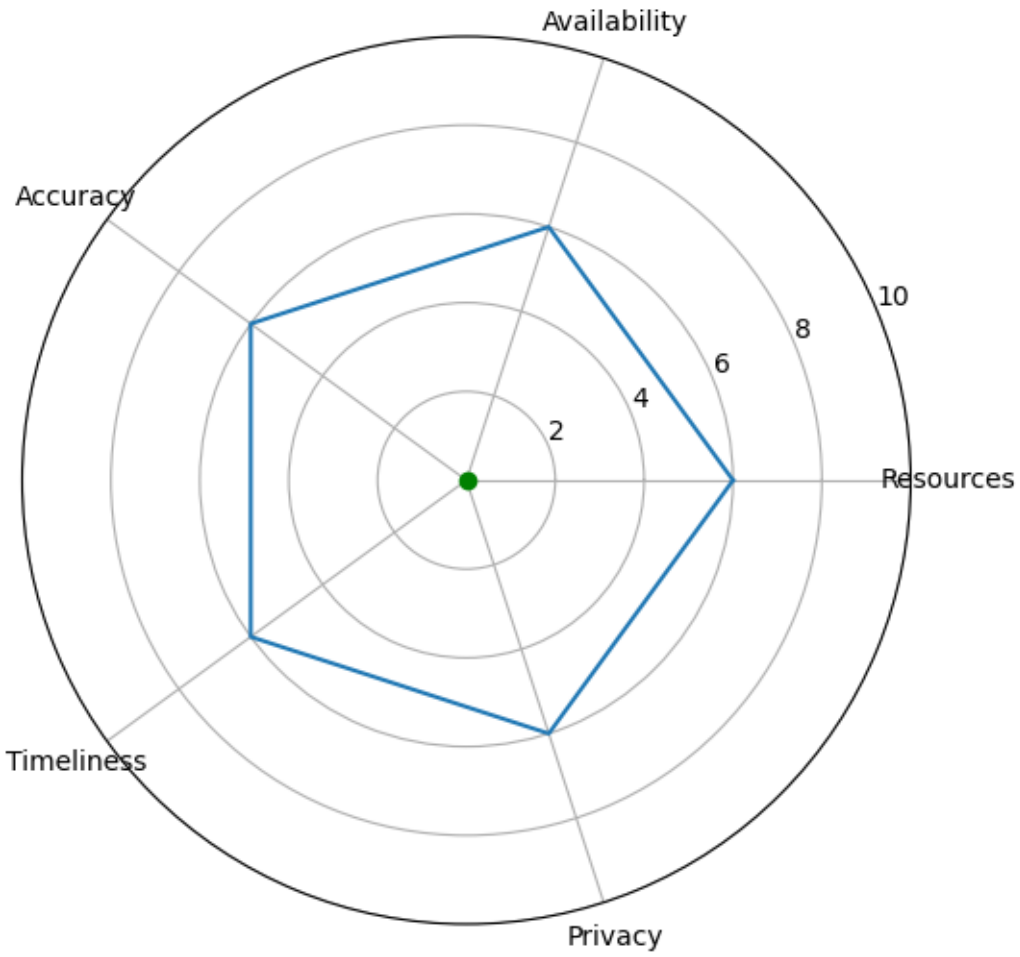

Figure 2.2: Sample point Radar Chart representation

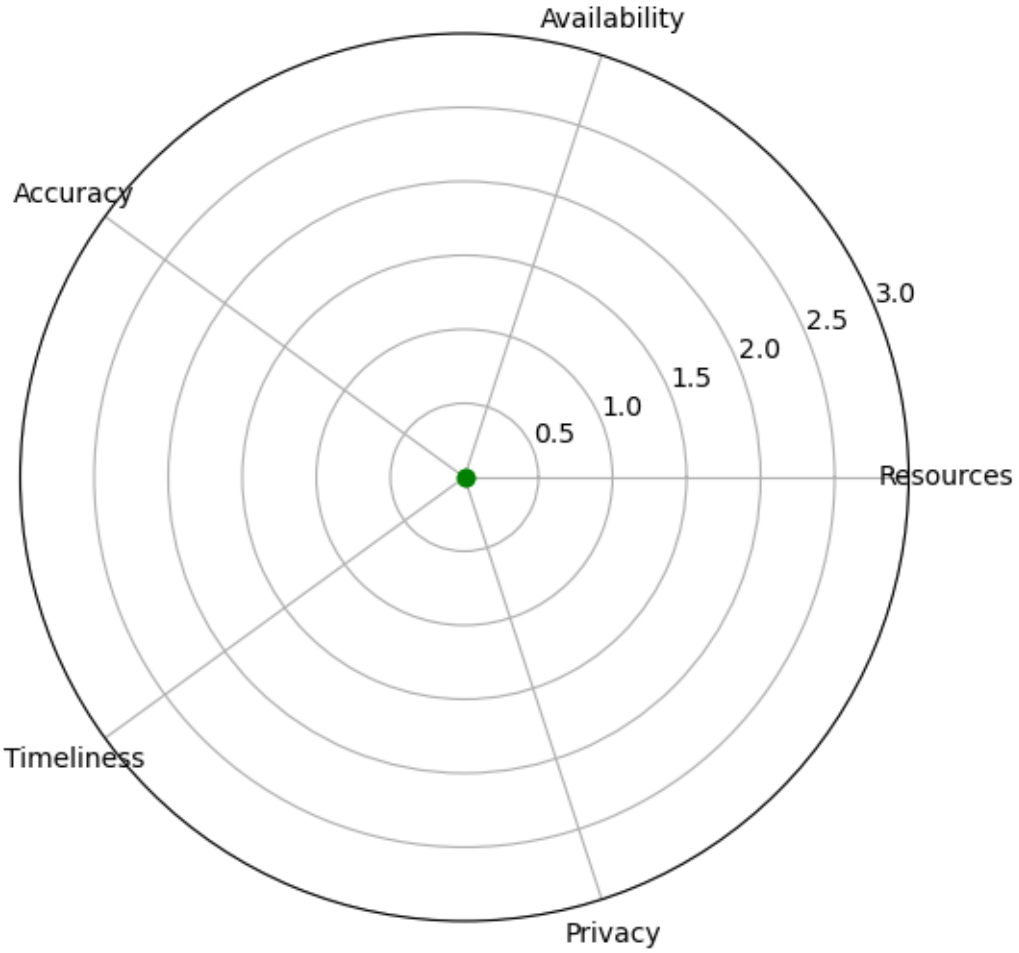

Figure 2.3: Centroid zoomed representation

3 | Related Work

Mobile Deep Learning Applications (MDLAs) have witnessed a significant growth in recent years and garnered considerable attention owing to their tangible advantages for diverse user segments.
User-oriented MDLAs are primarily designed to offer personalized and diversified intelligent services to mobile users, with the user being the primary beneficiary. These MDLAs leverage lightweight and portable data collection devices, such as cameras, microphones, and sensors integrated into mobile devices, to gather data that can be processed on nearby mobile devices or edge servers. Two notable applications are sensor data processing and computer vision.

MDLAs employ lightweight deep learning techniques that are portable and readily accessible to users. The design and development of such MDLAs are centered around the user, making them inherently user-centric, and will be the object of discussion in the next sections of this paper.

Notably, MDLAs offer several benefits, such as facilitating user verification [30, 71], enabling mobile visual tasks [12, 45], monitoring human activities [25], tracking medical health indicators [69, 70] and many other applications.
There is a current research trend focused on identifying the strengths and weaknesses of MDLA that continuously proposes methodologies and solutions.
Despite the efforts put forth, some limitations in MDLA cannot be mitigated, not due to a lack of ideas or solutions, but rather due to inherent physical constraints imposed by the limited resource capabilities of current mobile devices and IoT technologies.
In addition, a lack of coherence among the various proposed solutions and a unified perspective of the MDLA domain remains.

This thesis provides an initial overview of the critical dimensions to take into consideration, and then the most effective techniques and solutions to implement based on the

constraints of the specific application scenario.
Although benchmarks and previous work exist to assess models or frameworks across various devices [13], they must be evaluated within the context of usage and the novelty of the application scenario.
Effective improvement in the usage of Deep Learning (DL) on mobile devices in real-world applications can only be achieved through a conscious and accurate implementation and combination of various techniques.

Research efforts in this domain are primarily focused on developing various techniques that enable reducing the size and complexity of models while simultaneously improving their response time.
Additionally, research efforts are also aimed at devising effective strategies for deploying such models, with a particular focus on systematization and networking.
In this regard, research is currently progressing in two main directions: local optimization for running models efficiently on local devices and distributed deployment for effectively deploying models across a network of devices such as remote clouds and edge networks [32].

MEC With the continuous development of Mobile Edge Computing (MEC) and 5G, the Ultra-Reliable Low-Latency Communications (URLLC) scenarios defined in these systems provide extremely low delays and high reliability.
Unlike the remote public clouds in conventional cloud computing systems such as Amazon Web Services and Microsoft Azure, MEC enhances radio access networks (RANs), which is close to mobile users, with computing capability [36]. To achieve higher energy efficiency or better computation experience, computation offloading strategies for MEC have been widely investigated recently. For short-term optimization over quasi-static channels, some algorithms have been studied in [5, 8, 65].

Considerable progress has been made in the field of MDLA, yet there remains a vast scope for further exploration.
In order to have a comprehensive perspective of the facets of such a vast research field, in this thesis it was considered grouping the main aspects of MDLA into dimensions such as privacy, timeliness, accuracy, resources, and availability.
By considering these dimensions, developers can select design patterns and technologies that align with the application's objectives.

Privacy Privacy is a crucial consideration for MDLAs, as user data is often processed by these applications on the cloud. The privacy dimension is a crucial factor to consider when deciding between local computation and offloading computation.
The choice of approach must be made while taking into account the sensitivity of the input data and the potential risks associated with transferring or sharing that data across networks [55, 58, 66].
Various techniques have been developed to obfuscate or hinder the reconstruction of input data that is transmitted during offloading [21, 41]. These techniques are aimed at enhancing the privacy and security of data that is transmitted between mobile devices and remote cloud servers.
Thus, privacy-enhancing techniques are employed to ensure user trust and compliance with privacy regulations.

Timeliness Timeliness is another important dimension, as many MDLAs require real-time or near-real-time response. Therefore, performance optimization and latency minimization are necessary. Research in the field of deep learning has identified several techniques to mitigate the time complexity of deep neural networks, including low-rank tensor decomposition [50], computation acceleration [44], knowledge distillation [20] and data transformation or reuse [18, 23]. These techniques focus on reducing the computational cost of deep neural network models, particularly those deployed on resource-constrained mobile devices.

Accuracy Accuracy is critical in applications such as medical diagnosis or predictive maintenance, where incorrect predictions can have serious consequences. Developers must evaluate the model's accuracy and optimize it accordingly.

Resources and Availability Resources and Availability are often limited by hardware and network capabilities, requiring techniques such as model pruning [2, 12, 38], parameter sharing [9, 17], and network quantization [68].

In summary, the structured approach to MDLA development enables developers to analyze and evaluate several dimensions that affect the application's design and deployment.
By considering these dimensions, developers can create effective and responsible MDLAs that prioritize user privacy, performance, and ethical principles.

4 | Methodology

In this section, the dimensions of the proposed methodology will be deeply investigated to have a concise view of each of them and how to exploit them.
The detailed comparisons and considerations will then lead to the intrinsic relationship that each dimension may have with another.

4.1. Dimensions

The present time necessitates the establishment of fundamental principles for deploying DL applications in a pragmatic manner. Such principles would help in avoiding trivial yet consequential errors that can harm both the performance of the mobile device, which is intrinsically fragile, and the functionality of the deployed application, leading to disruption in the end-user experience.

To represent the problem of developing Mobile Deep Learning Applications in a structured manner, a systematic approach involving the use of various dimensions has been adopted in this thesis. This approach employs metrics to quantify and comprehend the strengths and limitations of the application under development, enabling efficient planning and production.
By using this approach, the challenges of MDLA development can be systematically analyzed, leading to a more comprehensive understanding of the problem and facilitating the identification of effective solutions.

4.1.1. Resources

The Resources dimension considers the computational power and essential hardware characteristics available to the model for performing inference. It encompasses various components and capabilities that contribute to the model's computational efficiency and overall performance. Some key resources considered within this dimension include CPU (Central Processing Unit), GPU (Graphics Processing Unit), RAM (Random Access Memory), Battery and additional hardware characteristics such as storage capacity, network connectivity options (e.g., Wi-Fi, cellular), specialized accelerators (e.g., ASICs, FPGAs), or dedicated AI processors (e.g., TPUs).
It emphasizes the importance of considering the available computational power, memory, and other hardware attributes when designing and deploying models.

Understanding the capabilities and limitations of the underlying hardware infrastructure enables efficient utilization of resources and optimization of model performance. Techniques like model compression, hardware-specific optimizations, or selecting appropriate algorithms can be employed to make the best use of available resources and improve inference efficiency.

Capacity When it comes to integrating a machine learning model in a mobile app, we need to enclose in the app bundle a model (i.e., tflite in the case of a TensorFlow model), which usually results in several dozens of MB, implying that sometimes just integrating a medium-large DNN model in an existing application can drastically increase its size.

Unfortunately, it is a solution that, at the present state of the art, is not still affordable in terms of storage capacity: according to a pool [6] from over 4'000 Android users, it resulted that they use on average between 64GB and 128GB of storage.
Nevertheless, the benchmark storage capacity for Android and other non-iOS devices is steadily increasing: according to Counterpoint's latest report on smartphone memory, the average smartphone NAND flash capacity crossed 100GB in 2020 for the first time. However, it differs in iOS and Android phones.
In iOS phones, the average NAND capacity reached 140.9GB in Q4 2020, compared to 95.7GB in Android phones during the same period. The proportion of devices with at least 128GB storage grew to just under 40% in Q4 2020 compared to 24.9% a year prior. Devices with more than 256GB of storage saw a demand of around 9.3% in Q4 2020, slightly lower than a year before [56].
Despite the advancements in mobile device technology, not everyone has access to the

latest and most powerful models, leading to a significant variance in the storage, CPU, and battery capacity.

In the end, considering the great storage space needed by multimedia files, pushed forward with the incredible growth of camera resolution and quality, and space allocated from apps themself to download additional features bundles or assets, there remains a little wiggle room to increase the size of apps. It is crucial to consider this dimension when developing MDLA, as it represents a significant limitation that must be taken into account during the application design process.

A further challenge arises from the need for modern deep learning systems to process vast amounts of data during model training. For instance, the size of the ImageNet 2012 dataset [11] used for training (and inference) can reach up to 138 GB respectively, which exceeds the storage capacity of most commercially available off-the-shelf (COTS) mobile devices, as seen in previous considerations.
As a consequence, to access high-quality deep learning services, users are required to upload their data, such as images and motion data, to the cloud for collaborative computing.

The Resource dimension is indeed correlated with both timeliness and availability, and different components within this dimension play a role in ensuring the desired performance and operational characteristics of the model.

Motherboard Regarding timeliness, components such as GPU, CPU, and RAM are crucial for achieving fast response times in local deployments. The computational power of the GPU and CPU, along with the available memory in RAM, directly impact the speed at which the model's computations can be performed.
Adequate resources in these areas enable efficient processing of data and calculations, reducing inference latency and ensuring timely responses.

The mean inference latency may not always accurately reflect the actual performance experienced on the hardware side, especially in the context of CPU inference, where thermal protection mechanisms like Dynamic Voltage Frequency Scaling (DVFS) or CPU throttling are in place.
Thermal protection mechanisms are implemented in CPUs to prevent overheating and ensure the long-term reliability of the device. These mechanisms dynamically adjust the CPU's voltage and frequency based on its temperature. When the CPU temperature

reaches a certain threshold, the frequency and voltage are scaled down to reduce heat generation and maintain a safe operating temperature.
As a result, during intensive computational tasks such as AI inference, the CPU performance may be temporarily reduced due to thermal throttling. This can lead to increased inference latency and potentially affect the overall performance of AI applications running on resource-constrained devices.

Lifetime Regarding availability, components like battery status become vital in local deployments. The battery plays a critical role in powering the device, and its capacity determines the runtime and availability of the model's services. Low battery status can limit the device's functionality or even cause the device to become unavailable for inference tasks.
Monitoring the battery status and managing resource usage effectively are important considerations to maintain availability and ensure uninterrupted operation of the model.

4.1.2. Availability

Availability is a measure of the reliability and readiness of a service to fulfill its purpose whenever it is needed [40].
In the context of a model or application, availability refers to the ability of the service to be accessed and utilized by users or other systems.

In the case of local implementation on a device, availability is closely related to the device's battery status. The availability of the model's service depends on the remaining battery life of the device. If the battery is depleted or low, the service may not be available or may experience limitations in terms of processing power or functionality.
Monitoring the device's battery status and managing resource usage efficiently can help maintain availability. This may involve optimizing power consumption, scheduling computations during periods of higher battery levels, or implementing power-saving strategies.

When it comes to offloading, availability is influenced by connectivity and mobility constraints. The availability of the service relies on having a stable and reliable network connection. If the network connectivity is weak, intermittent, or unavailable, it can affect the availability of the offloading service.
Additionally, if the device is in motion or frequently changing network environments (e.g., moving between Wi-Fi networks or switching between cellular towers), it can introduce

challenges to maintain a continuous and reliable connection, thus impacting the availability of the offloaded service.
Ensuring network connectivity and addressing mobility constraints are essential to maintain the availability of the service. Techniques such as network handoff, adaptive offloading decisions based on network conditions, or utilizing multi-access edge computing (MEC) infrastructure can help enhance availability in offloading scenarios.

The goal of ensuring availability is to maximize the service's accessibility and operational continuity, allowing users to utilize the service whenever required.
High availability is particularly crucial for critical services, such as healthcare systems, financial transactions, and real-time applications, where downtime or unavailability can have significant consequences.

4.1.3. Accuracy

Improving model accuracy often involves scaling up neural networks, resulting in deep learning models that require significant computation and storage resources. This can prove challenging for mobile and embedded devices, particularly during the inference phase. As a result, it is imperative to develop techniques for reducing the size of the model and, at the same time, keeping accuracy as high as possible.

However, reducing model size can come at the cost of accuracy, leading to a potential conflict between the accuracy and capacity dimensions. For instance, the Quantization technique involves reducing the number of bits used for parameter representation, which may cause a significant decrease in accuracy. On the other side, the vast number of parameters of deep learning models are quite redundant, making it possible to compress and accelerate deep learning models without significantly degrading their accuracy.

LeCun et al. [29] were the first to demonstrate that certain unimportant weights in pre-trained networks can be removed without impacting accuracy.
Suyog et al. [16] further strengthened this thesis by demonstrating that deep network weights can be represented using 16-bit fixed-point numbers without significantly reducing classification accuracy.

4.1.4. Timeliness

Timeliness concerns the response time of the model and is related to Resource dimension when the inference computation is performed locally and Availability Dimension when we are offloading.

When the inference computation is performed locally on the device, the timeliness of the model's response depends on the available computational resources, such as the CPU, GPU, or dedicated accelerators. Insufficient resources or high computational demands can increase inference latency, resulting in a slower response time.

On the other hand, when offloading computations to a remote server or cloud infrastructure, the timeliness of the model's response is influenced by the availability of the network and the performance of the server. The response time depends on factors such as network latency, bandwidth, and the processing capabilities of the server. If the network connection is slow or unreliable, it can introduce additional latency and impact the timeliness of the model's response.

In both cases, timeliness is a crucial dimension to consider during the design and deployment of the model. Applications with strict real-time or near-real-time requirements, such as video analytics or autonomous systems, demand low-latency responses.
Therefore, ensuring a timely response is vital to meet user expectations and maintain the application's functionality and usability.

The Resource dimension, when considering local computation, involves provisioning sufficient computational resources to minimize inference latency and achieve the desired response time.
This may include optimizing algorithms, utilizing hardware accelerators, or employing efficient model architectures to reduce the computational burden.
Similarly, in the offloading scenario, it involves designing a robust network infrastructure and leveraging cloud resources with high availability and low latency to meet the response time requirements.

4.1.5. Privacy

Studies have revealed that utilizing an individual's date of birth, zip code, and gender, a linkage attack can be performed to infer sensitive attribute values with a nearly 60% success rate [46].

Such attributes have the potential to uniquely identify a significant proportion of the population. By aggregating and analyzing multiple data sources, including those that have been disseminated or disclosed to third parties, a complete dataset can be obtained for the purpose of performing linkage attacks.

Selecting appropriate anonymization techniques for a given dataset necessitates an assessment of both the anticipated utility and the risk of de-identification. This involves carefully balancing the potential benefits and drawbacks of various anonymization approaches, taking into account the intended use cases and the likelihood of re-identification attacks.

In the context of releasing anonymized data, the foremost concern is to obviate the disclosure of sensitive information about individuals. Such disclosure can occur in three distinct forms:

- Identity disclosure: occurs when a guessable algorithm and insufficient anonymization are used, allowing reidentification by linking a specific record in the anonymized data [26]
- Attribute disclosure: appears when new data about an individual is exposed.
- Inference disclosure: may occur if the adversary obtains confidential information about an individual by correlating with different datasets

Generalization Generalization is a data anonymization technique that involves substituting a more general but semantically equivalent value instead of the original value.
This approach is typically applied at the cell level, where certain data values are retained but with added obfuscation to impede attackers from inferring sensitive information.
It should be noted, however, that generalization cannot be uniformly applied across all attributes, particularly those such as ID, name, and postcode, which carry significant identifying power.

Suppression Suppression is a data anonymization technique that removes an entire portion of data from a dataset by replacing its values with a non-meaningful symbol such as "****" [46].
This approach is particularly useful for protecting sensitive attributes such as name and ID number, which threaten individual privacy.
However, suppressing other attributes such as address, postcode, and usage may not be appropriate, as it can negatively impact the utility and usability of the data.

It is important to note that the use of anonymization techniques such as suppression and replacement can result in a significant loss of information. Replaced or deleted values may not only limit the utility of the dataset for its intended purposes but also impact downstream analyses and applications that rely on the availability of complete data.

Distortion Distortion is a data anonymization technique that involves modifying the original data to an alternate form that can be reverted back to its original state using the appropriate decryption method.
One such example is hashing, where the original value is encrypted and compared to a hashed value stored in a database to identify a match. It is important to note, however, that certain methods, such as dictionary attacks and brute force techniques, can potentially compromise the security of the hashed value and reveal the original data.

Swapping Swapping is an anonymization technique that involves randomly rearranging variables within each column of a dataset. It should be noted that this approach is not suitable for all attributes, as it may introduce inaccuracies that could impact research outcomes.
Moreover, a key consideration when using this technique is the risk of obtaining the same value as the original data due to the randomization process.

Masking Masking is a data anonymization technique that replaces selected characters in an attribute with a different character, rendering the variable incomprehensible.
However, a drawback of this technique is that it can be computationally intensive, requiring significant resources for both checking and changing the masked values. Furthermore, the resulting data may be rendered useless for research purposes due to the loss of the original attribute values.

4.1.6. Relations among dimensions

The intentional ordering of the dimensions in the star graph, with Resources, Availability, Accuracy, Timeliness, and Privacy positioned in a counterclockwise sequence, highlights the relationships and alignments between these dimensions.
By placing concording dimensions near each other and placing discording dimensions in diametric opposition, the graph aims to emphasize their interconnectedness and the trade-offs that may exist between them.

Privacy, positioned diametrically opposite to Accuracy, signifies the potential trade-offs between these dimensions. Privacy concerns may necessitate certain measures, such as data anonymization or encryption, which could impact the accuracy of AI systems. Finding the right balance between privacy protection and maintaining accuracy is a crucial challenge.
Furthermore, also Resources is opposite to Accuracy since the ability to generalize of the model is directly correlated with its size: models with a high capability of generalization and the ability to handle diverse data may require more computational resources, such as larger memory, powerful CPUs, or GPUs, and increased storage capacity. These resource-hungry models are often more complex and have a higher capacity to generalize, allowing them to handle a wider range of inputs and produce more accurate outputs.
This dilemma is largely due to the heterogeneous nature of mobile device configurations: while high-end smartphones may offer top-level specifications, affordable Android devices often come with much lower capabilities.
As a result, there is a fundamental trade-off between accuracy and resource requirements in this domain.

The relationship between timeliness and privacy, as well as on the counterpart side of the chart with resources and availability, will be thoroughly investigated in the following sections. These dimensions play a crucial role in determining the appropriate design solutions.

4.2. Design Solutions

4.2.1. Local deployment

The first design to be considered is also the most controversial and difficult to obtain. Due to the heterogeneity and limitations by-design of devices, deploying AI as a local-only solution is nowadays really challenging.

On the other hand, it is essential to investigate such solutions since network connection or offloading availability is not always guaranteed.

Issues and Limitations

The issues of DNN From the perspective of underlying principles, the computational predicament of DNNs is due to the following reasons:

- Memory overhead: An oversized network is a common approach to achieving low generalization error. It is important to note that a large capacity does not necessarily guarantee low generalization error. However, a large hypothesis space raises the upper bound of the generalization ability, thus increasing the possibility of achieving a low error.
- Curse of dimensionality: The high dimensionality of data exacerbates the consumption of computational resources required by deep neural networks (DNNs). These networks often require a large amount of training data to ensure the generalization ability of the trained model.
  If A is the number of required training data points in a one-dimensional sample space, then the number of training data points needed in an n-dimensional sample space is A^n [28].
- Time and Energy overhead: backpropagation is an iterative algorithm that minimizes the training error in deep learning networks. The gradient is computed concerning the weights and other parameters, but the large number of parameters in deep networks results in slow convergence. A single deep learning network can have more than a million parameters [48], and processing a single input can require billions of multiply-accumulate (MAC) operations [19].

The capacity limitation Unfortunately, this kind of approach suffers from a series of drawbacks. Firstly, the size of the app combined with the DNN model can suffer from drastic enlargement, which is not seen as a good property during the software engineering

of an application. We should notice that the average Android app file size is 11.5MB, and the average iOS app file size is 34.3MB [7]. Moreover, this trend will tend to grow in the following years.
To relieve this problem, in the next section, several techniques as model pruning or model distillation will be investigated to reduce the impact that the implementation of such models will have on the devices.

Secondly, due to the intricate nature of existing deep learning systems, such as TensorFlow, which can comprise in excess of 650,000 lines of complex code and algorithms, significant engineering effort is required to develop an incrementally executable design that is both compatible and efficient for model inference within these systems.
However, 90% of smartphones are based on the same Arm Intellectual Property (IP) architectures [3], and this can allow researchers to focus on hardware and software low-level optimization shared across devices.

In deep learning, various factors can induce time or energy overhead, such as backpropagation, memory operations, and hyperparameter tuning.
To improve time efficiency, the redundant computation can be eliminated through algorithmic optimizations, especially in matrix-matrix or matrix-vector multiplications.
Additionally, time can be saved by reusing intermediate results of convolution, parallelizing computation on digital processors, and fine-tuning code on digital processors.

However, memory operations are difficult to optimize due to the von Neumann architecture of traditional digital computers, which have independent processing and memory units. To address this, techniques such as Boolean logic minimization can be used to reduce memory operations and associated energy consumption.

Solutions

Model compression and acceleration offer effective techniques to decrease the storage and computational cost of deep neural network (DNN) models. There are many efforts to make deep learning locally applicable and faster on COTS mobile devices such as DeepX [27], DeepMon [39] and Paddle [43] or Tensorflow Lite [52].
Compressing large DNN models can substantially alleviate the resource and energy demands of inference on mobile devices.

An approach made by Wu et al. [63] tries to break down a high quality deep learning model (original neural network unit) into a small size of input-to- execute code blocks, which are then executed on system-on-chip (SoC) in sequence. This implementation of a platform is called DeepShark and has shown results of using less than 300 MB RAM and achieving an average 70% of memory consumption reduction for one-time trial image recognition. All this is also achieved without accuracy loss.
DeepShark is designed to be transparent to deep learning developers and achieves this by overloading default system functions of TensorFlow and Caffe. This allows developers to invoke DeepShark APIs in the same manner as they would call TensorFlow or Caffe APIs.

However, generally, these approaches might downgrade algorithm accuracy with, on average a loss of 10 percent, and even up to a loss of 15 percent [67]. Presently, three primary solutions are employed for DNN model compression:

- parameter pruning and sharing
- low-rank factorization
- model distillation

Parameter Pruning The rationale behind parameter pruning and sharing is based on the understanding that numerous parameters in a network are not vital for the model's performance.
There are three techniques commonly used for parameter pruning:

- Network quantization: involves compressing the DNN by reducing the number of bits required to represent the parameters in the network. This reduction in the precision of parameter representation helps in reducing storage and computational requirements.
- Weight and connection pruning: aims to eliminate redundant weights in the DNN model. By identifying and removing these unnecessary weights, the overall model

complexity and computational cost can be significantly reduced without compromising performance.

- Structural Matrix: represents an $m \cdot n$ matrix, using a structured matrix that requires significantly fewer parameters than the original mn matrix. This approach allows for efficient representation and compression of the network parameters [1].

These techniques collectively contribute to model compression, enabling reduced storage requirements and computational costs while maintaining acceptable performance levels.

Low-rank Factorization Low-rank factorization is a powerful technique for reducing convolution layers, where kernels are conceptualized as 4D tensors.
The inherent structure of 4D tensors often exhibits a significant degree of redundancy, which can be effectively eliminated.
Notably, this method is not limited solely to convolution layers; it can also be employed with great success in the context of fully-connected layers, which can be represented as 2D matrices.
Consequently, by leveraging low-rank factorization, one can achieve substantial gains in technical efficiency and mitigate issues related to redundant information within these layers.

Model Distillation Model distillation is an effective technique utilized to compress deep neural networks (DNNs) into shallower architectures while preserving their performance characteristics.
This compression is achieved by training a smaller, simplified model, known as the student model, to mimic the behaviour and predictions of a larger, more complex model, known as the teacher model. Through this process, knowledge is transferred from the teacher model to the student model, enabling the student model to capture the essence of the teacher model's decision-making process.

By distilling the knowledge from the teacher model, the student model can effectively learn from the teacher's insights, enabling it to achieve comparable performance while having a significantly reduced network depth. This compression in architecture size offers numerous benefits, including reduced computational requirements, lower memory footprint, and improved efficiency in deployment on resource-constrained devices.

In essence, model distillation facilitates knowledge transfer from a large, intricate teacher model to a compact student model, enabling the student model to exhibit similar performance characteristics while being more lightweight and computationally efficient.

Considerations

As described in this chapter, the two dominant dimensions of local implementation are Privacy and Timeliness. Further considerations of the remaining dimensions then, are necessary during the architecture design of the application.

This implies that our centroid in the radar graph will be polarized on the bottom-left part of the chart, where these two dimensions are dominant (as shown in Figure 4.1).
In Table 4.1, there's a summary of the non-dominant dimensions:

Dimension	Pain	Relief
Resources	Nowadays devices have low memory computational power and lifetime	The trend in technology research is continuously pushing up the barriers and limitations in the devices architectures
Accuracy	Model with high accuracy and low generalization error are heavy	Techniques such as Model Pruning can help decrease the model size
Availability	Lifetime (battery capacity) of such devices is limited when making resource-hungry computations such as SGD	Using ad-hoc hardware and exploiting techniques such as Low-rank Factorization or Model Distillation can significantly extend the lifetime span of the device

Table 4.1: Considerations about Local Deployment non-dominant dimensions

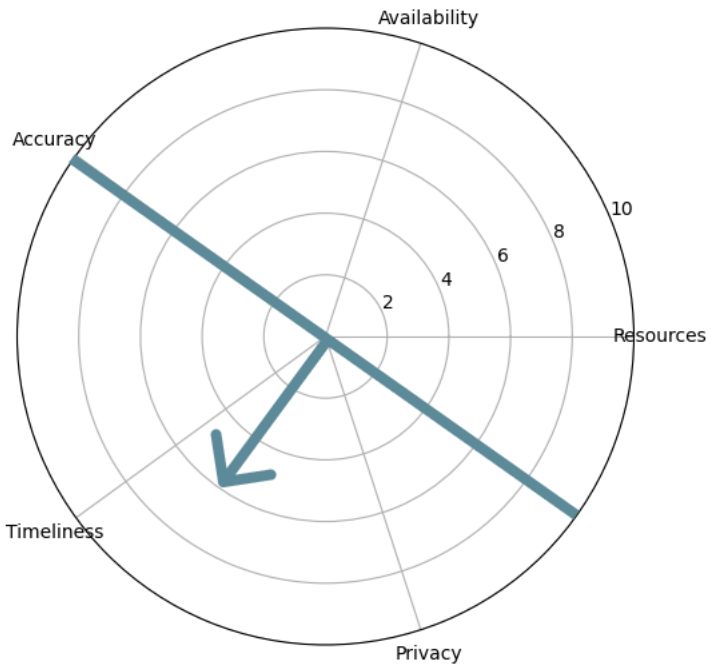

Figure 4.1: Visualization of the polarized part of Local Deployment in the Radar Chart

4.2.2. Distributed deployment

The integration of cloud-based technologies and automation with deep learning in Mobile Edge Computing (MEC) is anticipated to enhance resource utilization and efficiency, augment system resilience, optimize power consumption, boost revenues, and simplify operational procedures for service providers.
MEC combines the functionalities of mobile computing and edge computing, complementing the edge computing infrastructure with the resources of mobile or IoT devices that have low-consumption computing and storage hardware limitations.

Mobile computing (MC) refers to a network environment that is decentralized and distributed, comprising mobile devices, IoT devices, and other elements that share computing, storage, and networking resources. Edge computing (EC) is a system that provides computing, networking, and storage services near end devices, typically located at the network's edge.
This makes it a highly attractive solution for supporting mobile and the Internet of Things (IoT) applications that require low latency. Moreover, this scheme improves the quality-of-service (QoS) of mobile network operators and the quality-of-experience (QoE) of UEs.

The concept of MEC, aims to alleviate the burden on backbone networks by bringing the computation and storage resources closer to the user equipment (UE). By doing so, MEC effectively bypasses the long propagation delays associated with transmitting data from mobile devices to remote cloud computing infrastructures.

Furthermore, it brings cloud-based storage, computation, measurement, and management closer to the end-user, enabling the edge network to optimize resource utilization, ensure QoE, and generate revenue for network operators by integrating the internet communication infrastructure with the cloud.
One of the major advantages of MEC is that it enables computation to take place on the edge of the network, reducing the load on centralized cloud resources, which is particularly crucial in low-latency applications where real-time processing is essential.

Issues and Limitations

Computation offloading, although beneficial, is not without its challenges. The process involves wireless data transmission, which can lead to congestion on wireless channels, thus creating the need for decision-making or optimization to address the entire communication and computation integrated system. This means there is a need to jointly allocate communication resources and computation resources of edge nodes.
According to Wang et al. [59], optimization methods used in MEC may encounter several challenges specific to the use cases they are applied in.
These challenges include:

- Uncertain Inputs: optimization methods rely on certain key information factors as inputs, but obtaining them can be challenging due to factors such as variant wireless channels and privacy policies.
- Dynamic Conditions: The integrated communication and computation system in MEC is subject to constant change, and the optimization methods may not effectively address these dynamics.
- Temporal Isolation: Many optimization methods used in MEC do not consider the long-term effects of current decisions on resource allocation.

Other specific aspects to take into consideration will be discussed in the following paragraphs.
In summary, these challenges may limit the effectiveness of optimization methods in MEC and highlight the need for continued research to develop more robust and adaptable approaches. On the other hand, a recently emerging trend is the application of Artificial Intelligence techniques to optimize wireless communication. A deeper discussion can be found in Section 4.2.2.

Task Offloading With the proliferation of smartphones, tablets, wearables, and IoT devices, there has been an inflation in communication that can result in congestion of the core network that connects to centralized cloud servers.
This congestion can be intensified during peak streaming periods when there are multiple requests for popular videos, leading to suboptimal quality of experience (QoE) for users and inefficient utilization of network resources.

Inference on the cloud has two serious problems:

- It requires internet access to enable the interaction between the device and the cloud server, which results in a constraint that sometimes cannot be fulfilled due to

the instability of the mobile phone nature since it is in continuous spatial moving and resource constrained and so subject to insatiable connectivity (Availability and Timeliness)

- The input data for the model inference needs to be completely uploaded to the cloud server, resulting in violating the privacy requirement that the application could have. (Privacy)

Indeed, during the design phase of model integration and deployment, it is crucial to thoroughly investigate the dimensions of Availability, Timeliness, and Privacy. Neglecting these aspects can have a detrimental impact on the quality of service (QoS) provided by the deployed model.

Availability refers to the accessibility and reliability of the model. It is essential to ensure that the model is consistently available for users, with minimal downtime or disruptions. Adequate infrastructure, fault-tolerant systems, and appropriate deployment strategies should be considered to maximize availability.

Timeliness refers to the responsiveness and efficiency of the model in delivering results within acceptable timeframes. Depending on the application requirements, real-time or near-real-time performance may be necessary. Latency, throughput, and optimization techniques should be carefully considered to meet the desired timeliness goals.

Privacy, as discussed earlier, relates to the protection of sensitive user data. It is imperative to incorporate privacy-enhancing measures, such as data anonymization, encryption, secure transmission protocols, and user consent mechanisms, to safeguard user privacy and comply with relevant regulations and policies.

By deeply investigating these dimensions during the design phase, potential pitfalls can be identified and addressed proactively. Failure to consider these aspects can result in poor QoS, leading to user dissatisfaction, compromised data security, and unreliable system behaviour.

Mobility It is important to consider that performing offloading on mobile devices can be challenging due to their inherent mobility nature, which can affect the connectivity and so the timeliness dimension. Predicting mobility patterns is a critical step in understanding mobile network demands.

Mobility models can be developed by considering various environments, such as urban [60] or highway patterns, to forecast the next station that a user is likely to connect to [61], thereby reducing costs for operational tasks such as handover [42]. Other studies utilize Long Short-Term Memory (LSTM) to predict the position of the User Equipment (UE) over time [10].

Quality of Service Quality-of-Service (QoS) is another further limitation, and it is usually the most tangible and effective tool we have to measure the user feedback.
Quality of Service is a set of performance indicators that measure the quality of network services experienced by end users. These indicators include bandwidth, latency, and error rate, among others, and changes in these parameters can significantly impact the network's ability to provide critical services.
The data transmission over a network can lead to a significant utilization of available bandwidth. This effect aggravates as the number of users accessing the network increases, resulting in greater traffic loads that can ultimately degrade the QoS of the cloud infrastructure.
Additionally, the latency associated with data transmission between local mobile devices and remote cloud infrastructure may be significant, resulting in delays that can impact the overall user experience [33].

To obviate the problem, each user connects to the network with specific QoS requirements, which may be more stringent for latency-sensitive applications such as on-demand video streaming and voice-over IP (VoIP). This ensures both safety and Quality of Experience (QoE).

Furthermore, QoE can be optimized based on user similarity within groups or geographical regions to dynamically allocate resources according to group needs [64]. Achieving ultra-reliable and low-latency communication (URLLC) is one of the major challenges in 5G networks. The curse of dimensionality can also impact QoS in high-dimensional datasets by making it difficult to identify meaningful patterns and relationships between variables.

Privacy issue Before large-scale ML systems are deployed, network providers must investigate any potential privacy violations that may occur in the collection or use of user data, particularly in the case of smart cities or personal electronics such as connected cars and IoT devices that may reveal sensitive information about the public as a whole.
When working in 5G mobile edge computing, security must be guaranteed due to the

need for sharing physical infrastructure between slices and the potential for information leaks regarding data or usage patterns.

The potential exposure of sensitive content through network uploading is indeed a significant concern and a notable flaw in offloading approaches. When the context involves handling sensitive data such as face photos, documents, or personal information, it becomes crucial to prioritize privacy dimension.
In such cases, favoring a local-only implementation, where the data remains on the user's device, can be a prudent choice. This approach minimizes the risk of data exposure during network transmission.

However, it is worth noting that techniques are available to mitigate privacy risks even when utilizing offloading methods, as described in Section 4.1.5.
Anonymization techniques can be employed to safeguard user data while still allowing for remote processing.
These techniques ensure that any uploaded data is stripped of personally identifiable information or other sensitive details, thereby preserving user privacy.

By adopting appropriate anonymization techniques, it is possible to strike a balance between the advantages of offloading computations and maintaining the privacy of user data. This enables the benefits of remote processing while minimizing the potential risks associated with sensitive content exposure during network transmission.
It is essential to carefully consider the privacy dimension and select the most suitable approach based on the specific requirements and sensitivity of the data involved.

Strength

The Advent of 5G By integrating mobile edge computing with 5G networks, cloud computing can be seamlessly connected with edge computing, thereby facilitating the development of innovative applications. Nonetheless, various challenges need to be addressed before DL can be fully leveraged in the 5G mobile edge.
Many of these challenges are also pertinent to machine learning systems in general.

Model versioning management One advantage of offloading tasks to a cloud server in the context of deep neural network (DNN) models is the ability to update the model without modifying the local apps. By leveraging the cloud server for resource-intensive tasks, the burden of executing computationally demanding operations is shifted away from the local devices.

This flexibility enables rapid iterations and advancements in the model's performance, accuracy, and functionality, without inconveniencing the end-users. With the cloud server handling the resource-hungry tasks, local apps can operate with reduced computational requirements, resulting in improved performance and responsiveness on the user's device. This can lead to a better user experience, especially on resource-constrained devices, as they can offload the heavy lifting to the cloud infrastructure.

Furthermore, offloading resource-intensive tasks to the cloud server can also alleviate the limitations imposed by local device constraints, such as limited processing power, memory, or battery life, and so influencing the Resource Dimension.

AI-supported networks Task offloading systems are designed to respond to changes in real-time demand for computation and supporting resources at mobile edge computing nodes. As the scale of task offloading initiatives grows, and each edge node simultaneously receives and runs computational requests, the scheduling objective becomes almost intractable without the use of machine learning techniques. To address this challenge, intelligent systems have been developed that leverage Deep Q-Networks (DQNs) to schedule Augmented Reality/Virtual Reality (AR/VR) offloading tasks while simultaneously minimizing costs of energy, computation, and delay [22].

These systems also rely on additional DL techniques to reallocate resources at the mobile edge in real-time. In addition to cost minimization, task-offloading schemes can also be trained to minimize delay or maximize users' Quality of Experience/Service (QoE/QoS) [10].

As cited before, a recent emerging trend is the application of Artificial Intelligence (AI) techniques to optimize wireless communication [34, 35]. This includes but is not limited to the use of AI for optimizing the physical layer (PHY), data link layer, and traffic control [51].

Real World Applications

The European 5GPPP has identified various sectors that would benefit from mobile edge computing, including IoT, caching, video streaming, augmented reality, healthcare, and connected vehicles.
As large-scale 5G networks are built, researchers are exploring the many application spaces that could benefit from the low latency, proximity, high bandwidth, location awareness, and real-time insight provided by mobile edge computing.

This growth of mobile edge computing will disrupt the current cloud computing paradigm in favour of localized computing near the user.

By 2030, the number of IoT devices is expected to increase to 29.42 billion [54], all of which will require network connectivity.
Mobile edge computing can provide benefits to both small-scale personal IoT devices and large-scale deployments, such as smart cities and new industrial applications. Small devices, like Amazon Alexa, Nest Cam, and Google Home, can use mobile edge computing to offload computational tasks that are too complex for their limited memory capacity.

Considerations

When considering an offloading implementation of a model, it is essential to consider dimensions such as privacy and timeliness as crucial factors in the decision-making process. The suitability of offloading depends on the specific context and requirements of the application.

If the data being handled is non-sensitive and does not involve any privacy concerns, then offloading can be a viable option. Offloading computational tasks to a remote server can provide benefits such as reducing local resource usage, improving energy efficiency, and leveraging the scalability and computational capabilities of cloud infrastructure.

Similarly, the timeliness dimension is critical in certain applications. If there are no strict requirements for real-time or near-real-time responses, and the latency introduced by offloading does not affect the user experience or application functionality, then an offloading implementation can be considered.

However, it is important to note that even in cases where data is not sensitive and response time constraints are not strict, privacy and timeliness should still be considered as relevant factors. It is good practice to adopt anonymization techniques and ensure reasonable response times to maintain user trust, even when dealing with non-sensitive data.
In Table 4.2, there is a summary of the non-dominant dimensions:

Dimension	Pain	Relief
Timeliness	Timeliness can be a problem due to the intrinsic behaviour of a mobile device	An emerging trend is the application of Artificial Intelligence (AI) techniques to optimize wireless communication.
Privacy	Privacy is put at hard risk during offloading, exposing sensitive user data to the network	Techniques such as Generalization, Suppression or Distortion can be exploited to hide user information and preserve privacy

Table 4.2: Considerations about Distributed Deployment non-dominant dimensions

This consideration leads to a hypothetic centroid located in the upper-right part of the radar chart, where Resource and Availability dimensions are dominant (as shown in Figure 4.2).

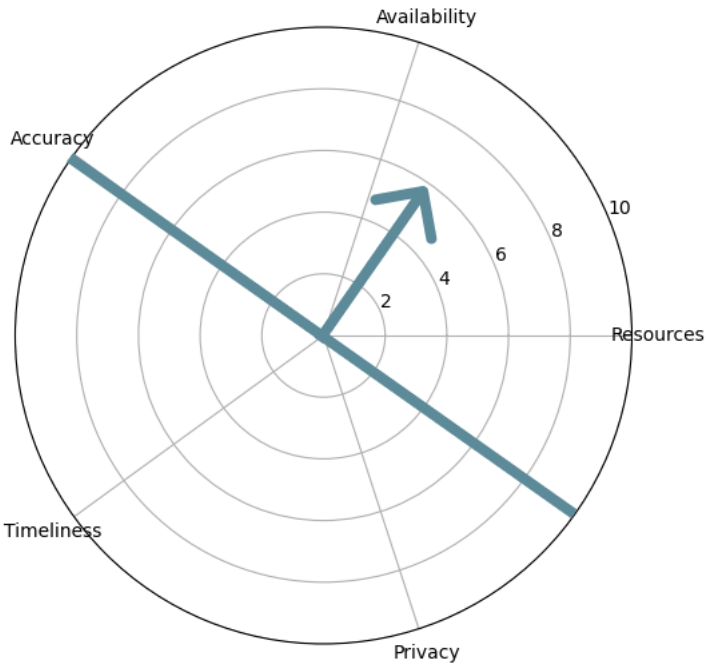

Figure 4.2: Visualization of the polarized part of Distributed Deployment in the Radar Chart

4.2.3. Hybrid deployment

The hybrid approach, which combines the strengths of both offloading and local implementation, can be a compelling design implementation for certain scenarios. This approach leverages the benefits of offloading resource-intensive tasks to the cloud while also retaining some computations on the local device.

In a hybrid implementation, the decision of whether to offload or process locally can be based on various factors, such as the nature of the task, data sensitivity, network conditions, and user preferences.

Some considerations for employing a hybrid approach are:

- Task Partitioning: The application can intelligently partition the workload, offloading computationally heavy tasks to the cloud while keeping critical or latency-sensitive computations locally. This ensures efficient resource utilization and optimized response times.
- Privacy and Security: By processing sensitive data locally, the hybrid approach addresses privacy concerns as the data remains within the user's control. Non-sensitive computations can be offloaded, benefiting from the cloud's computing power without compromising privacy.
- Network Conditions: The hybrid approach considers the variability of network conditions. When a stable and high-bandwidth connection is available, offloading can be advantageous. However, in situations with limited connectivity or high latency, local processing ensures uninterrupted functionality.
- User Experience: The hybrid approach allows a more responsive user experience. Latency-sensitive tasks can be performed locally, reducing network delays, while non-critical tasks can be offloaded, minimizing the impact on device performance.
- Resource Optimization: By distributing computations between the cloud and the local device, the hybrid approach optimizes resource usage. Local processing reduces reliance on network bandwidth and cloud infrastructure, while offloading reduces the strain on local resources.

By staying up-to-date with current research trends and architectures in hybrid solutions, designers and developers can leverage state-of-the-art techniques to enhance existing designs or explore new possibilities.

These advancements contribute to more efficient and effective hybrid solutions, improving overall system performance, privacy, and user satisfaction.

Privacy Intermediate data in deep learning models usually have semantics different from those of row data. It could result in difficulty to understand original information by only observing the features extracted from the original data by CNN filters.
We can then offload high layers of the model, and then offload abstract data processed at the bottom layers of the mobile model side to the background. Finally, partial local computing leverages the user's private data to a certain abstraction and protects it when it is offloaded to the central server.

This could result in a viable option, especially when dealing with applications that interest a consistent group of people and gather sensible information from them. An example scenario could be a crowd-sourcing application.
A novel method called noisy training, where deliberately noisy samples are injected into the training data, has shown results of not only privacy preservation but also improvements in inference performance.

Another solution is Federated learning (FL), which is an approach that combines local training on edge devices with centralized model aggregation.

Federate Learning In this architecture, models are trained locally on user devices, leveraging their data, while periodically sending updates to a central server for global model aggregation. Federated learning addresses privacy concerns by keeping sensitive data on local devices, while still benefiting from collaborative model training and improvement.
Some of these particular benefits include:

- Edge Intelligence: MEC systems combine edge computing capabilities with FL to enable edge intelligence. By training models at the edge, FL leverages the computational power of edge devices and harnesses localized data patterns. This enables real-time or near-real-time decision-making and enhances responsiveness in applications that require low latency, such as IoT analytics, smart cities, and autonomous systems.
- Energy Efficiency: In MEC systems, energy conservation is a crucial concern for devices with limited battery life. FL helps mitigate this challenge by reducing the need for continuous data transmission to a centralized server.

- Data Privacy: FL inherently preserves data privacy as the training data remains on the local devices, and only model updates are shared with the central server.
- Lower Communication Overhead: In MEC systems, devices are typically connected to an edge server or an access point. FL leverages this proximity by performing local model training and aggregating updates at the edge server. This reduces the communication overhead as compared to transmitting raw data to a centralized server.

Timeliness and Resources According to [57], a mobile cloud-based approach could also offer benefits in terms of timeliness or resources dimensions.
This approach involves dividing the Deep Neural Network (DNN) into two parts: a local-side part and a cloud-side part. During the inference phase, the local neural network transforms sensitive data by extracting general information from it and then sends it to the cloud for further complex inference.
By perturbing the original data using nullification and random noise, this approach ensures privacy preservation.

Moreover, it can be advantageous in terms of computational cost since the size of the data to be transmitted is smaller. This is because only the abstract representation of the raw data is transmitted, which reduces the amount of data that needs to be sent to the cloud for processing.

Zhang et al. [63] instead, proposed a novel collaborative inference design aimed at accelerating mobile deep learning applications by formulating the problem of minimizing the total processing cost across several mobile devices.
To this end, they developed a heuristic algorithm based on Particle Swarm Optimization (PSO) to minimize the total time costs associated with collaborative inference efficiently. This approach incorporates a dynamic procedure for selecting the optimal computing nodes from local mobile devices.
This problem can be formulated as a non-linear programming problem that is NP-hard in general.
These mobile devices can connect using direct links, e.g., Bluetooth and WiFi Direct, or cellular links. While this approach appears to be well-suited for scenarios involving groups of users engaged in a shared activity within a confined area, such as virtual reality environments, it is not without limitations.
Experimental testing of the approach with a group of 30 users has revealed that scaling up

to larger user populations can result in significant performance degradation, particularly concerning the time required for the PSO algorithm to identify optimal solutions.

To achieve the desired behaviour, policies can be employed, which consist of a set of rules governing user behaviour. A policy is essentially an ordered list of rules that define a trade-off between time efficiency and memory-usage efficiency.
For instance, a policy might dictate that no more than 500 MB of RAM be used, and that multi-threading (if available) be employed to perform VGG model inference using TensorFlow.

Policies Policies can be employed, which consist of a set of rules governing user behaviour. A policy is essentially an ordered list of rules that define a trade-off between time efficiency and memory-usage efficiency. As we have seen for local-only implementation, we could reuse this idea to directly decide if perform local computation or offloading depending on the current device resources status.
Machine learning techniques, optimization algorithms, or rule-based decision-making approaches can be employed to derive and refine the policy based on real-time or historical data.

Indeed, the concept of using a policy to determine whether to perform local computation or offloading can be a valuable approach in achieving the best performance while considering the trade-off between time efficiency and memory-usage efficiency. By establishing an ordered list of rules or constraints based on the dimensions of the system, a policy can effectively guide the decision-making process.

In the context of resource-constrained devices, such as mobile devices or IoT devices, the policy can take into account factors such as available CPU resources, memory capacity, battery level, network conditions, and specific application requirements.
The policy can be designed to dynamically assess the current device resource status and make informed decisions on whether to perform computations locally or offload them by considering all the dimensions of the implementation chart, especially in terms of Timeliness, Resources and Availability dimensions.

For example, if the device's CPU and memory resources are limited, and the application has no strict timeliness requirements, the policy may prioritize offloading tasks to a remote server to optimize resource utilization and conserve local resources .
On the other hand, if the device has ample resources and low-latency requirements, the

policy may favour local computation to minimize communication overhead and improve timeliness.

Availability We could encounter connectivity constraints, not only in terms of completion time but also in terms of data usage constraints. This is the example of Wi-Fi and cellular data. Using Wi-fi we can focus only on minimizing the completion time since we are not constrained by the internet provider in terms of data usage.
While using cellular data we can face the problem of the data usage constraints. This case has been studied by Lu et al. [31], who focused on this challenge in mobile video analysis, where the task was to identify specific objects through videos sent by crowd-sourcing. Under Wi-Fi connection, where we have to minimize the completion time, an algorithm called split-shift is proposed.
Instead, under cellular data, the optimization goal is the trade-off between processing time and energy consumption, unified with a possible constraint in data usage.

At present, learning-based methods mainly use two strategies, to fulfil the decision-making problem of trade between battery usage, network status, resource allocation, etc. : DDns, which are usually used to construct classifiers, and DRL, which have an excellent performance in decision-making.

However, it is a common belief that gathering data from just one device is not enough to understand and succeed in the task of managing such complex and fragile devices.
Huber et al. aggregate samples from a larger community of devices using crowd-sense evidence traces in order to create a DNN offloading-decision classifier [14].

Cosiderations

This kind of implementation will also imply the possibility of considering that, in case one wishes to employ a different design (i.e., shifting our point from one cluster to another), it will be necessary to lose points in certain dimensions and perhaps gain them in others (vector from the old point to the new one).

A centroid being on the blue distinction line, means having a borderline situation, where there could be not enough information to distinguish which solution could be used. It is near this line, shown in Figure 4.3, that a hybrid approach could be the best solution.
The line is positioned:

- On the Accuracy Axis: this design choice emphasises that Accuracy is the dimension, among all others, that does not keep a polarization on one design solution against another. We can both have a high accuracy by using our local-only implementation and by offloading.
- Between the Resource and Privacy Axes: accordingly to the Accuracy reasoning, in this case, we have our centroid that would be between two of the most polarized dimensions.
  On the one hand, we have the resources dimension which by definition requires an offloading solution due to the device limitation, while on the other hand, we have the Privacy Dimension which is one of the most important aspects of a local-only solution.

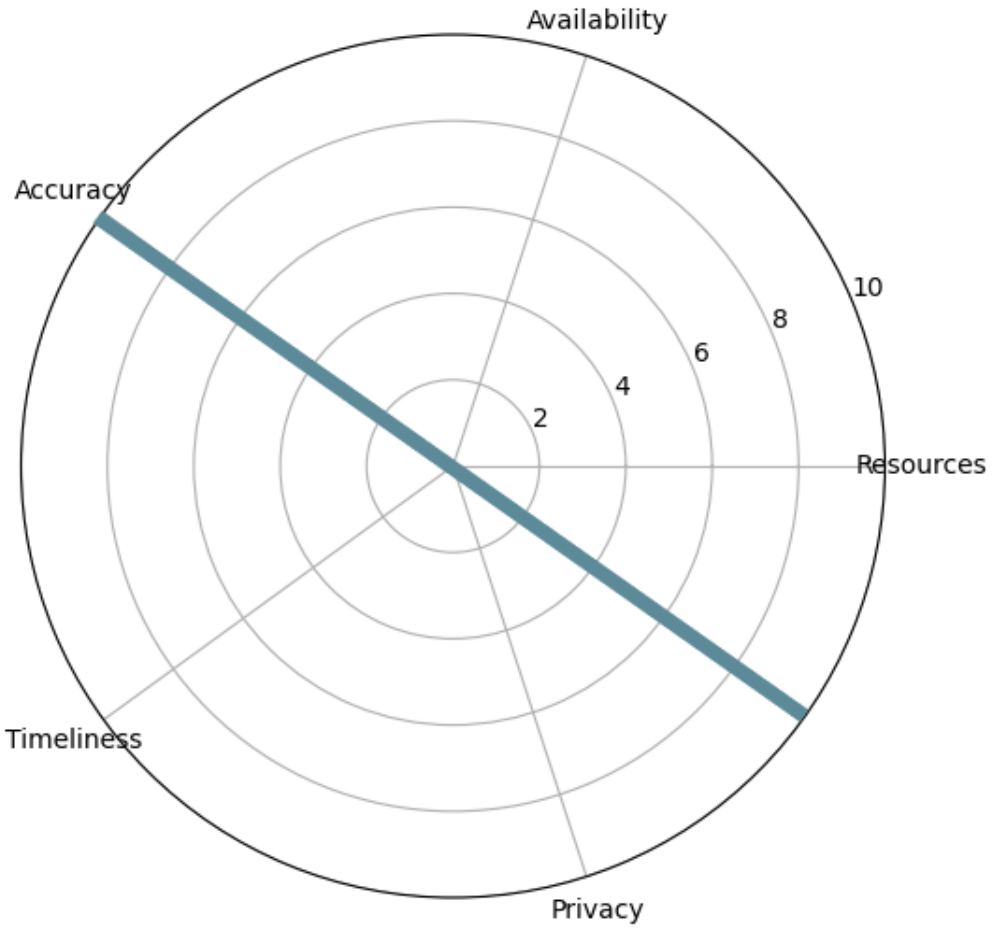

Figure 4.3: Visualization of the polarized part of Hybrid Deployment in the Radar Chart

5 | Use Case

The experimental results shown in this report are inspired by my current work at Reply [47] for one of the leading Fintech companies in Italy and Northen Europe.
The assignment regards implementing a credit card OCR (Optical character recognition) inside the fintech banking app, which is available for both iOS and Android.

The main requirements of the development were to:

- Do not overweight the app (considering its current file size)
- Keep it fast and as available as possible for all kinds of devices.

OCR Optical character recognition (OCR) refers to the process of converting images containing typed, handwritten, or printed text into machine-encoded text. This can be done electronically or mechanically, using various techniques.
OCR is commonly applied to scanned documents, photographs of documents, scene photos capturing text on signs and billboards, and even subtitle text overlaid on images, such as those seen in television broadcasts. The goal is to extract the textual information from these sources and transform it into a format that can be easily manipulated and understood by machines.

The following technical and practical implementation is made only for the Android app, but it can be as well developed for iOS just using the corresponding libraries and technologies for Apple Devices Development.

5.1. Dimensions Measurement and Considerations

5.1.1. Experimental Setup

The current banking app is already available in the Google Play Store and its current size is around 120 MB. We cannot estimate the exact file size of an app since when launched in production, the app bundle which encapsulates the app will then install only essential and required libraries depending on each specific device.
This implies that we can have a gain or even a loss of the final file size, depending on the Android device that is installing the bundle.

5.1.2. Measurement

Given the previous architectural constraints and limitations of a real-world application such as an OCR, the methodology developed in this thesis will consist in assigning a Score value to each of the 5 dimensions with a value in a range from 0 to 10; where 0 means that the dimension is non-dominant at all concerning what is necessary to the model, while 10 is the maximum value to express the dominance of a dimension.

Table 5.1 explains with a brief argumentation the scores of each dimension:

Dimension	Score	Description
Resources	5	Dealing with images for a text classification and recognition task, resulting in two light models
Availability	4	The OCR functionality is only used during credit card registration and no more necessary (rarely used functionality)
Accuracy	6	Numbers are standardized and easily predictable
Timeliness	8	The tool should be as fast as possible (wrt manual input)
Privacy	9	Sensitive information such as front and back credit card numbers

Table 5.1: Use Case Dimension scores

The resulting point (highlighted in blue) in the dimensional space, is represented in the chart represented in Figure 5.1.

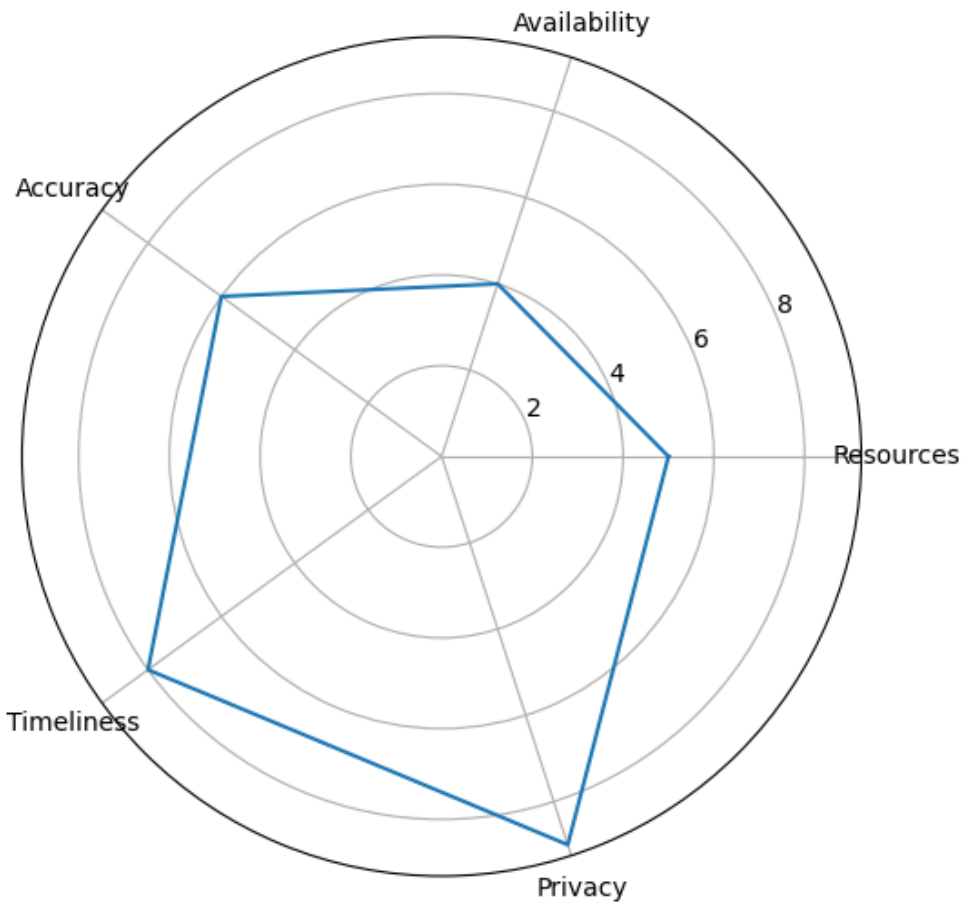

Figure 5.1: Star Plot of Use Case OCR functionality

This results in a centroid (highlighted in blue) weakly polarized on the local design part.

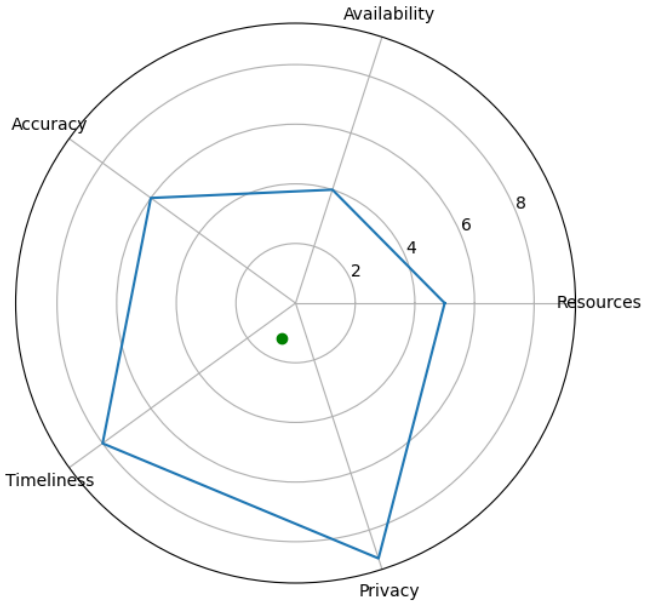

Figure 5.2: Star Plot with centroid of Use Case OCR functionality

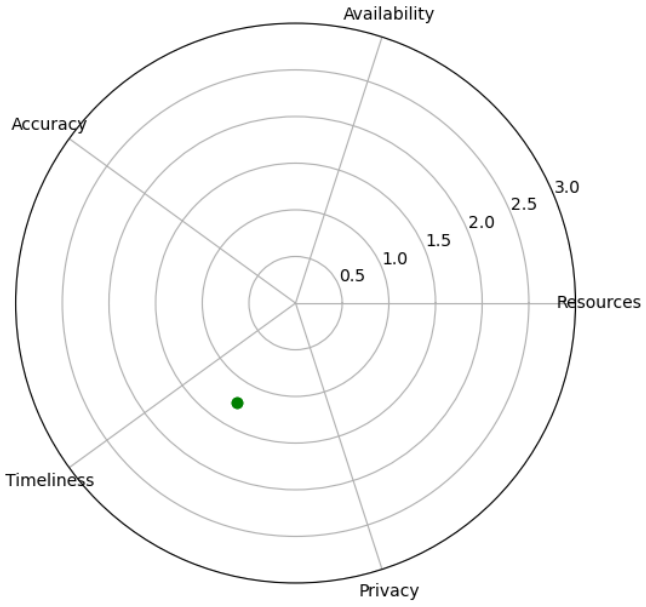

Figure 5.3: Centroid of Use Case OCR functionality

5.2. Practical Implementation

As resulted in Table 5.1 and then highlighted in Figure 5.3, a local deployment is highly recommended since the dominant dimensions are indeed timeliness and privacy.

5.2.1. OCR Model

The model used for the OCR use case is the Keras-OCR [37] architecture, which combines two different models, which are then connected in a pipeline:

- Detector: detects the text on the image
- Recognizer: recognizes the resulting text in each portion of the Detector output

Detector

The model used for OCR detector implementation refers to the CRAFT Model [4], which is composed of a fully convolutional network architecture based on VGG-16 as a backbone. The Figure 5.4 is a graphical representation of the model architecture.
The final output has two channels as score maps:

- the region score: provides information about the extent or scope of an entity or relation within a sentence. It helps identify the boundaries or limits of an entity and its associated context.
- the affinity score: measures how likely two words or phrases are related to each other within the context of a sentence. It indicates the strength of the association or connection between two textual elements.

These outputs will be used to crop the original image with the predicted text regions, and given as input to the recognizer.
In this specific case, given a picture of a credit card, the detection model will recognize the position of the PAN (Payment card number) numbers and expiration date as shown in the Figure 5.5, where the detected text is highlighted in red.

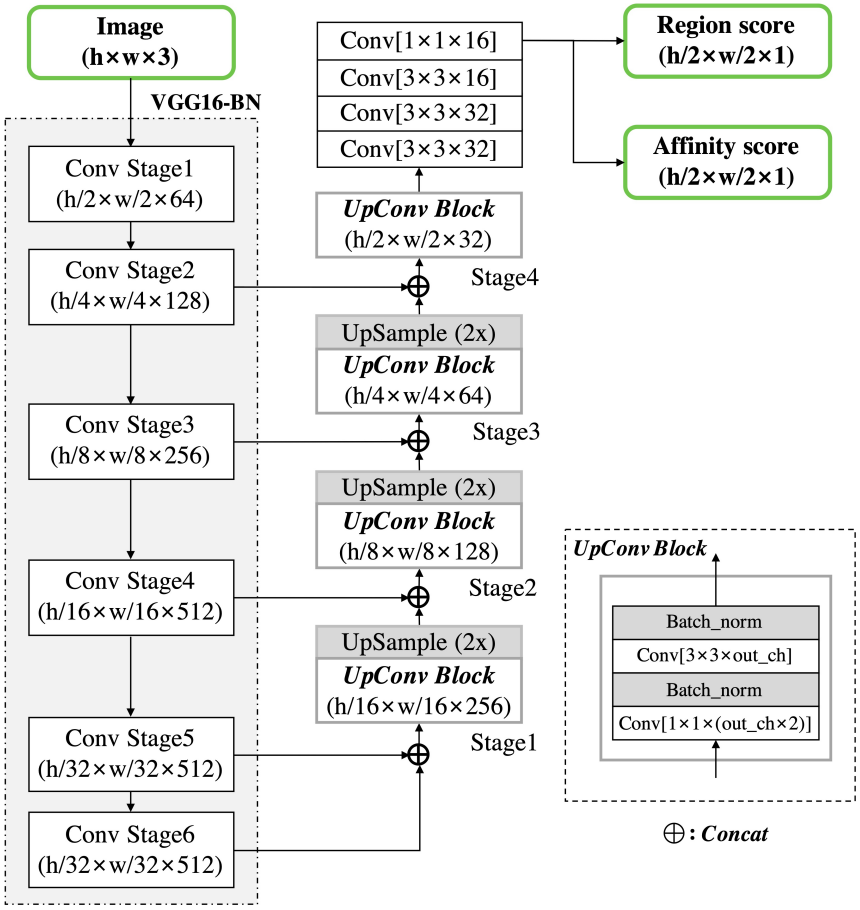

Figure 5.4: CRAFT Detection Model

Figure 5.5: Example of Detection

Recognizer

Once the text has been detected and cropped, the second model is used to recognize the text using a CRNN. The cropped PAN numbers and expiration date are given as input to the recognizer, returning the 16 numbers and the date in format MM/YY.

CRNN Sequential data can be processed and recognized with the aid of CRNNs (Convolutional Recurrent Neural Networks) [49], which fuse both CNNs (Convolutional Neural Networks) and RNNs (Recurrent Neural Networks) into a deep learning structure, as shown in Figure 5.6.
The CRNN architecture is often utilized for end-to-end OCR because it does not need explicit segmentation or feature engineering, and can handle text of any length.

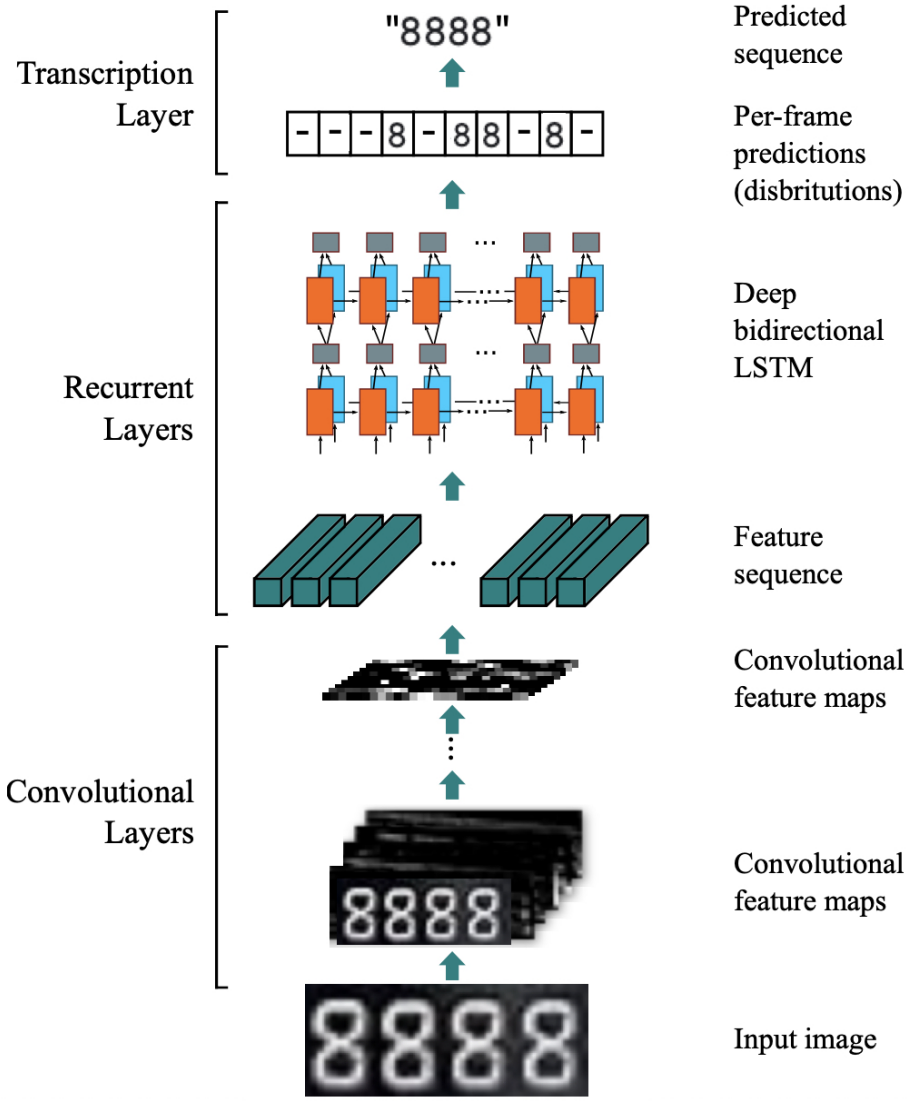

Figure 5.6: CRNN Recognizer Model

5.2.2. Model Conversion

Once trained the model, in order to convert the Keras-OCR model into a supported format for mobile inference, TFLite was used [52]. TFLite, which stands for Tensorflow Lite, is an open-source deep learning framework developed by Google. It's made specifically for mobile and embedded devices as a lightweight version of the TensorFlow framework.

Well-suited for resource-constrained devices like microcontrollers, tablets, smartphones, and other edge devices, TFLite boasts optimized performance.
It facilitates efficient model execution, low-latency inference, and minimal memory footprint enabling models to run locally on the device.

TFLite TFLite offers a revolutionary solution for deploying models on mobile and embedded devices. This process of conversion converts and compresses the TensorFlow model into an optimized format, reducing the model size and minimizing the computations for enhanced inference performance.
Alongside its compatibility with a variety of platforms, including Android, iOS, Linux, Windows, and microcontrollers such as Arduino and Raspberry Pi, TFLite enables model quantization to further improve the model size and rate of inference.
Beyond this, TFLite offers users the opportunity to customize models on their device of choice, enabling the use of locally collected data to fine-tune pre-trained models or train new models.

Quantization The post-training quantization variants available are:

- Dynamic Range Quantization: In this method, the model is quantized using per-channel or per-tensor dynamic range quantization. It involves determining the dynamic range (min and max values) of weights and activations during inference and quantizing them accordingly. Dynamic range quantization supports both integer quantization and float16 quantization.
- Integer Quantization: Integer quantization is a technique where the model's weights and activations are quantized to integer values. TFLite supports several types of integer quantization, including 8-bit quantization, 16-bit quantization, and symmetric quantization. Integer quantization significantly reduces the memory requirements of the model.

- Float16 Quantization: Float16 quantization is a technique that quantizes the weights and activations of the model to 16-bit floating-point values (half-precision). This reduces the memory footprint while maintaining a relatively high level of numerical precision compared to lower-precision integer quantization.

A summary is shown in Table 5.2.

Technique	Benefits	Hardware
Dynamic range quantization	4x smaller, 2x-3x speedup	CPU
Full integer quantization	4x smaller, 3x+ speedup	CPU,Edge TPU, Microcontrollers
Float16 quantization	2x smaller, GPU acceleration	CPU, GPU

Table 5.2: TFLite Quantization

5.3. Evaluation Results

Final considerations lead to a good balance in terms of Resource Dimension and Timeliness, which may vary depending on the quantization technique used.
Table 5.3 shows the benchmark results recorder using the TFLite library in Python.

Model	Quantization Type	Size	Inference
Detection	Dynamic Range	19.9 MB	0.15 sec
	Float16	39.6 MB	0.2 sec
Recognition	Dynamic Range	8.5 MB	0.2 sec
	Float16	16.8 MB	0.2 sec

Table 5.3: Benchmark of OCR models (using a Redmi K20 Pro with 4 threads)

The overall size for a complete pipeline with a Dynamic Range quantization would be 28,4Mb, which is around 20% of the app bundle; the overall inference instead would take in normal device conditions around 0.4 s.
These measurements will then double in the case of a float16 quantization, which would certainly let to more accuracy but, based on our initial assumptions, it is a non-dominant aspect for this specific task.

In summary, the OCR implementation using a local implementation resulted in a good choice, especially in terms of privacy, which is crucial in a banking app.
Nevertheless, exploring also the hybrid solution will certainly let to interesting results.

5.4. Alternative Design

Since our score function recommends the usage of local implementation, especially due to the privacy and timeliness constraints, it does not mean that alternative roads cannot be evaluated.

The nearest implementation in the star chart then, suggests a hybrid approach: this could lead to deeper considerations o reevaluation of the architecture design, especially in terms of Availability, Capacity, and Accuracy.

An alternative approach could consider splitting the OCR pipeline into two parts:

- Detection task performed locally, to detect the PAN and date numbers
- Recognition task offloaded, where the cropped detected text is locally pre-processed and then sent for offloading inference

Privacy In order to solve the Privacy issue, during recognition offloading, we can consider the anonymization of the input data.
The main issues indeed, are the offloading of sensitive content (Credit card information) to an external server and then giving the possibility of a malicious attacker to gather such information.
In this specific use case, the PAN numbers can be shuffled when offloaded and then re-ordered, following some anonymization technique as seen in the previous sections like Distortion Section 4.1.5. This would protect the offloading both by preserving privacy and protecting the data from potential attackers.

Availability, Capacity and Accuracy Easier versioning of the Recognition model would be possible while keeping it server-side. This would be beneficial also in terms of new integration since the finTech company continuously integrates into its circuit new banks, and so the model needs to be re-trained also based on newly available credit cards. This will indirectly influence also the Accuracy dimension.
Furthermore, by offloading the recognition model, more space will be saved, and also less computational effort will be performed on the device.

6 | Conclusions and future developments

Mobile Device design

The redesign of hardware architectures in mobile devices is a promising future direction to enhance the inference of AI models and ensure sufficient device lifetime.
Current hardware architectures in mobile devices are often constrained by factors such as limited power consumption, heat dissipation, and computational capabilities.

However, advancements in hardware design can help overcome these limitations and enable more efficient and effective AI model inference on resource-constrained devices. These advancements will not only enhance the performance of AI-enabled applications but also contribute to extending device lifetime and improving user experience.

Data Management

To achieve improved operation of mobile deep learning, it is crucial to focus on managing contextual data input and continuous feedback.
This involves addressing several key aspects:

- Enriching Mobile Information Acquisition Devices: Mobile devices are equipped with various sensors such as cameras, microphones, and temperature sensors. It is important to explore additional ways to obtain more comprehensive and diverse feature-dimensional data.
  This may involve the integration of new sensors or leveraging existing sensors in innovative ways to capture richer contextual information.

- Improving Data Accuracy: Enhancing the accuracy of data acquired by mobile information acquisition devices is essential. This includes reducing the amount of "dirty data" or erroneous data that may be collected.
  Techniques such as noise reduction algorithms, data filtering, and outlier detection

can be employed to improve data quality. Additionally, efforts should be made to enhance the ability of mobile devices to acquire accurate data even in challenging environments, such as low-light conditions or noisy surroundings.

- Addressing Sensor Heterogeneity: Mobile devices come in various models and configurations, resulting in heterogeneity in sensor quality. It is necessary to tackle the challenge of sensor heterogeneity by developing techniques to normalize or calibrate sensor data across different devices. This ensures consistent and reliable data inputs for mobile deep learning models.

By addressing these aspects, the mobile information collection equipment can be improved to provide more robust and accurate data for mobile deep learning applications.
This, in turn, enhances the overall performance and effectiveness of mobile deep learning systems and enables them to operate optimally in diverse real-world scenarios.

Privacy and Security Offloading user data can introduce privacy and security challenges. Several key issues need to be addressed to ensure the privacy of user data in the offloading process, as described in Section 4.2.2.
Addressing these privacy concerns may require both technical and non-technical considerations:

- Non-technical issues: Designing reward mechanisms or incentives to encourage users to share their data while ensuring their privacy is protected. This involves considering factors such as transparency, trust, and user empowerment in the data-sharing process.
- Technical measures: Implementing data encryption techniques to protect the confidentiality of sensitive information, applying feature abstraction methods to anonymize and de-identify data, and adopting privacy-preserving algorithms or techniques that allow computation on encrypted data without exposing the raw data

Personalized Integrating various input devices and sensors in smartphones has transformed the computing experience and enabled personalized perspectives for users.
These devices generate a vast amount of data, ranging from images and audio recordings to location information and sensor readings. However, due to the limited storage capacity of mobile devices, it is not feasible to store all this data locally.

In the field of MEC architectures, designing detailed strategies and frameworks for Data Lifecycle Management (DLM) becomes essential.

This involves considering the specific needs of users, enterprises, and the overall data ecosystem. By implementing efficient data acquisition, storage, consistency, recovery, updating, and cleaning mechanisms, mobile devices can effectively manage massive data volumes and contribute to enhanced user experiences and enterprise operations.

Network

As discussed in Section 4.2.2, mobile devices by their very nature, possess mobility and often rely on unstable wireless networks and cellular connections. These characteristics introduce challenges in achieving high fault tolerance and maintaining stability when offloading data or computations.
We need to redefine the current networking architecture and move to a device-centric architecture.
To address these challenges, several approaches can be considered:

- Adaptive Offloading Strategies: Designing offloading mechanisms that dynamically adapt to changing network conditions and device locations can improve fault tolerance and stability. This involves continuously monitoring network connectivity, assessing server availability, and making real-time decisions on whether to offload, cache data, or perform computations locally based on the current context.
- Caching Optimization: Employing intelligent caching strategies can mitigate the impact of mobility on offloading decisions. By caching data closer to the device's current location or leveraging nearby servers, the effects of network disruptions and location changes can be minimized, ensuring efficient data access and reducing reliance on remote data transfers.
- Robust Network Resilience: Implementing network resilience techniques, such as connection retries, network protocol optimizations, and error correction mechanisms, can help mitigate the impact of unstable wireless networks. These approaches aim to enhance the fault tolerance and stability of offloading processes in the presence of intermittent connectivity.

By addressing the challenges posed by mobility and unstable wireless networks, it is possible to improve the effectiveness and reliability of offloading strategies, enabling mobile devices to leverage remote resources efficiently while adapting to changing network conditions and device locations.

The power of AI

Ultimately, it is crucial to recognize the immense power of AI as a tool and to employ it judiciously. This regards considering the implications of AI technologies on various aspects of society and prioritizing actions that align with ethical principles. With the incessant rise of smartphones, which are slowly replacing PCs, these considerations must also be combined with the limitations and new opportunities that mobile devices bring.

AI can be leveraged as a powerful tool to solve complex problems, enhance decision-making, and improve various aspects of human life.
It is essential to prioritize the well-being and interests of individuals and society as a whole, ensuring that AI technologies are used for the benefit of humanity while respecting ethical standards and values.

7 | Future Work

Further work is needed to complete and perfectionize this technique. The tool proposed in this work is only a prototype of a final and robust method to implement every kind of AI model in resource-constrained devices.

The Ethical dimension As discussed in Chapter 6, conscious and ethical use of the technology should be followed, and the introduction of the Ethical 6th dimension should be taken into consideration.
Some applications may deal with sensible data, location or even healthcare values. Indeed, security policies and privacy constraints may be adopted to prevent data leakage or even worse the manipulation of such data.
Worst-case scenarios could lead to damage to user health, where the unauthorized access of a malicious attacker to such kinds of applications can give the possibility to block healthcare devices such as insulin pumps or pacemakers.

Benchmarking A survey made on actual real-world projects, considering the deployment type and their overall performance, can be useful to practise this tool and refine it. Making a score based on each project assumption, without taking into consideration which implementation has been effectively deployed, and then comparing the prediction from the graphical tool proposed in this thesis and the real architecture, could help in understanding its real effectiveness of it.

Dimensional policy approach As discussed in Section 4.2.3, a policy rule-based approach constrained on the 5 dimensions can be used. This approach will consist of a balanced trade-off between hardware resource usage and model performance, with a set of parameters that will keep track of the current resource usage, connectivity constraint, battery and availability of the service.

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems, 2016.

[2] S. Anwar, K. Hwang, and W. Sung. Structured pruning of deep convolutional neural networks. *J. Emerg. Technol. Comput. Syst.*, 13(3), feb 2017. ISSN 1550-4832. doi: 10.1145/3005348. URL `https://doi.org/10.1145/3005348`.

[3] ARM. Arm limited - roadshow slides q4 2019, 2019. URL `https://group.softbank/system/files/pdf/ir/presentations/2019/arm-roadshow-slides_q4fy2019_01_en.pdf`.

[4] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee. Character region awareness for text detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9365–9374, 2019.

[5] S. Bi and Y. J. Zhang. Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading. *IEEE Transactions on Wireless Communications*, 17(6):4177–4190, 2018. doi: 10.1109/TWC.2018.2821664.

[6] I. Bonifacic. We asked, you told us: Here's how much storage the average aa reader uses, 2020. URL `https://www.androidauthority.com/phone-storage-poll-results-1177469/`.

[7] B. Boshell. Average app file size: Data for android and ios mobile apps, 2017. URL `https://sweetpricing.com/blog/index.html%3Fp=4250.html/`.

[8] M. Chen and Y. Hao. Task offloading for mobile edge computing in software defined ultra-dense network. *IEEE Journal on Selected Areas in Communications*, 36(3): 587–597, 2018. doi: 10.1109/JSAC.2018.2815360.

[9] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen. Compressing neural networks with the hashing trick, 2015.

[10] W.-C. Chien, S.-Y. Huang, C.-F. Lai, H.-C. Chao, M. S. Hossain, and G. Muhammad. Multiple contents offloading mechanism in ai-enabled opportunistic networks. *Computer Communications*, 155:93–103, 2020. ISSN 0140-3664. doi: https://doi.org/10.1016/j.comcom.2020.02.084. URL https://www.sciencedirect.com/science/article/pii/S0140366419314239.

[11] D. et al. Imagenet large scale visual recognition challenge 2012 (ilsvrc2012), 2012. URL https://www.image-net.org/challenges/LSVRC/2012/.

[12] B. Fang, X. Zeng, and M. Zhang. Nestdnn: Resource-aware multi-tenant on-device deep learning for continuous mobile vision. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, MobiCom '18, page 115–127, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450359030. doi: 10.1145/3241539.3241559. URL https://doi.org/10.1145/3241539.3241559.

[13] M. Febvay. Low-level optimizations for faster mobile deep learning inference frameworks. *MM 2020 - Proceedings of the 28th ACM International Conference on Multimedia*, 2020. doi: 10.1145/3394171.3416516.

[14] H. Flores, P. Hui, P. Nurmi, E. Lagerspetz, S. Tarkoma, J. Manner, V. Kostakos, Y. Li, and X. Su. Evidence-aware mobile computational offloading. *IEEE Transactions on Mobile Computing*, 17(8):1834–1850, 2018. doi: 10.1109/TMC.2017.2777491.

[15] F. M. I. Global and C. P. Ltd. Mobile applications market to expand at a cagr of 8.6% and to reach us$ 170.2 billion by 2033 | future market insights, inc., 2023. URL https://www.globenewswire.com/en/news-release/2023/01/11/2586687/0/en/Mobile-Applications-Market-to-Expand-at-a-CAGR-of-8-6-to-Reach-US-170-2-Billion-by-html.

[16] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan. Deep learning with limited numerical precision. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1737–1746, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/gupta15.html.

[17] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, 2016.

[18] S. Han, H. Shen, M. Philipose, S. Agarwal, A. Wolman, and A. Krishnamurthy. Mcdnn: An approximation-based execution framework for deep stream processing under resource constraints. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '16, page 123–136, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342698. doi: 10.1145/2906388.2906396. URL https://doi.org/10.1145/2906388.2906396.

[19] M. Hanif, M. U. Javed, R. Hafiz, S. Rehman, and M. Shafique. *Hardware–Software Approximations for Deep Neural Networks: Methodologies and CAD*, pages 269–288. Hanif, 01 2019. ISBN 978-3-319-99321-8. doi: 10.1007/978-3-319-99322-5_13.

[20] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network, 2015.

[21] H. Huang, Z. Hongyuan, C. Hu, C. Chen, and Y. Li. Find and dig: A privacy-preserving image processing mechanism in deep neural networks for mobile computation. In *Find and Dig: A Privacy-Preserving Image Processing Mechanism in Deep Neural Networks for Mobile Computation*, pages 1–8, 07 2021. doi: 10.1109/IJCNN52387.2021.9534066.

[22] L. Huang, X. Feng, C. Zhang, L. Qian, and Y. Wu. Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing. *Digital Communications and Networks*, 5(1):10–17, 2019.

[23] L. N. Huynh, Y. Lee, and R. K. Balan. Deepmon: Mobile gpu-based deep learning framework for continuous vision applications. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '17, page 82–95, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349284. doi: 10.1145/3081333.3081360. URL https://doi.org/10.1145/3081333.3081360.

[24] M. IQBAL. App download data (2023), 2023. URL https://www.businessofapps.com/data/app-statistics/.

[25] W. Jiang, C. Miao, F. Ma, S. Yao, Y. Wang, Y. Yuan, H. Xue, C. Song, X. Ma, D. Koutsonikolas, W. Xu, and L. Su. Towards environment independent device free human activity recognition. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, MobiCom '18, page 289–304, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450359030. doi: 10.1145/3241539.3241548. URL https://doi.org/10.1145/3241539.3241548.

[26] A. Kumar, M. Gyanchandani, and P. Jain. A comparative review of privacy preservation techniques in data publishing. In *2018 2nd International Conference on Inventive Systems and Control (ICISC)*, pages 1027–1032, 2018. doi: 10.1109/ICISC.2018.8398958.

[27] N. D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, L. Jiao, L. Qendro, and F. Kawsar. Deepx: A software accelerator for low-power deep learning inference on mobile devices. In *2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 1–12, 2016. doi: 10.1109/IPSN.2016.7460664.

[28] S. Lawrence, C. Giles, and A. Tsoi. What size neural network gives optimal generalization? convergence properties of backpropagation. -, 03 2001.

[29] Y. Lecun, J. Denker, and S. Solla. Optimal brain damage. In *Optimal Brain Damage*, volume 2, pages 598–605, 01 1989.

[30] L. Lu, J. Yu, Y. Chen, H. Liu, Y. Zhu, L. Kong, and M. Li. Lip reading-based user authentication through acoustic sensing on smartphones. *IEEE/ACM Transactions on Networking*, 27(1):447–460, 2019. doi: 10.1109/TNET.2019.2891733.

[31] Z. Lu, K. Chan, S. Pu, and T. L. Porta. Crowdvision: A computing platform for video crowdprocessing using deep learning. *IEEE Transactions on Mobile Computing*, 18 (7):1513–1526, 2019. doi: 10.1109/TMC.2018.2864212.

[32] P. Mach and Z. Becvar. Mobile edge computing: A survey on architecture and computation offloading. *IEEE Communications Surveys and Tutorials*, 19(3):1628–1656, 2017. doi: 10.1109/COMST.2017.2682318.

[33] P. Mach and Z. Becvar. Mobile edge computing: A survey on architecture and computation offloading. *IEEE Communications Surveys and Tutorials*, 19(3):1628–1656, 2017. doi: 10.1109/comst.2017.2682318. URL `https://doi.org/10.1109%2Fcomst.2017.2682318`.

[34] Q. Mao, F. Hu, and Q. Hao. Deep learning for intelligent wireless networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 20(4):2595–2621, 2018. doi: 10.1109/COMST.2018.2846401.

[35] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief. A survey on mobile edge computing: The communication perspective. *IEEE Communications Surveys & Tutorials*, 19(4):2322–2358, 2017. doi: 10.1109/COMST.2017.2745201.

[36] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief. A survey on mobile edge computing: The communication perspective, 2017.

[37] F. Morales. Keras-ocr, -. URL https://github.com/faustomorales/keras-ocr/.

[38] F. Moya Rueda, R. Grzeszick, and G. A. Fink. Neuron pruning for compressing deep networks using maxout architectures. In V. Roth and T. Vetter, editors, *Pattern Recognition*, pages 177–188, Cham, 2017. Springer International Publishing. ISBN 978-3-319-66709-6.

[39] L. Nguyen Huynh, Y. Lee, and R. Balan. Deepmon: Mobile gpu-based deep learning framework for continuous vision applications. In *DeepMon: Mobile GPU-based Deep Learning Framework for Continuous Vision Applications*, pages 82–95, 06 2017. doi: 10.1145/3081333.3081360.

[40] NI. What is availability?, 2023. URL https://www.ni.com/it-it/shop/electronic-test-instrumentation/add-ons-for-electronic-test-and-instrumentation/what-is-systemlink-tdm-datafinder-module/what-is-rasm/what-is-availability-.html#section--2084781321.

[41] S. A. Osia, A. S. Shamsabadi, S. Sajadmanesh, A. Taheri, K. Katevas, H. R. Rabiee, N. D. Lane, and H. Haddadi. A hybrid deep learning architecture for privacy-preserving mobile analytics. *IEEE Internet of Things Journal*, 7(5):4505–4518, may 2020. doi: 10.1109/jiot.2020.2967734. URL https://doi.org/10.1109%2Fjiot.2020.2967734.

[42] M. Ozturk, M. Gogate, O. Onireti, A. Adeel, A. Hussain, and M. A. Imran. A novel deep learning driven, low-cost mobility prediction approach for 5g cellular networks: The case of the control/data separation architecture (cdsa). *Neurocomputing*, 358:479–489, 2019. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2019.01.031. URL https://www.sciencedirect.com/science/article/pii/S0925231219300438.

[43] PaddlePaddle. Paddle, -. URL https://www.paddlepaddle.org.cn/en.

[44] J. Park, S. Li, W. Wen, P. T. P. Tang, H. Li, Y. Chen, and P. Dubey. Faster cnns with direct sparse convolutions and guided pruning, 2017.

[45] J. Ren, L. Gao, H. Wang, and Z. Wang. Optimise web browsing on heterogeneous mobile platforms: A machine learning based approach. In *IEEE INFOCOM 2017*

- *IEEE Conference on Computer Communications*, pages 1–9, 2017. doi: 10.1109/INFOCOM.2017.8057087.

[46] X. Ren and J. Yang. Research on privacy protection based on k-anonymity. In *2010 International Conference on Biomedical Engineering and Computer Science*, pages 1–5, 2010. doi: 10.1109/ICBECS.2010.5462427.

[47] Reply. Reply, —-. URL `https://www.reply.com/en`.

[48] A. Shawahna, S. M. Sait, and A. El-Maleh. Fpga-based accelerators of deep learning networks for learning and classification: A review. *IEEE Access*, 7:7823–7859, 2019. doi: 10.1109/ACCESS.2018.2890150.

[49] B. Shi, X. Bai, and C. Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition, 2015.

[50] C. Tai, T. Xiao, Y. Zhang, X. Wang, and W. E. Convolutional neural networks with low-rank regularization, 2016.

[51] F. Tang, B. Mao, Z. M. Fadlullah, N. Kato, O. Akashi, T. Inoue, and K. Mizutani. On removing routing protocol from future wireless networks: A real-time deep learning approach for intelligent traffic control. *IEEE Wireless Communications*, 25(1):154–160, 2018. doi: 10.1109/MWC.2017.1700244.

[52] Tensorflow. Tensorflow lite, —-. URL `https://www.tensorflow.org/lite`.

[53] Tibco. What is a radar chart?, –. URL `https://www.tibco.com/reference-center/what-is-a-radar-chart`.

[54] transformainsights. Current iot forecast highlights, 2023. URL `https://transformainsights.com/research/forecast/highlights`.

[55] T. Tuor, S. Wang, T. Salonidis, B. J. Ko, and K. K. Leung. Demo abstract: Distributed machine learning at resource-limited edge nodes. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–2, 2018. doi: 10.1109/INFCOMW.2018.8406837.

[56] B. WANG. Average smartphone nand flash capacity crossed 100gb in 2020, 2021. URL `https://www.counterpointresearch.com/average-smartphone-nand-flash-capacity-crossed-100gb-2020/`.

[57] J. Wang, B. Cao, P. S. Yu, L. Sun, W. Bao, and X. Zhu. Deep learning towards mobile applications, 2018.

[58] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan. When edge meets learning: Adaptive control for resource-constrained distributed machine learning. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pages 63–71, 2018. doi: 10.1109/INFOCOM.2018.8486403.

[59] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen. In-edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning. *IEEE Network*, 33(5):156–165, sep 2019. doi: 10.1109/mnet.2019.1800286. URL `https://doi.org/10.1109%2Fmnet.2019.1800286`.

[60] X. Wang, Z. Zhou, F. Xiao, K. Xing, Z. Yang, Y. Liu, and C. Peng. Spatio-temporal analysis and prediction of cellular traffic in metropolis. *IEEE Transactions on Mobile Computing*, 18(9):2190–2202, 2019. doi: 10.1109/TMC.2018.2870135.

[61] D. S. Wickramasuriya, C. A. Perumalla, K. Davaslioglu, and R. D. Gitlin. Base station prediction and proactive mobility management in virtual cells using recurrent neural networks. In *2017 IEEE 18th Wireless and Microwave Technology Conference (WAMICON)*, pages 1–6, 2017. doi: 10.1109/WAMICON.2017.7930254.

[62] Wolfram. Geometric centroid, 2021. URL `https://mathworld.wolfram.com/GeometricCentroid.html`.

[63] C. Wu, l. Zhang, Q. Li, Z. Fu, W. Zhu, and Y. Zhang. Enabling flexible resource allocation in mobile deep learning systems. *IEEE Transactions on Parallel and Distributed Systems*, 30(2):346–360, 2019. doi: 10.1109/TPDS.2018.2865359.

[64] D. Wu, Q. Liu, H. Wang, Q. Yang, and R. Wang. Cache less for more: Exploiting cooperative video caching and delivery in d2d communications. *IEEE Transactions on Multimedia*, 21(7):1788–1798, 2019. doi: 10.1109/TMM.2018.2885931.

[65] W. Wu, F. Zhou, R. Q. Hu, and B. Wang. Energy-efficient resource allocation for secure noma-enabled mobile edge computing networks, 2019.

[66] Q. Xu and R. Zheng. When data acquisition meets data analytics: A distributed active learning framework for optimal budgeted mobile crowdsensing. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pages 1–9, 2017. doi: 10.1109/INFOCOM.2017.8057034.

[67] J. Xue, J. Li, and Y. Gong. Restructuring of deep neural network acoustic models with singular value decomposition. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pages 2365–2369, 01 2013.

[68] P. Yin, J. Lyu, S. Zhang, S. Osher, Y. Qi, and J. Xin. Understanding straight-through estimator in training activation quantized neural nets, 2019.

[69] H. Zhang, C. Song, A. Wang, C. Xu, D. Li, and W. Xu. Pdvocal: Towards privacy-preserving parkinson's disease detection using non-speech body sounds. In *The 25th Annual International Conference on Mobile Computing and Networking*, MobiCom '19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450361699. doi: 10.1145/3300061.3300125. URL https://doi.org/10.1145/3300061.3300125.

[70] H. Zhang, C. Xu, H. Li, A. S. Rathore, C. Song, Z. Yan, D. Li, F. Lin, K. Wang, and W. Xu. Pdmove: Towards passive medication adherence monitoring of parkinson's disease using smartphone-based gait assessment. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 3(3), sep 2019. doi: 10.1145/3351281. URL https://doi.org/10.1145/3351281.

[71] B. Zhou, J. Lohokare, R. Gao, and F. Ye. Echoprint: Two-factor authentication using acoustics and vision on smartphones. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, MobiCom '18, page 321–336, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450359030. doi: 10.1145/3241539.3241575. URL https://doi.org/10.1145/3241539.3241575.

List of Figures

List of Tables

Acknowledgements

To my mother, my father and my brother. To my grandparents, uncles and cousins. To all my family, that is my constant inspiration. To all their sacrifices made for me. To all of them my infinite gratitude.

To Francesca, who followed and supported me during this journey. Especially in the hardest times.

To my friends. To their love for life. They always remind me that there must be an equilibrium between cheerfulness and seriousness.