Technische Universität München
TUM School of Engineering and Design

Artificial Intelligence-augmented Edge Processing for the Physical Layer on Telecommunication Satellites

Michael Klaus Petry

Vollständiger Abdruck der von der TUM School of Engineering and Design der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Vorsitz:

Prof. Dr. rer. nat. Katharina Anders

Prüfende der Dissertation:

1. Prof. Dr. rer. nat. Martin Werner
2. Prof. Shiwen Mao, Ph.D.
3. Prof. Dr. phil. Alessandro Aliakbargolkar

Die Dissertation wurde am 18.02.2025 bei der Technischen Universität München eingereicht und durch die TUM School of Engineering and Design am 08.05.2025 angenommen.

Acknowledgement

First and foremost, I am deeply grateful to my PhD supervisor Martin for giving me the opportunity to be part of his team at the TUM Department of Aerospace and Geodesy. Throughout my journey, his support came in many facets - from guidance on technical challenges, to offering new perspectives from a different but related field, opening up international networks, and supporting my participation in conferences. All of this enabled me to build a stronger technical foundation and a deep international network. I'm also sincerely thankful to his team, whose input from diverse perspectives helped broaden my horizon.

This research journey has greatly benefited from the industrial perspective I was able to take on this topic, thanks to the unique cooperation between the Technical University of Munich and Airbus Defence and Space GmbH. I am deeply grateful to the company for enabling practice-oriented research in such a special setting - which was particularly valuable for my research in the field of aerospace, which is often enclosed within the walls of private companies.

In the end, of course, it's the people who make such ventures both meaningful and enjoyable by investing their time, energy, and support. I would especially like to thank Richard and Tim for granting me the scientific freedom within the company to pursue my research openly and without bias, and for enabling me to publish this work. I am also thankful to my colleagues who made this journey a fun and collaborative one - especially Andreas for our countless discussions on Machine Learning algorithms, Patrick for his ever-present support in bringing these algorithms to hardware, and Raphael for the fun we had bringing Artificial Intelligence and 5G together while prototyping a space-grade base station. Last but certainly not least, thank you Harvey for taking such good care of our projects on the management side, which allowed me to focus on technical problems rather than administrative ones.

I also feel lucky to have worked with many talented Bachelor's and Master's students who brought in fresh perspectives - thank you all. Your contributions became important parts of this work.

This thesis has also greatly benefited from discussions with fellow researchers on Artificial Intelligence for wireless communication beyond the walls of TUM, and also internationally. Having had the chance to be in touch with them from the beginning of this journey put me on the right track early on, something I am very grateful for.

Finally, the greatest thanks goes to my family for being a constant source of strength and support - not only during this journey, but from day one, and for everything still to come. Their reassuring counsel on the (sometimes unusual) paths I've taken has been the foundation of my personal and professional achievements.

Abstract

This dissertation explores the impact of Artificial Intelligence (AI) on satellite systems, with a particular focus on communication-capable payloads. The rapid advancements in AI, both in hardware technology and algorithms, have sparked a renaissance of innovation across numerous fields over the past decade - an evolution that has gained significant traction in satellite and communication technologies. While terrestrial communication systems, such as 5G and 6G, are already undergoing a fundamental transformation driven by AI - redefining design paradigms and functionality - this wave of innovation has yet to fully extend to non-terrestrial systems. This raises a key question: How will AI-driven advancements shape the future of telecommunications satellites, and how can these systems be designed to be future-fit? In this dissertation, we examine this question from multiple perspectives.

We begin by investigating hardware technology, a critical factor in enabling efficient, high-performance, and space-qualified on-board AI computing. After identifying the VERSAL AI Core System-on-Chip as a virtually unrivaled computing platform, we analyze its capabilities before shifting our focus from hardware to compatible AI algorithms for operation in the physical layer of communication systems. We explore two prominent approaches: AI-native algorithms, where all operations are performed exclusively using neural networks (NNs), and AI-augmented algorithms, where NNs with specialized functionality are integrated as supplementary components within a classical signal processing chain.

Although we demonstrate that a fully AI-native, end-to-end trained communication system can be successfully implemented on the aforementioned hardware - offering inherent reconfigurability through the ability to update or exchange neural networks - this approach quickly reaches its computational limits due to carrying the entire processing load. Consequently, we refine the application of deep learning by adopting the AI-augmented approach. We develop a combination of specialized NNs based on the Radio Transformer architecture which are tasked with the synchronization of radio frequency signals, an essential component of any communication system.

This transition to a hybrid intertwining of classical algorithms and AI exemplifies the growing complexity of future signal processing techniques. Their efficient implementation requires a heterogeneous computing architecture that integrates both general-purpose and AI-optimized hardware technologies while also ensuring their seamless interplay - an objective we successfully achieve.

Finally, we contextualize our findings with the AI-related requirements for Non–Terrestrial Networks (NTNs) as defined in the 5G standard. Based on these insights, we develop a future-proof prototype of an AI-enabled base station designed to meet foreseeable AI demands in future satellite communication systems.

Kurzfassung

Diese Dissertation untersucht die Auswirkungen von künstlicher Intelligenz (KI) auf Satellitensysteme mit einem speziellen Fokus auf kommunikationsfähigen Nutzlasten. Das explosionsartige Fortschreiten der Leistungsfähigkeit von KI bezüglich Hardwaretechnologie und Algorithmik hat in der letzten Dekade eine Renaissance an Innovationskraft in einer Vielzahl von Themengebiet entfacht, welche unter anderem im Bereich der Satellitentechnik als auch der Nachrichtentechnik großen Anklang gefunden hat. Obwohl terrestrische Kommunikationssysteme, wie zum Beispiel 5G und 6G, bereits schon heute einen KI-getriebenen fundamentalen Wandel bezüglich Designparadigma und Funktionalität unterworfen sind, hat die Innovationswelle nicht-terrestrische Systeme bisher noch nicht erreicht. Dies weckt die grundlegende Frage: Wie wird sich der KI-getriebene Fortschritt auf Telekommunikationssatelliten auswirken und wie kann man diese Systeme möglichst zukunftssicher gestalten? In dieser Dissertation betrachten wir diese Fragestellung von verschiedenen Perspektiven.

Wir beginnen mit dem Thema Hardwaretechnologie, welches ein kritischer Schlüsselfaktor beim Bereitstellen von effizienter, leistungsfähiger und den speziellen Umgebungsbedingungen des Weltraums standhaltender On-Board KI-Rechenkapazität ist. Nach Identifikation und Analyse des VERSAL AI Core System-on-Chips als nahezu konkurrenzlose Rechenplattform legen wir den Schwerpunkt von der Hardware auf die dazu kompatible KI Algorithmik mit Einsatz in der physikalischen Bitübertragungsschicht von Kommunikationssystemen. Hierbei verfolgen wir zwei populäre Ansätze: KI-native Algorithmik, in der jegliche Operationen ausschließlich mittels Neuronalen Netzen (NN) durchgeführt werden und KI-augmentierte Algorithmik, in der NN mit spezieller Funktionalität lediglich ergänzend in einer klassischen Signalverarbeitungskette platziert werden.

Obwohl wir zeigen, dass ein Ende-zu-Ende trainiertes KI-natives Kommunikationssystem erfolgreich auf obiger Hardware realisiert werden kann und dies zusätzlich eine inhärent gegebene Rekonfigurierbarkeit durch Austausch der NN ermöglicht, stoßen die KI-Rechenkapazitäten durch das Tragen der gesamten Rechenlast zu schnell an ihre Grenzen. Infolgedessen verfeinern wir den Einsatz von Maschinellem Lernen, indem wir den Ansatz der KI-Augmentation adaptieren. Wir entwickeln eine Kombination von speziell entworfenen NN basierend auf der Radio-Transformer Architektur, welche sich primär mit der Aufgabe der Synchronisierung von Radio Frequenz Signalen beschäftigt, einer essenziellen Komponente jedes Kommunikationssystems.

Dieser Übergang zu einer hybriden Verflochtenheit von klassischer Algorithmik und KI repräsentiert beispielhaft die Komplexität zukünftiger Signalverarbeitungsprozesse, welche hinsichtlich einer effizienten Implementierung eine heterogene Rechenarchitektur mit generischen und KI-optimierten Hardwaretechnologien voraussetzt, als auch deren

enges Zusammenspiel erfordert, welches wir erfolgreich realisieren können.

Schlussendlich ordnen wir die zuvor gewonnenen Erkenntnisse den im 5G Standard definierten Anforderungen an KI-Funktionalität im Hinblick auf nicht-terrestrische Systeme ein und entwerfen einen Prototyp einer KI-fähigen Basisstation mit dem Ziel, vorhersehbaren KI Anforderungen gerecht zu werden.

Contents

List of Figures

List of Tables

Acronyms

3GPP	3rd Generation Partnership Project.
5G	Fifth generation technology standard for cellular communications.
6G	Sixth generation technology standard for cellular communications.
ADC	Analog-to-Digital Converter.
AE	Autoencoder.
AGC	Automatic Gain Control.
AI	Artificial Intelligence.
AM	Amplitude Modulation.
APSK	Amplitude and Phase-shift Keying.
ASIC	Application-specific Integrated Circuit.
AutoDiff	Automatic Differentiation.
AWGN	Additive White Gaussian Noise.
BCH	Bose-Chaudhuri-Hocquenghem (Code).
BER	Bit Error Rate.
BICM	Bit-Interleaved Coded Modulation.
BLER	Block Error Rate.
CFO	Carrier Frequency Offset.
CNN	Convolutional Neural Network.
COTS	Commercial Off-The-Shelf.
CPO	Constant Phase Offset.
CPU	Central Processing Unit.
CRC	Cyclic Redundancy Check.
CU	Central Unit.
DAB	Digital Audio Broadcasting.
DAC	Digital-to-Analog Converter.
DDS	Direct Digital Synthesis.
DFKI	German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz).
DFT	Discrete Fourier Transform.

DL	Deep Learning.
DLR	German Aerospace Center (Deutsches Zentrum für Luft- und Raumfahrt).
DPU	Deep Learning Processor Unit.
DSP	Digital Signal Processor.
DU	Distributed Unit.
DVB	Digital-Video-Broadcast.
E2E	End-to-End.
ECC	Error Correction Coding.
ECSS	European Cooperation for Space Standardization.
EO	Earth Observation.
ESA	European Space Agency.
EVB	Evaluation Board.
FD-SOI	Fully Depleted Silicium on Insulator.
FDIR	Fault-Detection, Fault-Isolation and Recovery Techniques.
FEC	Forward Error Correction.
FFDL	Feed-forward Dense Layers.
FFT	Fast Fourier Transform.
FM	Frequency Modulation.
FPGA	Field Programmable Gate Array.
GAN	Generative Adversarial Network.
GenAI	Generative AI.
GEO	Geostationary Earth Orbit.
GNN	Graph Neural Network.
GPU	Graphics Processing Unit.
GSM	Global System for Mobile Communications.
HAPS	High Altitude Platform Systems.
HARQ	Hybrid Automatic Repeat Request.
HTS	High-Throughput Satellite.
IC	Integrated Circuit.
IF	Intermediate Frequency.
IoT	Internet of Things.
IP	Intellectual Property.
ISAC	Integrated Sensing and Communications.
ISI	Inter-symbol Interference.
KPI	Key Performance Indicator.

LCM	Lifecycle Management.
LDPC	Low-density Parity-check Codes.
LEO	Low Earth Orbit.
LLM	Large Language Model.
LLR	Log-Likelihood Ratio.
LO	Local Oscillator.
LSTM	Long short-term Memory.
LTE	Long Term Evolution.
M2M	Machine-to-Machine.
MAC	Medium Access Control.
MCS	Modulation and Coding Scheme.
MIMO	Multiple-Input Multiple-Output.
ML	Machine Learning.
MLP	Multi-Layer Perceptron.
NASA	National Aeronautics and Space Agency.
NCO	Numerically-Controlled Oscillator.
NLP	Natrual Language Processing.
NN	Neural Network.
NoC	Network-on-Chip.
NR	New Radio.
NTN	Non-Terrestrial Network.
NTSC	National Television Systems Committee.
OFDM	Orthogonal Frequency-Division Multiplexing.
OSI	Open Systems Interconnection.
OTA	Over-the-Air.
PAL	Phase-Alternating-Line.
PCIe	Peripheral Component Interconnect Express.
PL	Programmable Logic.
PLL	Phase-Locked Loop.
QAM	Quadrature Amplitude Modulation.
QoS	Quality-of-Service.
QPSK	Quadrature Phase Shift Keying.
RAN	Radio Access Network.
RF	Radio Frequency.
RL	Reinforcement Learning.
RNN	Recurrent Neural Network.
RTN	Radio Transformer Network.
RX	Receiver.

SAR	Synthetic Aperture Radar.
SDR	Software-defined Radio.
SFO	Sampling Frequency Offset.
SGD	Stochastic Gradient Descent.
SIGINT	Signal Intelligence.
SNR	Signal-to-Noise Ratio.
SoC	System-on-Chip.
SoM	System-on-Module.
STO	Sampling Time Offset.
SWaP	Size, Weight, and Power.
TN	Terrestrial Network.
TPU	Tensor Processing Unit.
TX	Transmitter.
UE	User Equipment.
USRP	Universal Software Radio Peripheral.
VPU	Vector Processing Unit.
WLAN	Wireless Local Area Network.

1 Introduction

1.1 Unleashing the Potential of Edge-AI on Telecommunication Satellites

Artificial Intelligence (AI) has took the world by storm since the beginning of this decade. With the release of Large Language Models (LLMs) and their accompanying applications in late 2022 like chat bots such as ChatGPT, nobody that uses modern technologies such as computers, smartphones, or the internet on a regular basis had a chance to escape this phenomenon. However, unlike the speed with which AI gained popularity, the technological foundation can be traced back to a rough start in the 1950s/1960s [1]. Two so-called AI-Winters in the late 20th century [2] delayed its success until roughly 2012, when Deep Learning (DL)-based Neural Network (NN) architecture could successfully scale up their performance with new magnitudes of data, processing power, and novel algorithms finally becoming available. Since then, no domain has been left out and AI through all of its incarnations, such as Machine Learning (ML), Reinforcement Learning (RL), DL, and most recently Generative AI (GenAI), is finding its way into all domains, such as Computer Vision [3], Natural Language Processing [4–6], Medicine [7], Biology [8, 9], Physics [10, 11], and Geo Sciences [12–14], and as a result, also into the space domain [15, 16]. However, it wasn't until recent advances, most notably the discovery of Transformer Networks in 2017 [17] which unleashed effective LLMs, that made AI finally skyrocket.

It was this impulse that mobilized the, frankly speaking, old-school and slow-moving space sector to finally recognized how AI technology could be applied to solving challenges in all kinds of its fields, including but not limited to Earth Observation (EO) [18] and Remote Sensing [19], Fault-Detection, Fault-Isolation and Recovery Techniques (FDIR) [20], network and resource orchestration [21], and also satellite communications [22, 23], which is one of the focal points of this thesis.

Although applying AI to communications has become a very popular research topic by itself since its inception nine years ago [24], the vast majority of contributions is focused on devising novel architectures and algorithms [25], with only a fraction of effort being allocated in actually realizing these concepts on real-world hardware platforms. If so, the most popular solution to satisfy AI processing demands is to throw general-purpose capabilities, in most cases Graphics Processing Units (GPUs), at the problem. Although it is known that academia lacks access to domain-specific AI hardware technology for reasons such as funding [26], vendor-lock through industry-owned Intellectual Property (IP) [27], access restrictions to proprietary hardware technology [28], and a steep learning curve associated with proprietary tools and programming environments, this approach

inherently widens the gap between theoretical advancements in AI-driven communication systems and their practical deployments. This becomes clear when realizing that this firmly established field has so far only found very few applications in everyday technology.

Taking a look at the space domain, it is widely recognized that space technology, especially processor and memory systems, lags several years - ranging from five to 15 [29] - behind terrestrial technologies. The first AI-accelerators specifically targeted for space deployment are just starting to get a hold in the community. It is without doubt that a research strategy that is closely aligned with the industry and, hence, adapts domain-specific technology, is imperative to optimally realize the potential of Edge-AI on satellites.

The rest of this section is structured as follows: Before diving into the topic on a technical level, the special research setting this dissertation is situated in is described. Afterwards, the envisioned benefits of AI on satellites are outlined, which is followed by a down-selection and presentation of the key research problems addresses in this thesis.

1.2 Bidirectional Research Methodology in a Close Loop with Industry Partners

In this thesis we employ a special research methodology: In a very tight cooperation between the Technical University of Munich and a global player in the space domain, Airbus Defence and Space GmbH in Ottobrunn, Germany, we align the research objectives addressed by this thesis with realistic industry demands of the space sector and specifically streamline any hardware-related research with technology that is foreseen to find operation on near-future deployment scenarios. Together with the Department for Telecommunication and Navigation, whose mission it is to design the next-generation AI-enabled processing payloads, specifically for telecommunication satellites, a bidirectional research methodology is employed that enables a close feedback loop between research results and insights on practical implications. In this settings, both parties realize optimal benefits. Through a tight integration into the work environment, the author of this work is provided access on hard-to-get technologies (e.g., export-restricted Evaluation Boards (EVBs) on new processing chips or internal prototypes) and insights into business cases and environmental conditions, whereas the company benefits from inputs regarding future demands or service offerings, specifically related to AI, which their products must fulfill to be future-fit.

The following section gives an overview of the benefits AI is envisioned to have on satellite platforms, which is then followed by an outline of the research problem which is at the center of this dissertation: AI-augmented communication systems on satellites and their efficient implementation on future platforms. The chapter concludes by summarizing the contributions of this work and placing them within the thesis structure.

1.3 Benefits and Challenges of AI for Satellite Systems

Innovation in the space sector is typically initially driven by national space agencies, such as National Aeronautics and Space Agency (NASA), European Space Agency (ESA), German Aerospace Center (Deutsches Zentrum für Luft- und Raumfahrt) (DLR), etc., in the form of research grants. In the late 2010s, they announced substantial interest in the field of AI, both on ground and in space, which lead to the formation of multiple national research groups, such as NASA JPL's Artificial Intelligence Group and ESA's and German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz) (DFKI)'s joint ESA_LAB@DFKI group. Their sudden interest is driven by the transformation of the space domain by commercial actors, summarized under the umbrella term of NewSpace. Herein, three categories are the major drivers:

1. Big Data: The amount of data generated by satellites has become so massive that both conventional processing systems (which often depend on human participation) as well as data down-links cannot keep up. Efficient on-board data preselection and potential preprocessing via means of ML is required to tame the data flooding of future satellites. This category is generally applicable to all domains, although Earth Observation is currently envisioned to be the most relevant use case.

   As an example, ESA's Φ-Sat, which is the first AI-enabled Cubesat with on-board hardware acceleration (for more details on this topic ref. to Sub-sec. 3.1.2), optimizes the downlink capacity of high-quality EO data by filtering out taken images directly on-board if they do not satisfy certain quality metrics, e.g., discarding images with significant portions obscured by clouds [30].

2. Autonomous Operation: With the emergence of satellite mega constellations, the sheer number of satellites has risen exponentially to above 10.000 operational satellites today [31]. Classical satellite operation foresees a dedicated control team on ground that manages a variety of tasks, such as mission planning and scheduling, command and control, health and anomaly monitoring, orbit maintenance and collision avoidance, resource management, and much more, for a small number of satellites. This approach does not scale with the NewSpace reality, mainly being too cost-, time-, and human workforce-intensive. AI-based autonomous self-management of satellites is inevitable to allow the space sector to scale up.

   SpaceX revealed that their STARLINK constellation, which has grown to over 6400 satellites, has autonomously performed nearly 50.000 collision avoidance maneuvers via means of on-board AI within a six-month period, and that number is expected to double every six month [32].

3. System improvement: The third and most generic point envisions improvements for general satellite-related use-cases that can benefit in performance and reduce the satellite processing requirements. For instance, a reduction of the Bit Error Rate (BER) of communication systems, expanding Signal Intelligence (SIGINT) capabilities [33], or reducing algorithm complexity for computer-vision-based tasks when replaced with more suited ML-based solutions.

Despite the growing interest in onboard AI, current hardware accelerators are not yet seamlessly integrated into satellite systems. This limitation has led the industry to prioritize lightweight use cases, primarily situated at the Application Layer, due to the relative simplicity of interaction and modification at this level. In contrast, tasks that require deeply optimized and efficient implementations - such as those in the physical layer for signal processing - pose significant challenges. As a result, the industry's current focus lies on "higher-layer" applications, including tasks like network resource orchestration or traffic management, which can leverage the flexibility of software-defined solutions without demanding extensive system-level re-engineering. However, this leaves a vast potential for efficiently implementing AI-driven methods into lower layers yet to be claimed, which is at the center of this thesis.

Fig. 1.1 gives an overview of use cases that are envisioned to be addressed by on-board AI on the right side. Green-shaded items are addressed in this thesis, non-shaded items are given for completeness. Furthermore, it denotes the opportunities with respect to processing solutions on the left side.

The central challenge that needs to be solved is how to enable the processing requirements that come along with AI on a space-based platform. While this is addressed in detail in this thesis in Chap. 3, the unique characteristics of space, such as tight Size, Weight, and Power (SWaP) constraints paired with harsh environmental conditions like radiation, make this is a problem yet waiting to be solved.

The majority of recent hardware technology that is optimized for the inference of Neural Networks, also being referred to as AI hardware accelerators in this dissertation, is based on a digital data interface and digital processing internally [34]. Although digital technology has become the standard for terrestrial systems outside of computer history museums, the world looks very different for telecommunication satellites. The next section discusses the technological transition from analog to digital that satellites have to undertake to make on-board AI feasible.

1.4 Regenerative Satellite Payloads as Key-enabler for On-Board AI

After the launch of the first satellite called "Sputnik" in 1957 [35], early commercialization of satellite communications began in the 1960s. However, at this time, technology in space-flight primarily relied on analog components, as hardware for digital processing was not available in this decade. The primary principle of communication satellites, which is still widely used nowadays, relies on the so-called bent-pipe architecture [36]. Satellites in this configuration simply receive uplink signals from ground stations, amplify them, and transmit them to designated downlink areas. This approach requires minimal onboard processing and is very effective in distributing signals to large areas. However, inherent to the design were inefficiencies, such as poor spectral reuse and high dependence on terrestrial ground infrastructure for signal processing and routing.

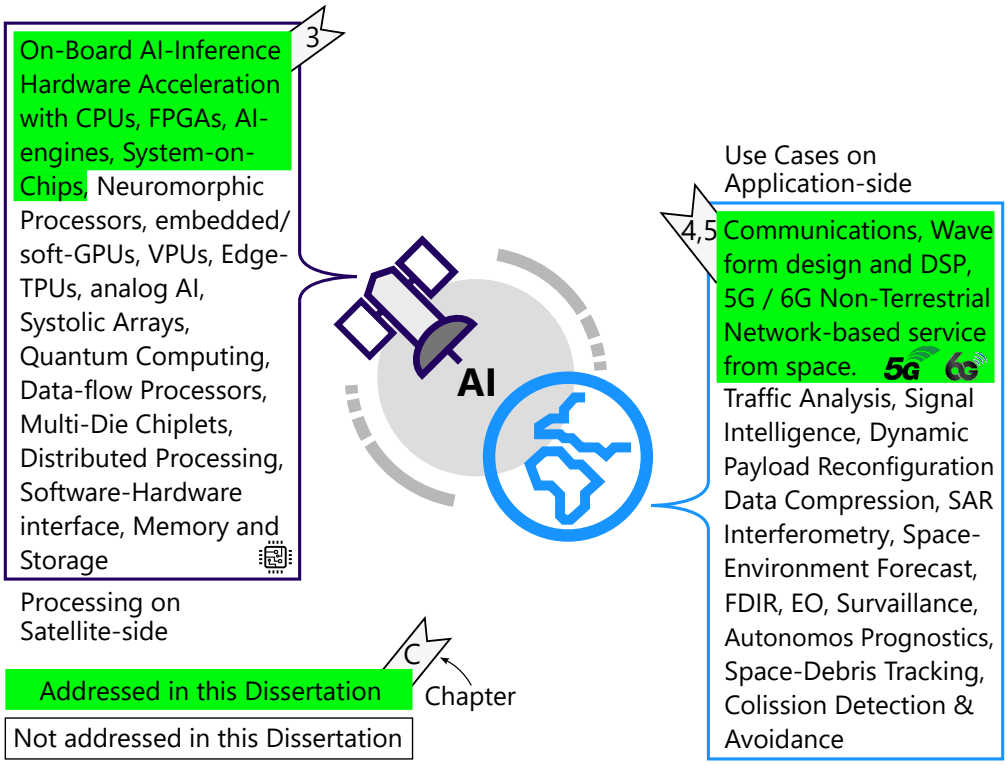


Figure 1.1: Visualization of Artificial Intelligence-related processing opportunities on the left side and use-cases on satellite payloads on the right side. Green-shaded topics are addressed in this dissertation. Abbreviations: Field Programmable Gate Array (FPGA), System-on-Chip (SoC), Graphics Processing Unit (GPU), Vector Processing Unit (VPU), Tensor Processing Unit (TPU), Synthetic Aperture Radar (SAR), Fault-Detection, Fault-Isolation and Recovery Techniques (FDIR), and Earth Observation (EO).

To address these inefficiencies, the concept of High-Throughput Satellites (HTSs) emerged. HTSs introduce the use of multiple narrowly focused spot beams, in contrast to traditional wide-area coverage. This innovation allowed spectral reuse, where the same frequency bands could be employed in non-overlapping beams, significantly increasing the total capacity of the satellite. HTS architectures were a major step forward in optimizing spectrum efficiency but were - and still are - reliant on the bent-pipe architecture.

As the demand for satellite communication grew, particularly with the advent of direct-to-home television services, mobile communications, broadband internet, and the higher integration level of non-terrestrial into terrestrial systems, the limitations of this model became increasingly apparent. This led to the development of regenerative satellite systems, a significant evolution in satellite architecture. In this configuration, regenerative satellites incorporate on-board processing capabilities on various levels, progressively enhancing their functionality.

The initial advancements centered around fundamental tasks, such as demodulation and modulation, which allowed satellites to process received signals and prepare them for retransmission. The next stage of development saw the integration of decoding and encoding capabilities. By regenerating the signals onboard, i.e., demodulating, decoding, re-encoding, and re-modulating them, regenerative satellites achieve a measurable Signal-to-Noise Ratio (SNR) improvement on the order of 2-4 dB compared to the transparent bent-pipe architecture. This gain primarily stems from the elimination of accumulated noise and interference across the uplink and downlink signal paths, the mitigation of signal impairments such as phase noise or synchronization-related perturbations, choosing optimal coding and modulation schemes depending on real-time channel properties, and by reducing cross-channel interference by applying digital filtering techniques.

Fig. 1.2 visualizes the shift from the analog to the digital domain in the physical layer. Depicted are two distinct components, Signal Conditioning, which comprises all signal transformations from digital data to the Radio Frequency (RF) baseband signal, and the Tuner, which comprises all signal transformations from base band signal to high-frequency RF signal. Both components are required when transmitting and receiving, however, data flows in different directions. It can be seen that the digital-analog boundary is shifting towards the RF side, meaning that digital systems are employed to solve tasks that were addressed before by analog components.

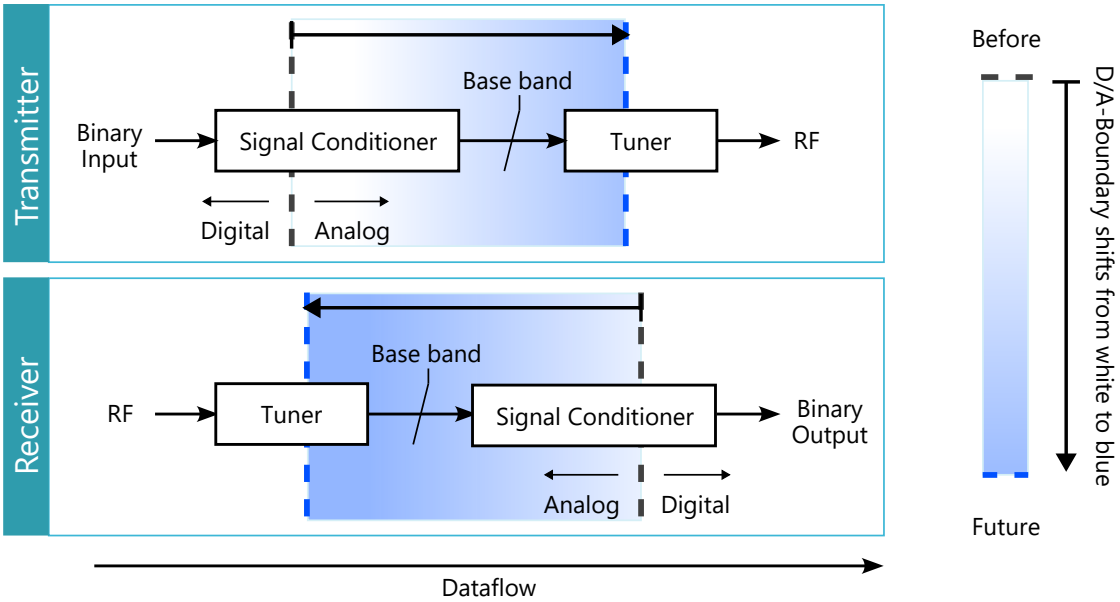


Figure 1.2: Depiction of the shift of the boundary between the digital and analog signal domain towards the analog side. Digital signal conditioning / processing is growing in importance and supersedes analog signal handling. Adapted from [37].

As onboard processing power continues to improve, satellites began to incorporate protocol switching and routing functionalities. These systems could now determine where and how to direct signals autonomously, optimizing bandwidth usage and reducing la-

tency.

Table 1.1 summarizes their key advantages and disadvantages between the transparent and regenerative satellite payload architectures.

Feature	**Transparent Architecture**	**Regenerative Architecture**
Complexity	✓ Low	✗ High
Cost	✓ Low development and manufacturing cost	✗ High development and manufacturing cost
Power Requirements	✓ Low - only signal amplification	✗ High - onboard processing and signal amplification
Flexibility	✗ Limited - Fixed frequency and beam configuration	✓ High - Dynamic spectrum allocation, (partly) reconfigurable signal conditioning
Latency	✗ High - Ground hop required	✓ Low - Onboard routing
Reliability	✓ High - Few points of failure	● Medium - Electronics failure and radiation risks
Mission Duration	● Medium - Non-upgradable	✓ Long - Highly upgradable

Table 1.1: Comparison of the advantages and disadvantages of bent-pipe vs. regenerative satellite architectures. Summarized from [35].

When stepping back and looking at telecommunication satellites from the perspective of the Open Systems Interconnection (OSI)-model, it can be seen, that the shift from transparent to regenerative satellites has started on the lowest layer, i.e., the physical layer, and is slowly working its way up to higher layers. This evolution, if continued, should ultimately culminate in digital processing capabilities at each layer. Circling back to our initial goal of integrating AI-based processing on-board satellites, it becomes clear that regenerative satellites form the foundation to realizing this goal. However, lots of open questions, such as where AI capabilities are needed the most, how to realize them (application-specific or generic), what hardware technology to use, how big the complexity is, and ultimately, if there is a real-life gain in using them, remain.

This dissertation addresses a subset of these questions and raises more, which are detailed in the following section.

1.5 Problem Statement and Research Questions

The primary goal of this dissertation is defined by the overarching research question

[RQ] *"How will the on-going paradigm shift in the field of communications fueled by the emergence of Artificial Intelligence impact telecommunication satellites and change the way how we design and interact with such systems in the future?"*

We aim to look at this question from multiple perspectives, such as operational modes and behavior (e.g., transparent vs. regenerative operation), functional and hardware capabilities (e.g., on-board AI-inference enabled by new technology), expected life-spans in the fast-paced AI environment, and new use cases enabled by AI. A specific focal point is set on the topic of communications.

To approach these topics systematically and allow for measurable results, we subdivide the overarching research question [RQ] into the following sub-research questions and practical problem statements:

First of all we focus on the technological aspect and raise the question

[RQ-1] *"How can AI-based processing capabilities be realized on regenerative satellite payloads under the constraints of the space environment?"*.

This refers to processing technology and a trade-off between Central Processing Unit (CPU), Field Programmable Gate Array (FPGA), embedded GPU, and Application-specific Integrated Circuit (ASIC)-based platforms, and joint combinations of these, framed as System-on-Chip (SoC).

We then concretize this question by considering the selected hardware platform based on our industry collaboration, which is AMD-Xilinx's VERSAL adaptive System-on-Chip platform, specifically focusing on the AI CORE programme line. We raise the question

[RQ-2] *"What realistic AI-related performance can be expected from the VERSAL AI Core aSoC?"*

with a focus on power consumption, NN architecture support, model size, and inference performance characteristics.

After answering in what capacity AI can be expected to be available in space, we take a look on practical use cases in the communications domain. First of all, we ask the general question

[RQ-3] *"In what form can AI-augmented communication systems be realized in space?"*,

which leads to the more concrete question

[RQ-4] *"How can DL be applied in the physical layer?"*,

with a specific focus on RF signal synchronization, addressing Carrier Frequency Offset (CFO), phase offset, and Sampling Time Offset (STO). Based on the proposed DL-augmented signal synchronization algorithm, we look into its efficient implementation on the named platform and ask

[RQ-5] *"Can a hybrid (classical and AI-based) signal processing chain be efficiently realized on next-generation platforms?"*.

Lastly, we place the results in a broader picture by asking

[RQ-6] *"How can a combination of AI and Fifth generation technology standard for cellular communications (5G) Non-Terrestrial Networks (NTNs) be implemented in concert on the same platform?"*.

1.6 Thesis Contributions

This publication-based dissertation is based on the following core publications:

1. [Pub4.A] **M. Petry**, A. Koch and M. Werner, *"Envisioning Physical Layer Flexibility Through the Power of Machine-Learning,"* 2023 IEEE Globecom Workshops (GC Wkshps), Kuala Lumpur, Malaysia, 2023, pp. 50-55, doi: 10.1109/GCWkshps58843.2023.10464871.
   Reference: [38] [Link]

2. [Pub3.A] **M. Petry**, G. Wuwer, A. Koch, P. Gest, M. Ghiglione and M. Werner, *"Accelerated Deep-Learning Inference on the Versal adaptive SoC in the Space Domain,"* 2023 European Data Handling & Data Processing Conference (EDHPC), Juan Les Pins, France, 2023, pp. 1-8, doi: 10.23919/EDHPC59100.2023.10396011.
   Reference: [39] [Link]

3. [Pub4.B] **M. Petry**, B. Parlier, A. Koch and M. Werner, *"Auto-Regressive RF Synchronization Using Deep-Learning,"* 2024 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN), Stockholm, Sweden, 2024, pp. 145-150, doi: 10.1109/ICMLCN59089.2024.10624754.
   Reference: [40] [Link]

4. [Pub4.C] **M. Petry**, A. Koch and M. Werner, *"Zero-Copy AI-augmented Signal Processing Pipeline on Heterogeneous Satellite Processors,"* 2024 58th Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, USA, 2024, pp. 354-361, doi: 10.1109/IEEECONF60004.2024.10943093.
   Reference: [41] [Link]

5. [Pub5.A] **M. Petry** and R. Mayr, *"AI-enabled 5G base-station with Hardware Acceleration for Non-Terrestrial Networks on a Space-Grade System-on-Chip,"* Proceedings of SPAICE2024: The First Joint European Space Agency / IAA Conference on AI in and for Space, 308–312, doi: 10.5281/zenodo.13885601.
   Reference: [42] [Link]

In the following we list the contributions of this thesis. For each contribution, the addressed research questions and, if applicable, the core publication presenting this contribution, are given.

- A thorough analysis of the AI processing capabilities of AMD-Xilinx's VERSAL AI Core adaptive SoC for popular NN architectures is presented on a system- and architectural-level [Pub3.A]. [RQ-1], [RQ-2]

- The concept of a generic but efficient NN-defined physical layer that exploits the advances of hardware-accelerated ML engines is presented. Functionality of a single-carrier, Bit-Interleaved Coded Modulation (BICM) communication system implementation is validated with Over-the-Air (OTA) communication using software-defined radios [Pub4.A]. [RQ-1], [RQ-3], [RQ-4]

- A DL-based signal processing architecture is devised that specifically addresses signal synchronization challenges, i.e., carrier frequency and phase offset as well as sample time offset compensation [Pub4.B]. [RQ-3], [RQ-4]
- Efficient implementation of hybrid (classic and ML-based) signal processing pipelines on AMD-Xilinx's VERSAL AI Core aSoC [Pub4.C]. [RQ-1], [RQ-2], [RQ-3], [RQ-4], [RQ-5]
- An AI-enabled 5G NTN base station hardware architecture optimized for AMD-Xilinx's VERSAL platform is presented. An initial performance analysis of physical layer computation offloading is presented [Pub5.A]. [RQ-1], [RQ-2], [RQ-6]
- A laboratory demonstrator for a full-fledged space-grade 5G base station (gNodeB) with Central Unit (CU) and Distributed Unit (DU) on-board, based on EVBs of the processing and RF hardware has been built. [RQ-6]
- The contributions are assessed from a holistic perspective and recommendations towards next steps for the space industry are provided. [RQ-1], [RQ-3], [RQ-6]
- Through a tight cooperation with industrial and governmental partners, specifically Airbus Defence and Space GmbH (ADS), the European Space Agency (ESA), and the German aerospace research and technology center (DLR), the AI capabilities of next-generation regenerative telecommunication satellite payload processors have been impacted and shaped by the work performed in this thesis. Expertise gained during this research has been actively transferred to ADS and brought practical deployment of AI-based algorithms on the edge in space a step forward towards becoming reality.

Fig. 1.3 gives a visual overview of how the chapters are related with the core publications, and which research questions the core publications address.

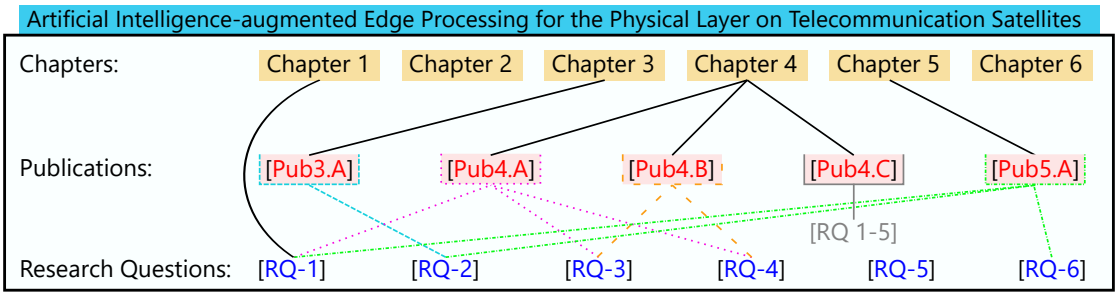


Figure 1.3: Visualization of the connection between chapters, research questions, and publications.

1.7 Thesis Organization

The remainder of this thesis is organized as follows:

To begin, Chapter 2 lays a common foundation by revisiting the fundamental principles of communications and information theory, Deep-Learning, and how it is applied to communications. Moreover, recent advances and related work towards the State-of-the-Art are discussed.

Chapter 3 dives into AI-based technologies in space and situates the thorough investigation of AMD-Xilinx's VERSAL AI Core platform in [Pub3.A], which is at the core of later presented AI-augmented signal processing concepts and implementations, in the context of high-throughput telecommunication spacecrafts.

Chapter 4 picks up AI-augmented communication systems and introduces two system concepts building upon Deep-Learning-based signal processing. First, the idea of a generic and flexible Neural Network-defined physical layer and its implementation on the VERSAL, presented in [Pub4.A], is discussed. This approach is then contrasted to more task-specific approaches, such as an Artificial Intelligence-based solution to the niche challenge of signal synchronization, presented in [Pub4.B]. A discussion trading off pros and cons of efficient hardware-tailored implementations such as [Pub4.C] concludes this chapter.

Chapter 5 takes a look beyond the horizon and assesses the implications of the growing popularity of Artificial Intelligence on satellite platforms in a bigger picture, for instance as part of in-orbit base stations for next-generation 5G/Sixth generation technology standard for cellular communications (6G) Non-Terrestrial Networks, as presented in [Pub5.A]. Furthermore, based on already standardized and future functionality specified by 3rd Generation Partnership Project (3GPP), we conclude this chapter by analyzing the compatibility and future-proofness of the presented concepts with normative activities.

Chapter 6 concludes this thesis.

2 Background

To make this thesis insightful and understandable for readers who are not familiar with wireless communications, non-terrestrial networks, or their applications to DL, in this chapter we revisit the foundations upon which this doctoral thesis rests. After introducing the essence of wireless communications and diving into the aspects of software-defined radio and signal synchronization, we summarize the essentials of Deep-Learning and review the current state-of-the-research in its applications to communications.

2.1 A Primer on Wireless Communication Systems

Communication systems are the backbone of modern connectivity, enabling seamless exchange of information across vast distances. The design of modern communication systems is a delicate balancing act, driven by the need to achieve optimal performance under practical constraints. Each design choice ripples through the system, influencing performance metrics such as data rate, error probability, and power efficiency. Hereby, this interconnectedness follows a fundamental principle: There is no free lunch - improving one parameter invariably impacts others. In wireless communications, this principle is particularly acute, where the environment, regulatory limits, and finite resources like power and bandwidth impose stringent challenges on system design.

Understanding the trade-offs and interdependencies among key system parameters has been a cornerstone of communication theory since its inception. Theoretical breakthroughs, such as Nyquist's sampling theorem [43] and Shannon's channel capacity [44], provide the mathematical foundations for analyzing and optimizing communication systems. These theories allow researchers to systematically tune practical performance metrics such as BER, Block Error Rate (BLER), and spectral efficiency towards achieving reliable and efficient information transfer for various scenarios.

This chapter revisits the theoretical underpinnings of wireless communication and connects them to practical system performance. After starting with fundamental concepts such as the relationship between the time-domain and the frequency-domain, noise, and the notion of a channel, we explore how bandwidth and power interplay together and define the limits of reliable communication by introducing capacity. Afterwards, key performance metrics and tools to quantify system performance used throughout this work are introduced. We then pivot into communication paradigms from a processing perspective and show how novel platforms, such as Software-defined Radios (SDRs), enable End-to-End (E2E) designed physical layers. We conclude by a brief review of synchronization techniques, one of the core topics of this dissertation.

2.1.1 Nyquist Meets Shannon - The Intuition Behind Communications

Time- and Frequency-Domain Relationship: In 1924, Harry Nyquist, by then a young engineer at the American Telephone and Telegraph Company (AT&T), worked on analyzing and improving the speed of electrical telegraphy, a point-to-point text messaging system. One of his greatest contributions is the formulation of the sampling theorem which states that for sampling a bandwidth-limited signal in the time-domain, one must sample it with at least twice the frequency of its bandwidth B, i.e., $f_S \geq 2B$ to capture its complete information and allow exact reconstruction. While a lower sampling rate leads to a loss of information, also known under the effect of aliasing, a higher sampling rate cannot yield any more information in the captured signal. Furthermore, he expands the theorem by stating that if the time signal consists of $M = 2^k$ discrete levels, which all represent a symbol holding k bits of data, then the maximum data rate carried by such signal can be calculated as

$$\text{Maximum data rate} = 2B \log_2 M \text{ bit/s [43].} \tag{2.1}$$

From the above equation it becomes evident that a signal's representation in the time-domain and the frequency-domain somehow stand in a relationship. We now want to build an intuitive understanding of how this relationship behaves.

Building on Nyquist's idea from above, the most primitive time signal $x(t)$ with M discrete levels (here: amplitudes) one could come up with is shown in Fig. 2.1a. For each symbol to transmit, a rectangular pulse scaled to the symbol's appropriate amplitude is placed in the signal for the symbol duration T_S, which is equal for all symbols. One might think now: "This was easy, all my information is embedded in the signal!" - but there is a big caveat. Nyquist's sampling theorem requires a signal to be bandwidth-limited in the frequency-domain to be able to correctly reconstruct it without a loss of information. However, our simple-looking signal actually exhibits an infinite bandwidth, as follows from Fourier analysis, also denoted by $\mathcal{F}$:

$$\mathcal{F}\left(x(t)\right) = \int_{-\infty}^{\infty} x(t) e^{-j\omega t}\, dt = X(\omega) \text{ [45]}, \tag{2.2}$$

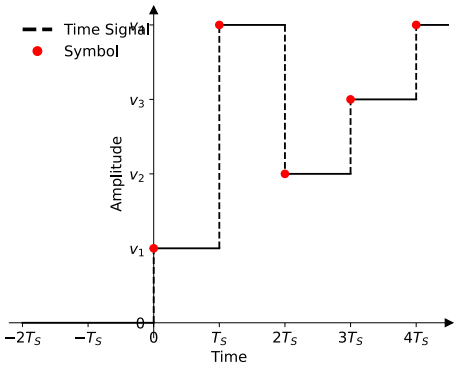
where $X(\omega)$ is the frequency-domain representation of $x(t)$, and $\omega = 2\pi f$ with f being the frequency in Hz. For simplicity, we use the symbols ○—● and ●—○ to represent the relationship between the time- (empty circle) and frequency-domain representation (filled circle) like $x(t)$ ○—● $X(f)$. For our signal made out of rectangular base-pulses $rect(t)$, given as

$$\text{rect}(t) = \begin{cases} A & \text{for } -T_S/2 \leq t \leq T_S/2, \\ 0 & \text{otherwise.} \end{cases}, \tag{2.3}$$

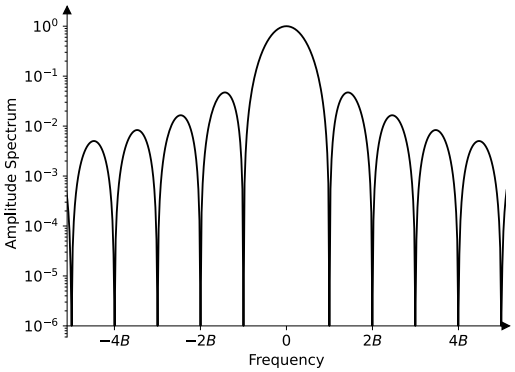
with A being the symbol's amplitude, the Fourier transform yields $\text{Rect}(f)$ as

$$\text{Rect}(f) = A \cdot T_S \cdot \text{sinc}\left(T_S \cdot f\right), \tag{2.4}$$

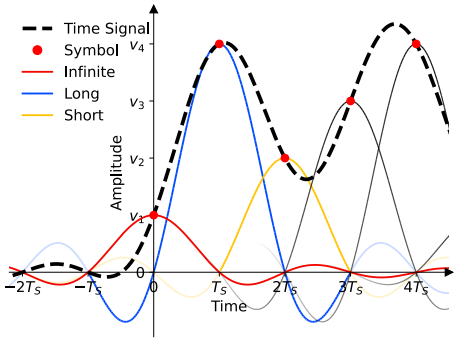
revealing $\text{rect}(t)$'s infinite bandwidth as a direct consequence of the sinc-function's nature of never completely decaying to zero. Fig. 2.1b shows the qualitative amplitude spectrum of this signal.



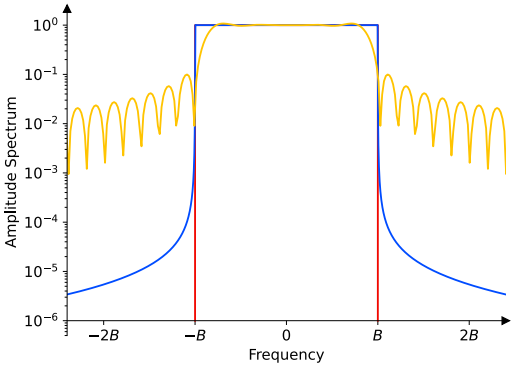
(a) Time-series visualization of a signal without pulse shaping.



(b) Normalized qualitative amplitude spectrum of the signal in Fig. 2.1a



(c) Time-series visualization of a signal (black-dashed) transmitting symbols (red circles) at discrete values. The underlying Nyquist pulses are depicted with different lengths.



(d) Normalized qualitative amplitude spectra of the Nyquist pulses in Fig. 2.1c with different lengths.

Figure 2.1: Simple visualization of the connection between a time-series signal and its spectral decomposition.

In the search for an alternative time signal with a more confined spectral energy, the **principle of duality** of the time-frequency relationship, a core intuition of signal processing, comes in very handy. In simple terms, it says that when swapping time- and frequency-domain representations, their respective characteristics also swap. With slight abuse of notation for simplicity, instead of using the rect(t)-function as base-pulse for the time-signal, we can use Rect(f)$|f = t$ as time function, i.e., a sinc-based pulse, and we obtain its original time-representation, rect(t)$|t = f$ in the frequency-domain. Fig. 2.1c shows the new time signal (dashed black line) based on sinc-pulses (colored and grey lines) that carry the same symbols as in our original signal. Although the new spectrum is now perfectly confined within a rectangle, as shown in Fig. 2.1d (red line), we have the problem of an indefinitely long and, hence, non-causal base-pulse (infinite extension of the sinc(t)-function) in the time domain, making its exact realization impossible. Remedy is always found in a trade-off. Here, a practical solution is to truncate or taper the pulse length, as indicated by the blue and yellow pulses, which leads to a small "leakage" of spectral energy into adjacent spectrum, also shown in 2.1d, which would need to be addressed.

Such a problem perfectly represents the challenge of designing communication systems, as tuning one knob in a sea of variables sets of a chain reaction of corollaries, as alluded to in Sec. 2.1, that need to be understood and controlled. A concrete corollary of the above solution is the need to lower the symbol rate, and hence data rate, to down-scale the occupied bandwidth to satisfy original constraints. To increase data rate again, a clever engineer might now come up with the idea of increasing the number of discrete signal levels M, potentially to infinity. This leads us to the overarching counterforce in wireless communications, noise, which is now introduced from the perspective of a channel.

Channel: Instead of looking at system performance and capacity from the perspective of the signal, it is much more insightful and intuitive to look at it from the concept of a channel. While we usually have full control in the selection and design of our signals, we are constrained by the characteristics of a channel, which in wireless communications often vary with time, space, and frequency. Among various perturbations, such as attenuation, multi-path propagation, or a non-flat frequency response (fading), noise is the most fundamental factor. Although not physically-grounded, another limiting factor, specifically in satellite communications, can be regulatory constrains, which dictate permissible transmission power, frequency bands, and bandwidth usage.

For characterizing information flow through a channel, the notion of entropy is suitable. Entropy is a measure of the randomness or uncertainty of a random variable, which is equivalent to information-content or -density of a signal for communication problems. An information source or signal can be formalized under the concept of random variables, since their (in a specific order) received realizations/samples, for instance bits (selection of 0 or 1) or discrete symbols (such as letters A-Z), provide information.

For an information source formulated as random variable X outputting discrete sym-

bols its entropy is calculated as

$$H(X) = \sum_{i=0}^{N-1} P(x_i) \cdot I(x_i), \tag{2.5}$$

where $H(X)$ denotes the average entropy, $P(x_i)$ denotes the probability of symbol x_i being selected, and $I(x_i)$ denotes its respective information density. If chosen $I(x_i) = \log_2(P(x_i))$, than the entropy's unit is in Bits per symbol. For communication through a channel, Fig. 2.2 visualizes the transformation of signal entropy from source signal X to the received signal Y at the sink with a Sankey-Diagram.

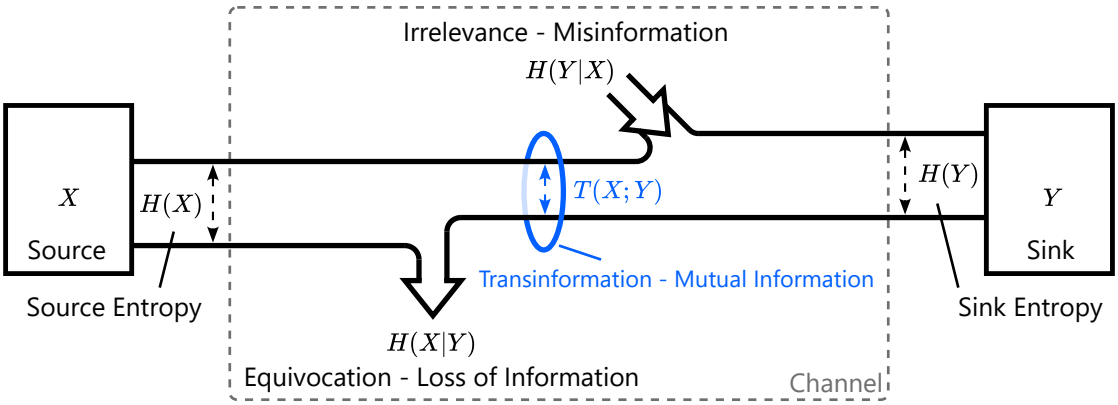


Figure 2.2: Visualization of the change of entropy from signal source to sink through a lossy channel.

Three metrics are of high relevance: The conditional entropy $H(X|Y)$ states the uncertainty in the original signal X when Y has been received and is known. It can be thought of as the lost part of information of X that the receiver never observes. It effectively results in the "receiver's problem" of inferring X from Y. For a channel characterized by discrete symbols and memoryless transitions with limited input and output alphabets, the conditional entropy is given as

$$H(X|Y) = -\sum_i \sum_j P(x_i, y_i) \log_2 P(x_i|y_j), \tag{2.6}$$

with $P(x_i, y_i)$ being the joint probability of $X = x_i$ and $Y = y_i$, and $P(x_i|y_j)$ being the conditioned probability of $X = x_i$ given $Y = y_j$ is known.

The transinformation $T(X;Y)$ denotes the part of information of the transmitted signal that can be successfully conveyed through the channel and is contained in Y. It is hence the mutual information between X and Y, making it the defining property regarding calculating information transfer capacity of a channel, as discussed shortly.

Lastly, the conditional entropy $H(Y|X)$ denotes the uncertainty in the received signal Y when X is known. It can be interpreted as noise or randomness injected into Y by the channel.

In summary, the entropies have the following intuitive meaning:

- $H(X)$: Useful information density of the transmitted signal
- $H(Y)$: Total information (including misinformation) density of the received signal
- $T(X;Y)$: Information density of the useful part of information that gets through
- $H(Y|X)$: The added noise/uncertainty by the channel
- $H(X|Y)$: The confusion/uncertainty of the receiver in inferring X from Y.

Coming back to our original goal of performance characterization of wireless systems, the mutual information $T(X;Y)$ lets us determine the channel's **capacity** of information transfer, which is discussed in the following.

Channel Capacity: The fundamental metric which gives a measure of how much information we can maximally transfer through the channel is the channel capacity. We want to emphasize, that this capacity describes *reliable* transmission, which practically boils down to being close to error-free[1]. The channel capacity, also known as Shannon capacity, is given by optimizing the input probability distribution $P(X)$ to maximize mutual information $T(X;Y)$, given as

$$C = \max_P(X)\ T(X;Y), \tag{2.7}$$

and is measured in bits per second. For a noiseless, bandwidth-limited channel, this can be trivially computed as shown in Eq. 2.1.

For a noisy channel, on first glance it is non-intuitive to be even able guarantee a certain capacity due to the unpredictability of noise. However, techniques like Forward Error Correction (FEC) that can resolve errors on the Receiver (RX)-side by embedding redundancy in the signal on the Transmitter (TX)-side, allow to get ahead of the channel's unpredictability within safety margins (typically upper BER limits) determined by statistical analysis. In 1948, Shannon derived the general capacity limit for a bandwidth- and noise-limited channel, also know as the Shannon-Hartley theorem [44]:

$$C = B\ \log_2\left(1 + \frac{P_{\text{Signal}}}{P_{\text{Noise}}}\right) \text{bits/s}. \tag{2.8}$$

It says, that the capacity for reliable transmission through such a channel depends solely on its bandwidth B and the ratio between signal and noise power, P_{Signal} and P_{Noise}, respectively, also denoted SNR, at the output of the channel. Furthermore, it presupposes a zero-mean Gaussian amplitude distribution for the noise. This type of noise is also referred to as Additive White Gaussian Noise (AWGN), since it adds "on top" of the signal, given as

$$y = x + n; n \sim \mathcal{CN}\left(0, 2{\sigma_{\mathcal{N}}}^2\right). \tag{2.9}$$

[1] Formally, a transmission is reliable if the error rate can be reduced to an arbitrary low value. Practically this would require infinitely long code-words, and hence, infinite latency until information is available at the receiver. As a compromise, a non-zero error-rate is traded-off for a limited code-word length.

In the base band x, y, and n are complex valued, with n being a realization of the complex random variable $\mathcal{N}$ having a Gaussian probability density function for both of its real-valued dimensions, given as

$$p_{\mathcal{N}}(n) = \frac{1}{\sqrt{2\pi{\sigma_{\mathcal{N}}}^2}} \exp \frac{-(n-\mu_{\mathcal{N}})^2}{2{\sigma_{\mathcal{N}}}^2}. \tag{2.10}$$

$\mu_{\mathcal{N}} = 0$ is its mean value and ${\sigma_{\mathcal{N}}}^2$ is its variance.

Although Eq. 2.8 defines the limits on reliable communication, it does not reveal how to design a communication system to achieve those limits, i.e., selecting modulation and coding scheme, pulse shaping, etc. However, it does reveal two fundamental regions of operation: The bandwidth- and the power-limited region.

By defining R_b as the gross bit rate (including the possible overhead of coding schemes), the signal power can be expressed using the Energy per bit E_b in the received signal as

$$P_{\text{Signal}} = E_b \cdot R_b. \tag{2.11}$$

Furthermore, the noise power can be expressed as

$$P_{\text{Noise}} = B \cdot N_0, \tag{2.12}$$

with N_0 being the noise spectral density, a measure of the noise power per unit bandwidth.

When substituting Eq. 2.11 and 2.12 into 2.8 and normalizing by B, we get the transcendental equation

$$\frac{R_b}{B} < \log_2\left(1 + \frac{E_b}{N_0}\frac{R_b}{B}\right), \tag{2.13}$$

where $\frac{R_b}{B}$ is the spectral efficiency in bits per second per unit bandwidth (Hz) and $\frac{E_b}{N_0}$ is the SNR normalized over bandwidth. Assuming operation at the Shannon limit, the gross bit rate is equal to the channel capacity, $R_b = C$, and we can define $\eta = \frac{R_b}{B} = \frac{C}{B}$ as the maximum achievable spectral efficiency.

After transformations of Eq. 2.13 we can compute a required E_b/N_0 to achieve certain spectral efficiency η as

$$\frac{E_b}{N_0} = (2^\eta - 1) \cdot \frac{1}{\eta}. \tag{2.14}$$

Typically it is more interesting to know the spectral efficiency for a certain normalized SNR, so the reversal function of Eq. 2.14 is visualized in Fig. 2.3. The area of practical operation is white, and the unreachable area (due to the Shannon limit) is red.

We want to summarize the Shannon capacity's key insights in the following:

- Bandwidth-limited regime: When operating at very high SNRs, spectral efficiency saturates due to the logarithmic nature of capacity growth with power. This indicates diminishing returns for increasing signal power and capacity is hence limited by the available bandwidth.

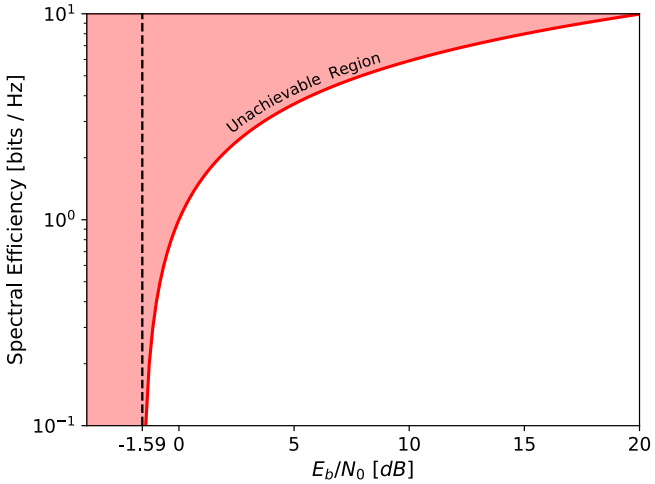


Figure 2.3: Spectral efficiency as a function of E_b/N_0.

- Power-limited regime: When operating at very small SNRs, the capacity scales linearly with power and is hence limited by the available power.
- For a fixed channel capacity, increasing bandwidth can reduce the required $\frac{E_b}{N_0}$, shifting the system into a power-efficient regime. Conversely, increasing $\frac{E_b}{N_0}$ allows operation in a bandwidth-efficient regime. These trade-offs are central to designing communication systems based on available resources and constraints.
- The theoretical minimum normalized SNR $\frac{E_b}{N_0} = -1.59$ dB represents the ideal baseline for power-efficient but reliable communication, assuming infinite bandwidth is available. Any practical system operates above this threshold, yet it serves as a critical benchmark for evaluating system designs.

For space-based communication, both power- and bandwidth-limited regions are used, although for very different use-cases. For deep-space communication, e.g., direct communication between Earth and Mars, spectrum availability is abundant due to only few communicating parties being active while the received SNR is very low due to an enormous path loss and limited transmission power for devices on Mars or in its orbit. Hence, bandwidth is sacrificed for power efficiency and capacity is limited by power [35]. Communication systems within the vicinity of Earth, such as ground-space links for satellite constellations, are heavily bandwidth-limited due to a very crowded spectrum caused by millions of ground-based user terminals and thousands of operating satellites.

While the Shannon capacity defines a theoretical upper limit, practical systems operate below this limit due to non-idealities like suboptimal Modulation and Coding Schemes (MCSs), channel estimation errors, and hardware imperfections. To analyze practical communication systems, the next subsection introduces essential tools.

2.1.2 Key Metrics for Quantifying System Performance

The performance of practical communication systems can be described well with key performance metrics such as (normalized) SNR and spectral efficiency (introduced above), as well as BER, and BLER. BER quantifies the fraction of received bits decoded incorrectly, offering insight into the reliability of a system. Closely related to BER is the BLER, which represents the likelihood of an entire block, typically a codeword, being in error, a more practical measure for coded systems.

Spectral efficiency measures the amount of information successfully transmitted per unit bandwidth, typically expressed in bits per channel use (bpcu). It balances the trade-off between bandwidth utilization and error performance, becoming a crucial metric for system optimization.

In the following we introduce two visualizations that depict the relationship between the above metrics.

Bit Error Rate (BER) performance as a function of Signal-to-Noise Ratio (SNR): An important graph depicts the BER as a function of the SNR. Typically, bandwidth-normalized SNR metrics, e.g., E_S/N_0 or E_b/N_0, are used. It describes the relationship between the communication system's level of reliability, here represented through the proxy BER, and the required amount of signal power at the receiver to achieve such level. It is an ideal tool to quantify performance of various system properties, such as the waveform configuration (modulation and coding schemes), environmental conditions (channel model and noise types), and algorithmic performance (e.g., decoding performance). Fig. 2.4 compares the performance of the coding schemes used to protect data transmission in 5G, i.e., Low-density Parity-check Codes (LDPC), to other classic schemes, like convolutional codes and Turbo codes. Each curve represents a specific MCS combination. In addition to these coding schemes, each coding scheme is paired with both 16-Quadrature Amplitude Modulation (QAM) (thick, solid lines) and 256-QAM modulation (thin, dash-dotted lines). All codes shown exhibit a code rate of 0.5 and have a codeword length of 128. The only exception is the long LDPC which has a codeword length of 16000 for illustration purposes.

First of all, the two performance regions, defined by the choice of modulation scheme, are clearly visible. Furthermore, it can be seen that some codes perform better than others. In general, LDPC (brown, red, and orange lines) outperform Turbo codes (green), which outperform convolutional codes (blue). The performance of codes also depends on the codeword length, which can be well illustrated with LDPC, which performs particularly well for very long codeword lengths, as can be seen with the brown line. Lastly, the type of decoding algorithm can also impact performance as shown by the comparison of Ordered Statistics Decoding (which provides close to maximum-likelihood performance) to Belief Propagation Decoding for the short-length LDPC, given by the red and orange lines, respectively.

Spectral efficiency as a function of Signal-to-Noise Ratio (SNR): The spectral efficiency vs. SNR graph evaluates the effectiveness of communication systems in utilizing available

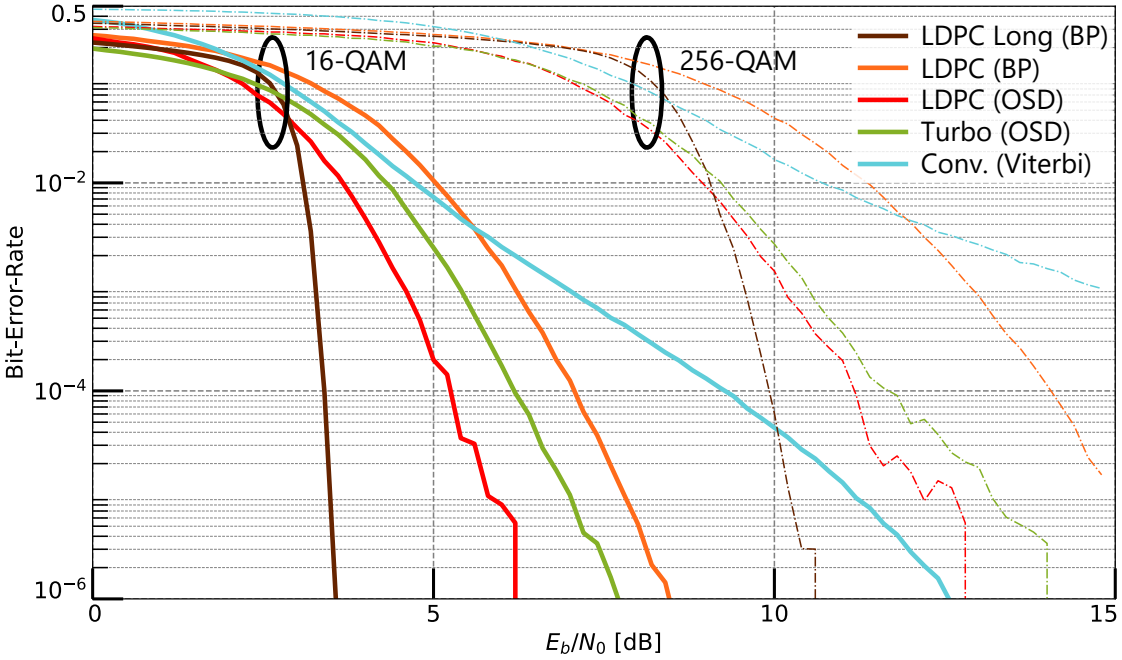


Figure 2.4: Comparison of modern 5G coding schemes (LDPC) to a Turbo and convolutional code. All codes have a code rate of 0.5 and codeword length of 128, except the long LDPC, which has a codeword length of 16000. The graph shows the BER over normalized SNR for the different MCSs with 16-QAM and 256-QAM, and illustrates the difference between Ordered Statistics Decoding (OSD) and Belief Propagation (BP) for selected examples. The codes have been simulated with the Sionna Python library [46].

bandwidth. Spectral efficiency, either normalized for unit bandwidth (Hz) or channel use (cu), represents the rate at which information can be reliably transmitted over a channel for a given level of signal quality. On the x-axis, the normalized SNR quantifies the trade-off between power and bandwidth efficiency, while the y-axis shows the achievable spectral efficiency. Fig. 2.5 shows the spectral efficiency of 16-QAM (blue) and 16-Phase-shift Keying (green) modulation schemes with demodulation based on soft-information (solid line) or hard-information (dash-dotted line) under AWGN (thick) and Rayleigh fading (thin) channel conditions.

The graph provides two critical insights:

- Performance Gap to the Shannon Limit: The thick red line and thin purple line at the top-left of Fig. 2.5 represent the theoretical maximum efficiency for an AWGN and a Rayleigh channel, respectively, as given by the Shannon limit. Practical modulation schemes like QAM combined with LDPC can approach this limit within a few dBs, with higher-order modulations performing better at high SNRs. However, the gap to the Shannon limit highlights the losses due to suboptimalities as mentioned above.

- Region of Operation: Different MCS combinations are optimized for specific SNR ranges. For example, higher-order modulation schemes like 256-QAM provide high spectral efficiency but require high SNR, making them ideal for bandwidth-limited scenarios. Conversely, lower-order modulation schemes like Quadrature Phase Shift Keying (QPSK) offer better performance at lower SNRs, suited for power-limited channels or challenging environments.

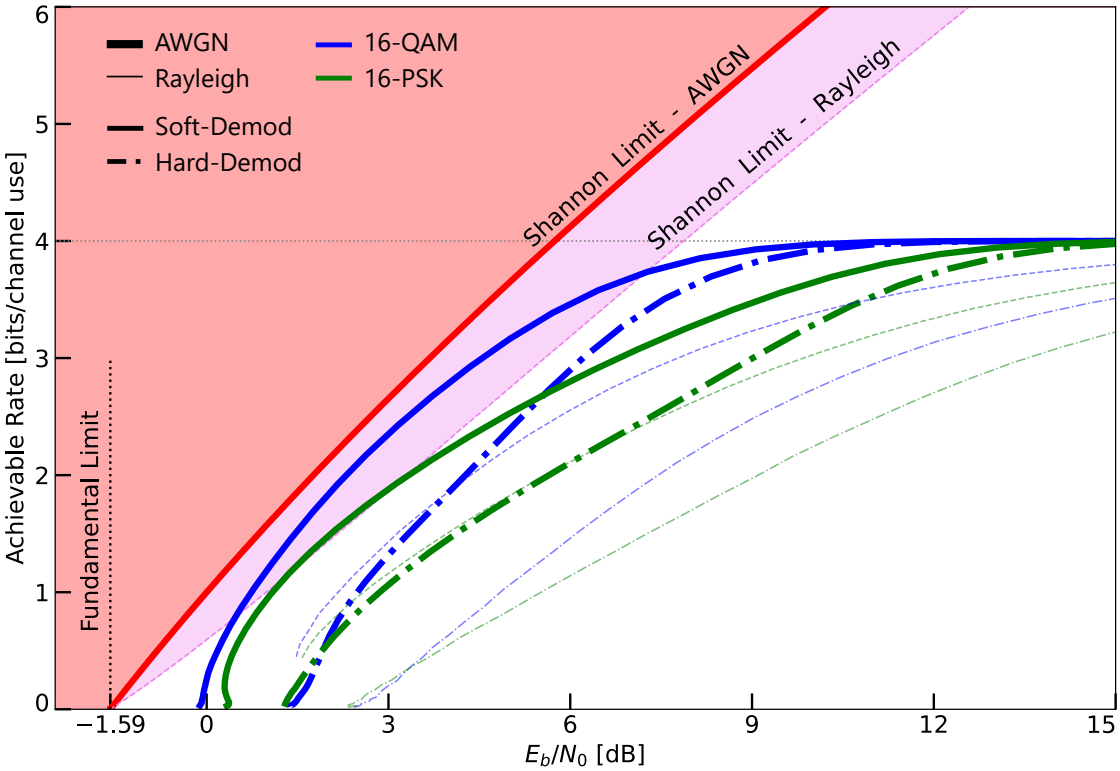


Figure 2.5: Comparison of the spectral efficiency of various modulation schemes and their gap to the Shannon limit under AWGN and Rayleigh channel conditions, as a function of the normalized SNR. Depicted is the Shannon Limit for AWGN (red, thick) and Rayleigh fading (purple, thin) channel conditions, and the spectral efficiency of 16-QAM (blue) and 16-PSK (green) modulation schemes using soft- (solid) or hard-information (dashed-dotted) for decoding for both channel conditions.

In modern systems, MCSs and other parameters are adapted on-the-fly to always operate as close as possible to the Shannon limit. This flexibility is a result of the paradigm shift from analog- to digital-based signal conditioning over the last decades. The following subsection presents this and discusses how this dissertation builds on this concept.

2.1.3 Communication Paradigms

The evolution of communication systems from analog to digital paradigms has been a cornerstone in modern telecommunications, as introduced in Sec. 1.4. This transformation is characterized by shifts in how information is represented and processed before the transmission over physical channels. A critical aspect of this evolution is the placement of the Digital-Analog boundary, which inherently delineates where signal processing transitions from digital to analog domains. The placement of this boundary has significant implications for system design, performance, and complexity. Before diving into its effect, we want to review the key components of communication systems, to better understand how they are affected. Afterwards, we explore the up- and down-sides of digital implementations and introduce the concept of software-defined radio.

Key components of communication systems and their categorization: Common signal transformation components that can be found in most communication systems can be grouped into three categories, as shown in Fig. 2.6. These categories and most important components are listed in the following.

1. Data Transformation: The input to this category is binary data, representing the payload information. The output is typically M-ary symbols, which are discrete values used in modulation schemes. Key processes include:
   - Encryption: Secures the binary payload against unauthorized access by applying cryptographic algorithms.
   - Cyclic Redundancy Check (CRC) Append: Adds cyclic redundancy check bits to facilitate error detection.
   - Coder: Introduces redundancy for error correction, using techniques such as convolutional coding or LDPC.
   - Rate Matcher: Adapts the coded data to match the transmission rate, often through puncturing or repetition.
   - Interleaver: Rearranges data bits to mitigate burst errors by distributing them across time or frequency.
   - Scrambler: Randomizes bit sequences to minimize predictable patterns and improve spectral properties.
   - Modulator: Maps binary data to M-ary symbols, such as in QPSK or 16-QAM, which are ready for waveform transformation.

   Although the above operations relate specifically to the TX-side, data transformations exhibit a duality between TX and RX, with the receiver typically implementing the respective reverse operations.

2. Waveform Transformation: The input to this category are M-ary symbols, and the output are baseband IQ samples. These represent the in-phase (I) and quadrature (Q) components of the signal. Processes in this stage shape the signal for robust transmission. Key components include:

- Layer Mapper: Maps symbols to different logical or physical layers in systems like Multiple-Input Multiple-Output (MIMO) to support spatial multiplexing.
- MIMO Encoder: Encodes the signal across multiple antennas to exploit spatial diversity and improve reliability.
- Precoder: Adjusts the signal phases and amplitudes across antennas to optimize performance under channel conditions.
- Beamformer: Directs signal energy spatially to improve link quality and reduce interference.
- Prefix/Suffix Append: Adds cyclic prefixes or guard intervals to mitigate inter-symbol interference.
- Window/Filter: Shapes the signal's spectral properties to meet bandwidth and interference requirements. On the RX-side, operations like signal synchronization and equalization counteract channel impairments, recovering IQ samples for further decoding.
- Signal Synchronization: Compensate for perturbations in the captured signal, such as frequency offsets in the carrier and in the sampling hardware. Typically more relevant on the RX-side.

3. Spectral Transformation: The input to this category is baseband IQ samples, and the output is an RF signal, ready for transmission over the physical channel. In the category, components convert the baseband samples to their final form for RF propagation. Key components include:
   - IQ Modulation: Combines the I and Q components with a carrier signal to create a modulated baseband signal.
   - Additional Filtering: Shapes the signal to remove out-of-band components and ensure spectral compliance.
   - Upconversion: Shifts the baseband signal to the desired RF carrier frequency using a Local Oscillator (LO).
   - Amplification: Boosts the RF signal to achieve the required transmission power. On the RX-side, a technique called Automatic Gain Control (AGC) amplifies the incoming signal to exploit the full range of Analog-to-Digital Converters (ADCs).

Systems with different Digital-Analog boundaries: We now want to outline the major communication paradigms and describe at which stage current systems, specifically satellite communication systems, are at. Three major system architectures are sketched in Fig. 2.6. An initial observation is that placing the boundary near the binary interface heavily burdens analog hardware with complex tasks, while placing it near the RF side enables to solve these tasks with digital means. Based on the above introduced

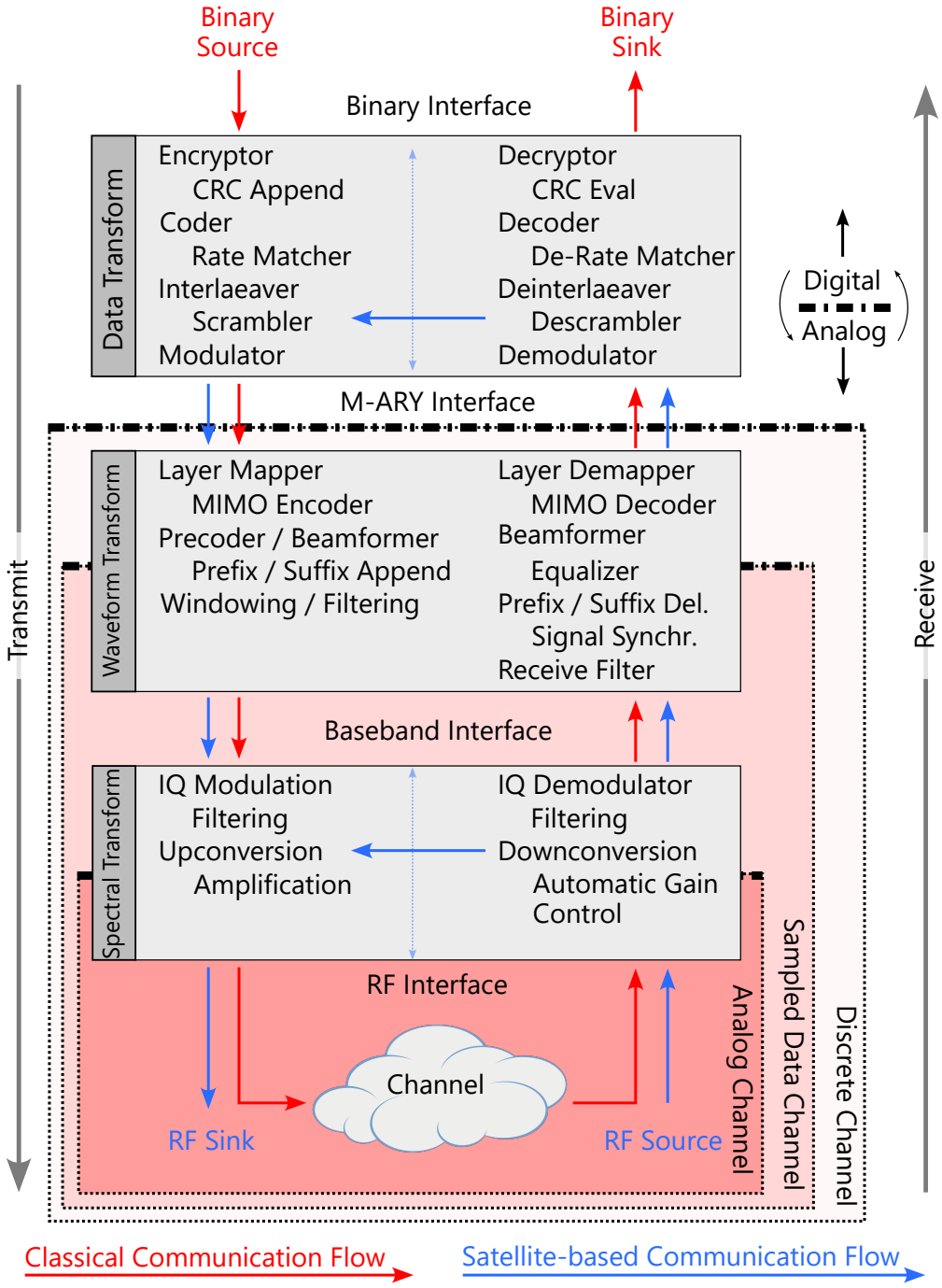


Figure 2.6: Visualization of the evolution of communication paradigms with respect to signal processing, highlighting the analog-digital boundary and communication component placement for the discrete, sampled data, and analog channel.

Metric	Discrete Ch.	Sampled Data Ch.	Analog Ch.
Flexibility	Medium	High	Very High
Performance	Medium	High	Very High
Hardware Efficiency	Low	Balanced	High
Power Efficiency	Low	Moderate	High
SWaP	High	Moderate	Low
Cost	Low	Medium	High

Table 2.1: Comparison of communication architectures based on Digital-Analog boundaries.

categories, system architectures can be grouped in three categories based on the relative position of their D/A boundary to (or in) the three transformation categories.

For this, we look at the word *channel* from the perspective of signal transformations, not from an RF perspective. Everything beyond the D/A boundary, i.e., on the analog-side, is what the digital side interfaces with. Depending on the boundary's position, we can categorize this into the discrete channel, sampled data channel, and analog channel, ordered from a small to large degree of digitization. Fig. 2.6 visualizes the relationship between the boundaries and their channel interpretation.

In the discrete channel architecture, the D/A-boundary is at the M-ary symbol interface, leaving all waveform and spectral transformations to analog hardware. While dynamic MCSs are supported, bandwidth-adaptability is limited by the rigid nature of the succinct analog filtering. Performance and flexibility are constrained by hardware precision, making this architecture largely obsolete today.

For the sampled data channel, the D/A-boundary moves into the waveform transformation stage, adding dynamic spatial spectrum access and resource management capabilities with techniques such as adaptive hybrid beamforming and antenna diversity schemes, such as MIMO systems.

The highest level of digitization is implemented with the analog channel architecture, which also performs IQ modulation and most spectral transformations digitally, leaving only upconversion and purely analog components for analog implementation.

Table 2.1 presents the architectures' capabilities in summary.

Classical communication devices have a binary and an RF-interface, and information flows from the binary to RF-interface for transmitters and vise-versa for receivers, as indicated by the red arrows in Fig. 2.6. To place satellites into this framework, we must slightly adapt the perspective. While Fig. 2.6 depicts a complete communication system, a satellite typically performs only a selection of the listed operations, as it is not explicitly part of the TX or RX endpoints, but an intermediary located in between. It either mediates communication by simply relaying analog signals, called bent-pipe architecture, or by serving as a gateway with some processing capabilities, e.g., to modify waveform and protocols between endpoints, called regenerative architecture. It's input and output interface is both RF data (depicted by the blue arrows), contrary to classic communication devices that have a binary data and RF interface for input and output, or vise-versa, for transmitters and receivers, respectively.

Bent-pipe and regenerative systems primarily differ in the satellite's capabilities to apply modifications to the waveform and the underlying data. In Fig. 2.6, this translates to the depth of the return-path (depicted by the horizontal blue arrow). Systems with few capabilities, such as transparent satellites, have an early return-path, typically near the down- and up-conversion and filtering stage. Systems with more capabilities, such as regenerative satellites, start with modulation/demodulation and can extend to coding/decoding, encryption/decryption, and even modifications to the user data itself, such as protocol transformations.

For both architectures the Digital/Analog-boundary is typically drawn at the same position, i.e., inside the spectral transform category. Additional functionality which is provided in regenerative satellite payloads is implemented digitally [35].

The advancement in digital processing capabilities enables a new radio technology that is primarily defined by algorithms implemented in software, not analog hardware, giving it the name Software-defined Radio (SDR). The next subsection introduces this concept.

2.1.4 Software-defined Radio

SDR represents a paradigm shift in communication systems by replacing fixed-function hardware with a reconfigurable architecture. At its core, an SDR consists of two main components, as depicted in Fig. 2.7: A flexible RF front-end and a dynamic, fully digital back-end. The RF front-end, which is based on analog components like mixers, filters, and amplifiers that are typically realized using Integrated Circuits (ICs), supports real-time reconfiguration of key communication parameters, including carrier frequency, sample rate, and filter bandwidth. This enables it to capture and transmit a wide range of waveforms, enabling seamless adaptation to diverse operating conditions. The connection between the RF front-end and the digital back-end is established via ADCs and Digital-to-Analog Converters (DACs). Additional control signals enable reconfiguration of the RF front-end through the back end.

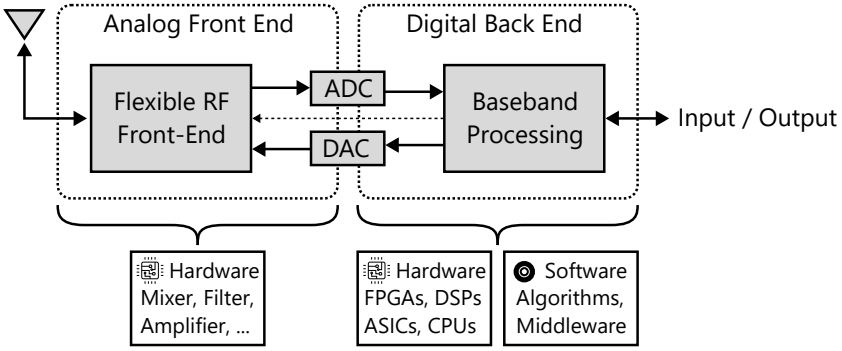


Figure 2.7: Simple Visualization of the concept of a Software-defined Radio.

The digital back-end, often referred to as the baseband processing unit, is where SDR truly shines. Implemented using FPGAs, Digital Signal Processors (DSPs), ASICs,

or general-purpose CPUs, it can execute user-defined software and essentially perform all tasks listed in Sub-sec. 2.1.3 in the digital domain, including but not limited to modulation, coding, and signal analysis. Unlike classical radios, where these operations are either hardwired or functionality is frozen in silicon, SDR's fully digital processing allows for on-the-fly updates and real-time adaptation to changing system requirements. The digital back-end is typically connected to a host-device which exchange either the raw information bits or even processed packets following some protocol, or the raw or modified I/Q-samples coming directly from the RF front-end. This setup adds further flexibility to the system, giving low-end hosts the option to receive data in the form of an "end-product", while high-end hosts can post-process data as required.

These capabilities make SDRs very attractive to the satellite communications domain, where the increased densification of spectrum necessitates flexible communication concepts with (partial) in-orbit reconfiguration to fill the available spectrum most efficiently. Moreover, it makes cost-efficient and rapid prototyping for wireless researchers that are working with novel algorithms and varying waveforms easily accessible.

For both of the above reasons, this dissertation makes heavy use of the SDR-concept, for instance for evaluating the OTA performance of AI-designed communication systems in [Pub4.A], implementing a hybrid AI and classically-based signal processing pipeline on next-generation telecommunication satellite payloads in [Pub4.C], and building a space-grade 5G base station in [Pub5.A]. The two utilized setups are depicted in Fig. 2.8. In both setups AMD-Xilinx's VERSAL AI Core SoC is used as baseband processing back end and host, and either the Ettus Research (NI) X300 Universal Software Radio Peripheral (USRP) or Analog Device's AD9082 chip is used as RF front end. While the AD9082 is essentially an RF front end only with negligible on-chip signal processing capabilities and, hence, requires an external digital back end, the X300 also offers integrated generic processing capabilities with its user-controlled FPGA framework RF-Network-on-Chip (NoC). This example shows that a SDR platform can take on multiple forms and is not defined by strict hardware-software combinations, but rather by the underlying concept.

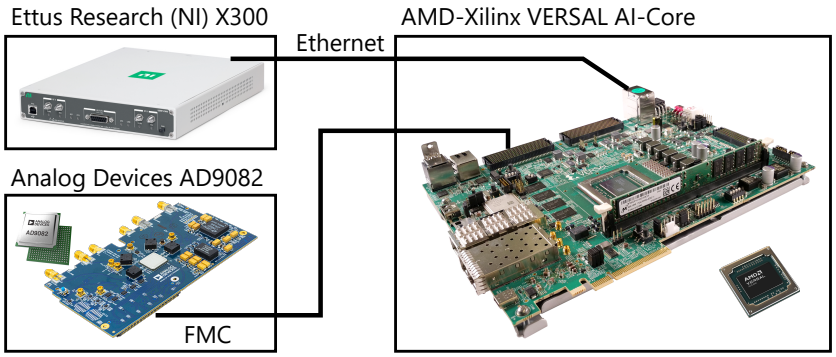


Figure 2.8: Radio Frequency and processing components used in this dissertation that together resemble a Software-defined Radio platform.

Among other things, an aspect that SDRs drive forward is physical layer research. In the next section we look at the physical layer and explore E2E-optimized processing

concepts, which are also applied in this dissertation.

2.1.5 From a Modular to End-To-End Optimized Physical Layer

The physical layer in conventional communication systems, no matter whether they are state-of-the-art or outdated, is fundamentally designed in a modular way [47]. All of its components, such as those discussed in Sub-sec. 2.1.3, are specifically designed to solve their particular task within a certain set of boundaries, such as computational complexity limits, theoretical assumptions (e.g., channel model, average or peak power constraints, etc.), or heuristics [48–50], while having a clear input- and output-interface. This gives rise to the main benefits of interchangeability and re-usability of different blocks in different communication systems depending on the practical scenario or environmental condition at hand [24].

Although this approach has been adapted in virtually all communication systems and standards today, it comes with two distinct disadvantages: Performance and complexity. If we look at performance from the perspective of the system's gap to the Shannon Capacity (ref. to Sub-sec. 2.1.1), it is obvious that some combinations of blocks perform "better" than others. This becomes easy to see on the example of pairs of modulation and coding schemes. Choosing the best possible option that closes the gap to the Shannon Capacity has been a continuous research problem. Furthermore, additional "practical" issues, such as achieving signal synchronization (introduced in the next Sub-sec.) between TX and RX, are very hard to consider during the design and optimization phase of the individual blocks, and are hence solved by allocating additional resources (e.g., signal energy) afterwards to the problem, degrading performance further.

Increasing E2E performance typically comes at the cost of higher system complexity. This complexity is evident in modern satellite-based communication systems like Digital-Video-Broadcast (DVB)-S2X. For example, DVB-S2X employs a wide array of Modulation and Coding Schemes, ranging from QPSK to 256-Amplitude and Phase-shift Keying (APSK), paired with advanced FEC schemes, such as concatenated Bose-Chaudhuri-Hocquenghem (Code) (BCH) and LDPC codes [51]. While these mechanisms come close to theoretical maximum spectral efficiency, their intricate interactions and varying operational modes significantly complicate system analysis and optimization. Furthermore, the modularity introduces layers of dependencies that must be carefully managed, particularly under varying channel conditions.

A promising alternative is to design systems that unify traditionally separate blocks into a single processing step, directly optimizing their collective function. For instance, joint MCSs, such as Trellis-Coded Modulation, have demonstrated how combining these operations can outperform sequential designs by leveraging shared optimization criteria [52]. Similarly, unified symbol mapping and precoding for MIMO systems can yield significant gains in spectral efficiency and robustness compared to their individually optimized counterparts.

Emerging NN-based autoencoders take this concept further by treating the entire physical layer as a single trainable model. These autoencoders can be optimized E2E to directly learn a mapping from input bits to received bits while incorporating various

components, such as modulation, coding, channel equalization, and even synchronization into a unified framework. This approach holds immense promise for future communication systems and its potential is explored in this dissertation. It's AI-aspect is discussed in detail in Sec. 2.3.

2.1.6 Signal Synchronization

In this last part of our primer on wireless communications we want to review the fundamentals of signal synchronization. In the following we identify the sources of this problem and boil them down to four critical perturbations they exert on the communicated waveform. Then we look at algorithmic solutions that have been devised to address this issue with a focus on analog and digital hardware implementations and relate that to the shift in processing domains as discussed in Sub-sec. 2.1.3. Finally, this section concludes by providing an idea on alternative solutions that are later discussed in this dissertation.

Synchronization in the RF-domain refers to the broad task of achieving a temporal and spectral alignment of two or more spatially distributed communicating end-points with respect to the transmitted signals. It is one of the most crucial issues in a communication system, particularly on the RX-side, since most components rely on a synchronized signal [53]. This is the case for both wired and wireless systems. It is necessary for multiple reasons:

- Mobility: In scenarios where the transmitter and receiver are not stationary relative to one another, a Doppler effect on the received signal can be noticed. Practically this leads to a dynamic change in the received signal's carrier frequency and phase on the receiver's RF front end [Pub4.B].
- Hardware imperfections: Distributed systems rely on a local clock generation using analog components such as LOs, Phase-Locked Loops (PLLs), frequency multipliers and dividers, mixers, etc., which are not exactly synchronized in frequency and phase due to thermal drifts, manufacturing tolerances, phase jitter, etc. [54].
- Propagation effects: Effects in the transmission medium can impact the signals timing and spectral characteristics. For instance, multi-path propagation leads to different arrival times of the same radio wave with different power values, also known as Inter-symbol Interference (ISI), necessitating adaptive interpretation of the received data. Moreover, a static TX and RX distance causes a propagation delay and a phase shift due to finite propagation speed. Lastly, atmospheric conditions, such as ionospheric scintillation or tropospheric effects can cause additional waveform distortion [54].

The consequences of these effects can be summarized in four distinct perturbations imprinted in the received signal [Pub4.B][55]:

- Sampling Time Offset (STO):
  Description: Misalignment in time of the sampling instances of the ADC and the

"pulse-center" of the received I/Q-symbols in the receiver. Static STO is typically caused by a random phase offset of the ADC's sample clock source and dynamic STO is caused by a Sampling Frequency Offset (SFO) (see below).
Consequence: Samples with an offset to the pulse-center have a lower SNR compared to samples at the pulse-center and are subject to distortion through ISI. In the constellation diagram, the symbols appear to be more noisy.

- Constant Phase Offset (CPO):
  Description: Quasi-static complex phase rotation within the observed I/Q-samples. It is caused by the non-synchronized phase of the transmitter's and receiver's carrier-frequency clock source, e.g., their LOs and/or PLLs, and by the path distance in relation to the wavelength of the transmitted signal.
  Consequence: If not corrected, the constellation diagram of the received I/Q symbols exhibits a constant phase rotation that leads to decision errors in the demodulator.

- Carrier Frequency Offset (CFO):
  Description: Frequency difference between the carrier frequencies of TX and RX. CFO is primarily caused by manufacturing differences and instabilities in the LOs and by mobility scenarios (Doppler effect).
  Consequence: The received I/Q samples are modulated with a time-dependent phase offset, making the constellation diagram appear to be rotating with a frequency equal to CFO, leading to decision errors in the demodulator if not corrected.

- Sampling Frequency Offset (SFO):
  Description: Fractional mismatch between the sampling clock sources of TX's DAC and RX' ADC. It is primarily caused by the limited precision and instabilities of LOs.
  Consequence: On the RX-side, the fraction between the ADC's sampling rate and TX's symbol rate is not an integer anymore, but slightly higher or less in practice. This means that fewer or more information is conveyed within the same amount of samples which would lead to either a loss of information or "hallucination" of non-transmitted data if not addressed using DSP means.

In this dissertation, we restrict the scope to STO, CPO, and CFO. In the following we outline the two main concepts of analog and digitally-based signal synchronization, and briefly cover modern digital techniques.

2.1.6.1 Analog vs. Digital Concept and Techniques

Signal synchronization can be implemented using either analog or digital approaches. Both methods aim to achieve accurate signal recovery, but their implementation strategies differ fundamentally, particularly in terms of hardware complexity, flexibility, and signal processing domain.

In an analog synchronization system such as depicted in Fig. 2.9, the signal is processed through a series of servo loops. Carrier frequency synchronization (red-shaded

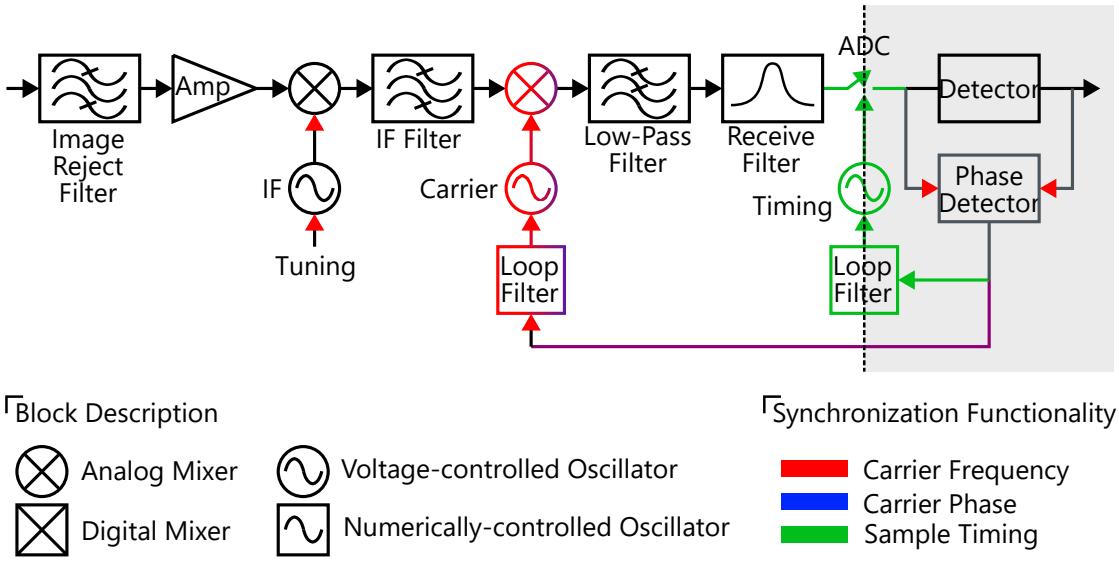


Figure 2.9: Receiver architecture concept with synchronization based on analog components. Adapted with modifications from [56].

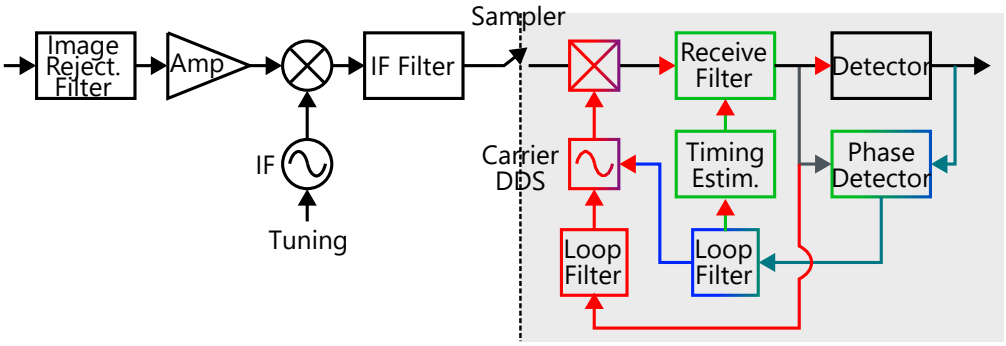


Figure 2.10: Receiver architecture concept with synchronization based on digital signal conditioning. Adapted with modifications from [56].

components) is achieved by controlling the LO using a PLL, which aligns the frequency and phase (blue-shaded components) of the received signal to the transmitter. A loop filter and phase detector dynamically adjust the LO with the goal of minimizing CFO and CPO. Simultaneously, a timing servo loop (green-shaded components) governs the sampler's clock, actively adjusting the sampling instances to minimize STO. The tight coupling between the analog LO and the sampling clock ensures that phase and timing synchronization are achieved in real-time with high accuracy. However, the performance of this approach heavily relies on precise analog components, which are susceptible to noise, drift, and manufacturing tolerances.

In contrast, a digital synchronization concept as shown in Fig. 2.10 processes the signal after an initial fixed-frequency sampling stage. For instance, in a superheterodyne receiver architecture, the received signal is amplified, filtered, and down-converted to an Intermediate Frequency (IF), then digitized using a fixed-rate ADC. Subsequent synchronization tasks are performed entirely in the digital domain. CFO and CPO synchronization are handled by mixing the signal with a Numerically-Controlled Oscillator (NCO), implemented in a Direct Digital Synthesis (DDS) architecture. Timing synchronization is also achieved via digital means, such as interpolation-based methods or poly-phase filter banks [57], which enables precise timing offset correction without the need for an adaptive sampling clock. This architecture allows for greater flexibility and adaptability, as digital processing can leverage different algorithms under varying channel conditions.

While analog synchronization excels in low-latency applications and minimizes reliance on high-speed digital hardware, it lacks the adaptability and scalability of digital methods. Digital synchronization, on the other hand, demands higher computational resources. The choice between these approaches depends on system constraints, including power consumption, latency requirements, and processing complexity. Modern systems, particularly in advanced communication standards such as 5G and Wireless Local Area Network (WLAN) favor digital synchronization due to its compatibility with modern transceiver architectures and waveforms, such as Orthogonal Frequency-Division Multiplexing (OFDM).

2.1.6.2 Review of Algorithms and Approaches

Synchronization techniques can be broadly categorized based on the energy source that carries the synchronization information, as given below [58].

Energy from Excess Bandwidth: Commonly used in broadcast systems, this approach extracts statistical information from the signal's excess bandwidth. While energy-efficient, it requires processing numerous symbols to integrate signal energy until reaching certainty in its estimation. It is unsuitable for peer-to-peer or burst communication scenarios.

Preamble/Pilot-Aided Synchronization: This method embeds synchronization energy in transmitted preambles or pilot signals, enabling fast acquisition in dynamic envi-

ronments. It is particularly effective for burst communication and mobile applications, where quick synchronization is essential.

Carrier Recovery Methods: Carrier recovery eliminates distortions such as phase rotation and frequency offsets. Key methods include:

- Band-Edge Filters: Extracts frequency offset by analyzing the energy imbalance between upper and lower signal bands. This non-data-aided method is simple and works well in systems with sufficient excess bandwidth but may struggle in spectrally efficient designs [59].
- Frequency-Locked PLL: Combines a frequency discriminator with a phase-locked loop to dynamically track CFO. Its strength lies in its ability to maintain lock over slow variations with low complexity [60].
- Pilot-Aided Recovery: Embeds known symbols (pilots) in the signal to directly estimate carrier frequency and phase offsets. This method is highly accurate and widely used in OFDM systems like 5G and WLAN standards but adds overhead to the transmission [58].
- Sweep-based recovery: An older approach where frequency and phase are incrementally adjusted until lock is achieved. While simple, it performs poorly in dynamic environments due to slow acquisition and limited tracking capabilities.

Timing Recovery Methods: Timing recovery aligns sampling instants with transmitted symbols, avoiding ISI. Prominent techniques include:

- Control clock at sampler: Directly adjusts the receiver's clock based on timing errors. While accurate, it requires tight integration with hardware and is less flexible compared to digital methods.
- Interpolation before matched filtering: Dynamically adjusts sampling instances by interpolating signal samples before matched filtering. This method decouples timing correction from the hardware clock, providing flexibility and software-driven adaptability.
- Gardner Loop: Calculates timing errors using mid-symbol samples, which are less sensitive to noise compared to early-late gates. This method performs robustly even under low SNR conditions and is widely used in practical systems [61].
- Mueller and Müller algorithm: A sample-by-sample timing recovery algorithm that estimates timing errors by interpolating between received samples. It maintains an internal state to track timing offsets, ensuring continuous adjustment. The key advantage lies in its simplicity and effectiveness without requiring preambles or pilot symbols [62].

- Maximum Likelihood Methods: The Early-Late Gate method compares early and late samples relative to symbol centers. When combined in a single filter in a derivative approach, it offers high precision at small computational complexity.

Although synchronization is essential for a functioning communication system, the topic has received very few attention in academia over the last decades and has been neglected in research. In Chap. 4 this dissertation presents a new DL-based synchronization approach that is deeply embedded in a End-to-End-designed communication system. In this approach, the modulation scheme itself is adapted in a training process to be slightly unsymmetrical to give a Neural Network the ability to determine the absolute phase offset without relying on pilots.

2.2 A Deep Learning Primer

Before learning how DL can be applied to communication systems, we want to provide a compact review of some of the fundamental concepts behind DL which this dissertation builds on. After placing DL within the scope of AI-based techniques and exploring its deployment schemes, we review the concept of automatic differentiation which is at the core of dynamic research with domain-specific operations. We conclude by introducing the notion of "old" vs. "new" ML and discuss which regime AI in space is situated in. For additional information we refer to exhaustive literature, such as [63], [64].

2.2.1 Contextualization of DL within the Domain of AI

DL is a subset of techniques within the discipline of representation learning, which is itself a subset of the field of AI. Although ML, DL, and sometimes AI are often used synonymously in literature (also in this dissertation), ML is, strictly speaking, a different subset of AI. It enables systems to learn patterns and make decisions or predictions from data without being explicitly programmed [65, 66]. While this definition keeps the role of learned components shallow, DL specifically refers to a chain of learned representations that is nested using multiple layers of abstraction. Both ML and DL rely on algorithms and statistical models to iteratively improve performance on specific tasks through experience (data). Their relation is visualized in Fig. 2.11.

2.2.2 Scopes of Learning

We can distinguish between the different learning and deployment scenarios of DL algorithms. [67] identified the following three different facets of learning:

- *A priori* learning (offline learning): Replacement of hand-designed rule-based algorithms by pre-trained NNs. Training of NNs is performed offline on a captured or generated dataset using a training-optimized hardware platform (e.g., GPU-based system). The ML algorithm is then deployed in production by replacing the classical algorithm with the pre-trained NN. The NN can in principle benefit the system in multiple metrics, such as reduced computational complexity, decision

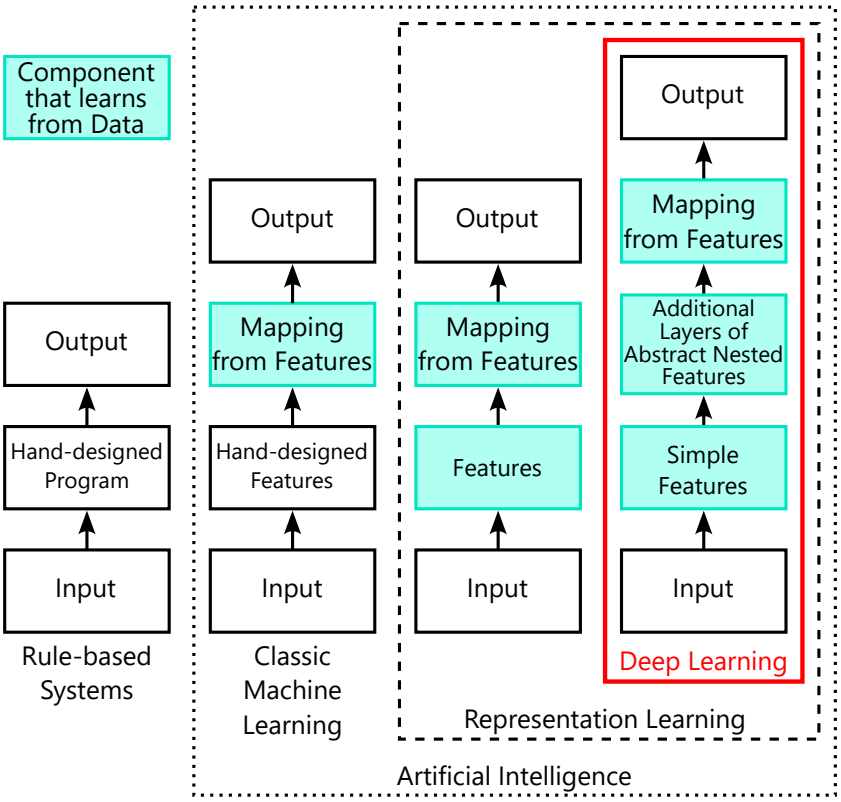


Figure 2.11: Contextualization of DL in the field of AI. Adapted from [63].

latency, and system performance (e.g., accuracy, Bit Error Rate, ...), where the later explicitly depends on the similarity between the training dataset and the real environment.

- *In situ* learning (online learning): Implementing online (at run-time) training to the production environment. A (pre-trained) ML algorithm is deployed in the production system which continuously refines its input-output relation to the current run-time environment by updating its trainable components periodically. This enables adapting to effects which can and/or are not accurately modeled in the offline dataset, and allows on-the-fly adaptation to situational changes, such as a blurred camera in autonomously-guided vehicles. In situ learning can limit the maximum allowed model size as training capacity is limited on the inference system.
- *Ad temporarium* learning (design-time learning): Benefit from insights/outcomes of powerful training algorithms by transferring information (e.g., learned parameters) from the learning process back to the classical algorithm. In contrast to both prior strategies, the trained NN does not replace the classic algorithm, but information from it, such as meta-information or trained parameters/weights, are incorporated into the classical algorithm.

2.2.3 Old and New Machine Learning

The field of ML has seen significant evolution over the past decades, transitioning from conventional architectures to increasingly sophisticated models. This section explores the distinction between what has been termed as “old” and “new” AI paradigms.

Old Machine Learning - From Neurons to Convolutional Neural Networks: The ”old” machine learning paradigm is characterized by foundational models such as feed-forward dense layers, convolutional neural networks (CNNs), and recurrent neural networks (RNNs). These architectures form the backbone of many classical AI applications:

- Feed-forward Dense Layers (FFDL) / Multi-Layer Perceptron (MLP): These are the simplest form of neural networks, consisting of fully connected layers where each neuron is linked to every neuron in the subsequent layer, followed by a non-linear activation function and an individual bias per neuron. They are widely used for structured data tasks, such as tabular data classification or regression.
- Convolutional Neural Network (CNN): Designed for spatial data processing, CNNs excel in computer-vision-related tasks, like image recognition and object detection. Their use of convolutional layers enables hierarchical feature extraction.
- Recurrent Neural Network (RNN): RNNs are specifically tailored for sequential data, such as time series or natural language, by using feedback connections that allow temporal dependencies to be modeled [68]. Variants like Long short-term Memory (LSTM) [69] or xLSTM [70] networks improve their ability to capture long-range dependencies.

These models dominated the field of AI, specifically DL, for a long time until applications in Natrual Language Processing (NLP) reached performance limits that could not be overcome with existing architectures [17]. This gave rise to "new" concepts which are outlined in the following.

New Machine Learning - From Transformers to Large Language Models: The "new" ML paradigm builds on the above introduced foundational concepts but introduces architectures that are substantially more complex and versatile. Key advancements include:

- Transformer Architectures: Originally introduced for NLP tasks, transformers have revolutionized the field by enabling efficient handling of long-range dependencies in sequential data through mechanisms like self-attention. They power LLMs and are increasingly adapted for computer vision and other domains.
- Advanced Recurrent Architectures: Attention-augmented RNNs extend the capabilities of traditional RNNs, improving their performance and stability on complex temporal tasks.
- Generative Models (GenAI): Generative AI models, including Generative Adversarial Networks (GANs) [71] and diffusion models, focus on creating new data that resembles the training data. These models are pivotal in applications such as image synthesis, content generation, and simulation. Recent advances integrate transformer architectures into generative modeling.
- Other Specialized Architectures: Emerging models often incorporate hybrid approaches, combining convolutional and attention-based mechanisms, or introducing entirely new frameworks tailored to specific applications, such as Graph Neural Networks (GNNs).

With these new architectures it is not uncommon that the number of trainable parameters can be orders of magnitudes greater compared to classic CNN-based models. Hence, the necessity of significant computational resources create a dependence on cutting-edge hardware technology like GPUs, Tensor Processing Units (TPUs), and custom-silicon-based AI accelerators. For these reasons, the adoption of AI techniques in the space-environment, specifically for Edge-AI on-board spacecrafts, is constrained by the lack of such advanced hardware (for details ref. to Sub-sec. 3.1.2) and mainly capitalizes on techniques of the "old" regime.

2.2.4 Automatic Differentiation

ML-models are trained by minimizing a user-defined loss function that mathematically formulates the "good-ness" or accuracy of the model's output. The model's performance, and hence the loss value, depends on the model's trainable parameters. In the training stage, these parameters are iteratively adjusted with the goal of minimizing the loss function. The predominant methods for calculating these adjustments are based on variations of Stochastic Gradient Descent (SGD), which connects each parameter with

the loss value via its gradient. This sub-section elaborates on a method called Automatic Differentiation (AutoDiff) which forms the cornerstone of modern achievements in AI.

Approaches to differentiation: The accurate and computational efficient computation of the gradients is key in training neural networks. Four different methods to calculate them can be applied [72]:

- Manual differentiation: Although analytically precise, manual differentiation is impractical for large nested expressions, prone to errors, scales very slowly, and cannot adapt dynamically as the expression evolves.
- Numerical differentiation: While conceptually simple, accuracy suffers from truncation and round-off errors making it unreliable for fine-grained training with precision. Furthermore, its computational cost scales poorly with dimensionality of the model.
- Symbolic differentiation: Based on automated algebraic manipulation of mathematical expressions, symbolic differentiation offers exact derivatives. However, as expressions increase in complexity, "expression swell" leads to a rapidly falling computational efficiency inhibiting the use of extensive branching, looping, or recursion.
- Automatic Differentiation: AutoDiff provides both exact and computationally efficient gradients by formulating a computational graph in the forward pass followed by the computation of individual gradients for each operation, which are then accumulated in the backward pass by applying the chain rule. It delivers a numerical evaluation as opposed to a derivative expression.

The concept of AutoDiff: AutoDiff leverages the fact that the computation through NNs is inherently sequential, no matter how complex the architecture is, and can be decomposed into a sequence of elementary operations (addition, multiplication, logarithm, trigonometric, etc.). In the dataflow programming paradigm such computation is represented by directed graphs [73, 74], where nodes and edges represent operations and data dependencies, respectively. AutoDiff knows two modes of operation, forward and backward, which corresponds to the order of how the chain rule is applied. While the forward mode is more computationally efficient for a fewer number of parameters than outputs, reverse mode applies to the opposite. Since ML-models typically have many more parameters than outputs (normally only one loss value), reverse mode is used and underpins the backpropagation algorithm [75].

Program Flow: AutoDiff operates in an adjoint program. First, the primal program is run, which simply performs a normal forward pass through the model to compute its output. Note, that the intermediate values are needed for the adjoint program and must be kept in memory. Secondly, the adjoint program is executed, which in reverse AutoDiff

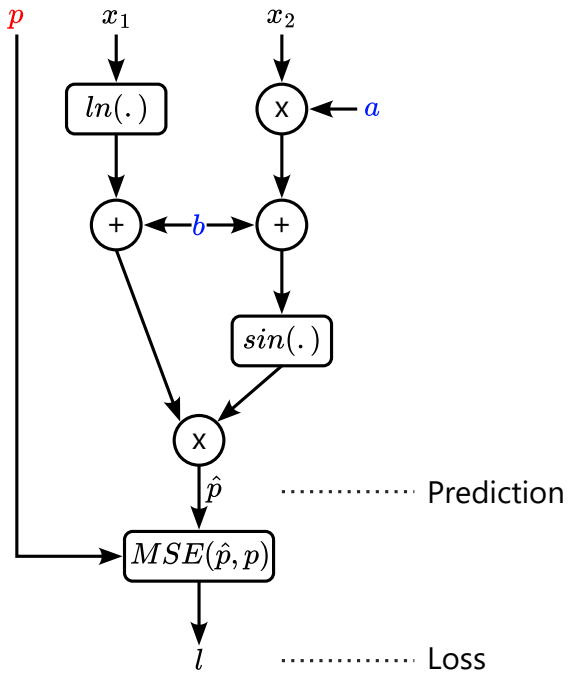


Figure 2.12: Mathematical computation with input variables and internal weights computing a prediction and loss, resembling the scheme of a Neural Network depicted as graph.

mode runs with an inverted control flow from output towards input and computes the local derivatives at each node, which are then finally aggregated using the chain rule.

Example Code: Let us consider an example. Fig. 2.12 depicts a simple computation as graph that relates to NNs. It features two external variables, x_1 and x_2, which could be inputs, and two internal variables, a and b, which could be trainable parameters or weights. It first computes a prediction $\hat{p}$ as

$$\hat{p} = (\ln x_1 + b) \cdot \sin(a x_2 + b), \tag{2.15}$$

which is then used to compute the loss l as (mean) squared error to the corresponding ground truth p. As discussed above, to minimize l, the trainable parameters are adjusted based on their gradients to the loss, i.e., $\frac{dl}{da}$ and $\frac{dl}{db}$. AutoDiff makes computing these gradients trivial, as demonstrated in the code example in Fig. 2.13. Essentially, all gradients between any variables that stand in a chronological relationship can be computed with no effort on the programmer-side.

Some notes are in order: Most importantly, to allow the gradient to "flow" through the complete graph, AutoDiff requires all traversed computations to be differentiable. Secondly, although tracing the graph adds overhead while executing the forward pass, in a training procedure this only needs to be done once at the start as long as the graph's structure does not change. Note that the variables' values are allowed to change. Lastly, one of the huge benefits of AutoDiff is that one can implement significantly more com-

```python
import tensorflow as tf
x1, x2 = tf.Variable(1.0), tf.Variable(2.0) # inputs
a, b = tf.Variable(3.0), tf.Variable(4.0) # weights
p = tf.Variable(5.0) # ground truth

with tf.GradientTape() as tape:
    p_hat = (tf.math.log(x1) + b) * tf.math.sin(x2 * a + b)
    loss = tf.reduce_mean(tf.math.square(p - p_hat))
    [dl_da, dl_db] = tape.gradient(loss, [a, b])
print("dloss/da=%f, dloss/db=%f" % (dl_da.numpy(), dl_db.numpy()))
Output: dloss/da=96.339973, dloss/db=55.977867

# Results can be verified using using WolframAlpha:
# D[(p-((log(x1)+b)*sin(x2 * a + b)))^2,a],x1=1,x2=2,a=3,b=4,p=5
# D[(p-((log(x1)+b)*sin(x2 * a + b)))^2,b],x1=1,x2=2,a=3,b=4,p=5
```

Figure 2.13: Implementation of the algorithm from Fig. 2.12 based on automatic differentiation using Tensorflow [76].

plex computational schemes that might include advanced components, such as a Discrete Fourier Transform (DFT) or linear system solver in between the NN, while still being able to compute gradients through these procedures at no cost to the programmer. This is a key enabler for joining ML-based procedures with classic procedures and optimizing performance in a joint way from end to end. In this dissertation, this feature is heavily used for the DL-augmented signal synchronization procedures outlined in [Pub4.B], which combine classic procedures and components, such as Fast Fourier Transform (FFT) and complex multiplication, with DL to, for instance, estimate and compensate for Carrier Frequency Offset.

2.3 Deep Learning in the Communications Domain

The evolution of communication technologies has been marked by paradigm shifts that redefined the field. The first revolution enabled long-distance radio-based communication through Telegraphy in the 19th century [77], paving the way for global connectivity. The second revolution transitioned analog communication systems to digital, with notable examples such as mobile communications evolving from 1G to 2G/Global System for Mobile Communications (GSM) [78], television broadcasting advancing from National Television Systems Committee (NTSC) and Phase-Alternating-Line (PAL) to digital formats like DVB-T (terrestrial) [79], and radio broadcasting shifting from analog Amplitude Modulation (AM)/Frequency Modulation (FM) to digital standards like Digital Audio Broadcasting (DAB) [80], DAB+ and HD Radio. These transformations funda-

mentally altered how communication systems were built and interacted with, creating opportunities for higher quality and efficiency.

However, the methods used to develop and optimize these systems remained largely rooted in traditional, model-based engineering principles. Today, the emergence of AI challenges the status quo and has the potential to initiate the third revolution, where ML and DL are transforming communications by introducing data-driven methodologies.

In this section we review how AI has anchored in communications, discuss encountered challenges and proposed solutions, and introduce the auto-encoder architecture used for E2E training of communication systems.

2.3.1 Review of Techniques

The motivation to explore AI in communications is both driven by new use cases with more stringent communication requirements, such as remote surgery, intelligent transport, and factory automation [81, 82] and the wish in an increase of performance and decrease of complexity, but also, frankly speaking, due to a general curiosity of exploring how AI could benefit the field. The many disciplines in communications are naturally distributed across all layers of the OSI-model.

AI has already found interest across all layers, for instance, for optimized resource allocation with distributed Medium Access Control (MAC) using deep RL-based protocol design [83, 84] in the data link layer, intelligent traffic control in the network layer [85], adaptive congestion control [86] in the transport layer, or semantic communications [87] in the presentation/application layer to name a few. Especially in the physical layer ML has sparked profound interest in the community for nearly a decade [24]. Hereby, various paths are explored:

On the one hand, existing processing blocks, such as those discussed in Sub-sec. 2.1.3, are re-implemented or augmented with AI in a mostly isolated fashion, such as GAN-based channel modeling for realistic performance evaluation [88–90], coding/decoding techniques [91, 92], learned precoding directly from data for more robust beamforming [93], pilot-less and less computationally complex OFDM [94, 95], frame synchronization [96–98], and many more.

One the other hand, the joint optimization of a transceiver by merging multiple physical layer blocks in the communication chain, such as detection, channel equalization, and decoding [99, 100], or constellation shaping with adaptive transmit and receive filters and detection [101], aim to learn complete waveforms based on the novel concept of autoencoders (see Sub-sec. 2.3.3).

Additionally, completely new research directions and applications have been enabled by ML. Simple use-cases, such as identifying the modulation scheme in unknown received data simply benefit from ML's ability to work effectively with high dimensional data, similarly as in the computer vision domain. This also gives rise to channel charting which allows to extract information from the high dimensional channel state in multi antenna systems to allow, e.g., location-aware services [102–104]. Differentiable ray-tracing simulates the radio wave propagation in concrete scenes and can be calibrated (material and geometry) to real-world observations by minimizing a formulated loss

between the digital prediction and measurements [105, 106].

For more detailed insights into these and other techniques, we refer the interested reader to [107–110].

As opposed to different fields such as NLP or computer vision, AI does come in all its flavors in communications when it comes to NN architectures, training methodologies and loss functions, scopes of learning (ref. to Sub-sec. 2.2.2), and node resources for deployment (user-device, base station, cloud).

It is already certain that AI will find its way into new communication systems and their standards, such as 5G and beyond [111–113]. Standardization organizations like the 3GPP are already promoting to augment regular functionality with ML [114], but ideas are already being shaped that go much further, such as to implement the radio access network using an AI-native air interface that is trained from E2E [115]. This could already come as early as 6G. New frameworks that marry communication functionality with AI, such as Sionna [46], which is heavily used in this dissertation, are rapidly finding popularity.

But even more general technology like GenAI [116, 117] and LLMs find appeal in the communications field. Pre-trained foundation models are fine-tuned on mostly textual communication datasets, such as normative documents, mathematics and algorithms, software documentation, or similar, and can help technicians configure a 5G base station or WLAN network [118, 119].

In essence, AI introduces the ability to take advantage of actually obtained data, hence, enabling a data-driven methodology. The next section discusses traditional model and new data-driven approaches and devises a trade-off that is pursued in this thesis.

2.3.2 Model- vs. Data-driven Approach

Model-driven approaches are currently at the backbone of communication systems. They rely on explicit mathematical and information theoretical formulations grounded on [44] to describe system behavior. Along being theoretically near-optimal in performance, key advantages of model-driven methods include robust performance, a predictable and/or deterministic output, and most importantly, explainable behavior. However, model-driven methods face limitations when the underlying assumptions do not hold. Real-world communication scenarios often involve complex, dynamic, and nonlinear phenomena that are difficult to model accurately. In such cases, traditional approaches may struggle to achieve optimal performance.

On the other side, data-driven techniques allow to design or evaluate system components directly based on real-world observations and not using models as proxy. This is indispensable when working with nondeterministic and highly-dynamic variables like channel state information [120]. Their key strengths are adaptability for complex scenarios even with poorly understood dynamics without the need for explicit modeling, increased performance in hard-to-model scenarios, and a high degree of automation in the sense of taking workload from the human developer. DL makes it exceptionally simple to design and train complete signal processing chains solely using data.

Despite their potential, data-driven methods are not without challenges: First and

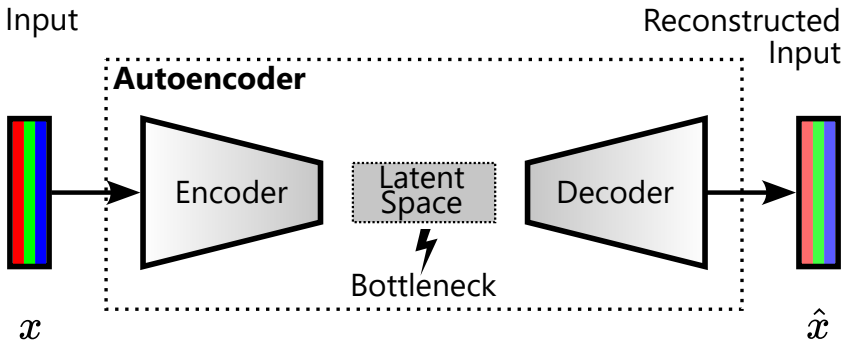


Figure 2.14: Block diagram illustration of the Autoencoder concept.

most obvious, a data dependency is inherent, requiring high-quality and ideally large datasets for training which might not always be available or come at a cost. Moreover, computational resources in the training phase are required, which can be costly and time-demanding. However, lastly and most importantly is the lack of explainability of such black-box algorithms, making it difficult to ensure reliability and trustworthiness of the result [121, 122], which can be an exceptionally strong blocking point for mission-critical functionality or verification procedures such as required by the aerospace industry.

This raises the question, which methodology to choose. What seems to have gotten traction are hybrid approaches that combine the best of both worlds by integrating expert domain knowledge into the structure of NNs, creating architectures that respect understood physical principles [123]. One architecture of choice that is heavily used in this dissertation is the Autoencoder (AE), which is elaborated in the following.

2.3.3 Autoencoder Inspired End-to-End Communication

«An AE is a neural network that is trained to attempt to copy its input to its output» [63]. When recalling Shannon's definition of the communication problem, which is «... reproducing at one point either exactly or approximately a message selected at another point» [44], we can observe an interesting parallel, which opens an opportunity: When formulating the communication problem using the concept of an AE, we can leverage this heavily studied architecture and its many variations implemented using DL to address the problem [124]. The AE concept was initially introduced by Rumelhart et al. [125]. Fig. 2.14 illustrates its block diagram and main functionality. Base on a learned encoder and decoder block, AEs find an efficient encoded and/or compressed representation of the input information in a latent space and automatically learn the corresponding inverse operations to recover the original data from the latent space [63, 126, 127]. Depending on the penalty or bottleneck applied to the data in the latent space, the characteristics of the AE can be influenced. Inherent auto-labeling of data makes AEs suitable for situations where labeled data is scarce or expensive to obtain [126]. Hereby, the following two categories are very applicable to communications [63]:

- Undercomplete AEs: The latent space has a smaller dimension than the input x.

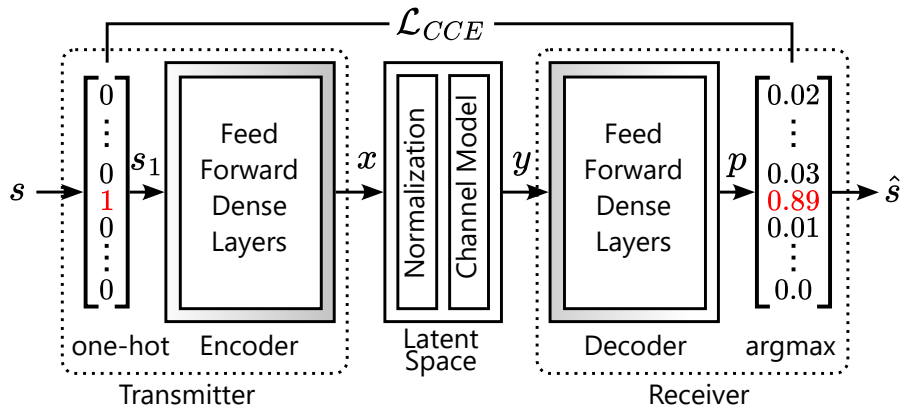


Figure 2.15: Autoencoder system concept for E2E-learned symbol-wise communication. Adapted from [24].

This forces the NN to capture the most salient features of the data. Furthermore, it can be used to enforce specific data shapes that represent "interfaces". For instance, a complex base band I/Q symbol/sample can be represented by a latent space of 2 units.

- Regularized AEs: Constraints are added to the loss function or the learned representation, such as:
  - Sparsity: A sparsity penalty on the loss function encourages the model to represent data with a limited set of active units, learning unique features.
  - Denoising: By training the decoder operation on corrupted or noisy inputs, the AE learns robust representations by capturing the data distribution.

To model the communications problem, a combination of under-completeness and regularization is suitable. Fig. 2.15 shows a practical implementation for communicating discrete symbols with an AE. The functionality is simple: After picking symbol s from N possible symbols at the transmitter, it is mapped to a one-hot vector s_1 of length N and then processed through the AE's encoder stage, producing x. In Fig. 2.15, both en- and de-coder are implemented using Feed-forward Dense Layers (FFDL). In the latent space, x is subject to penalties found in communication systems, such as an average and/or maximum power constraint, channel impairments (noise, fading, multi-path [128]), interference, etc., resulting in y. The decoder then produces a reconstruction of the one-hot vector p by giving probabilities per element using the softmax activation function. The received symbol is determined as the item with the highest probability.

The selection of the loss function is crucial for achieving information theoretical maximum performance. For symbol-wise communication with the above architecture, the loss is calculated by linking s_1 and p with the categorical cross entropy loss function [24]. However, there are some short comings, such as the need for manual bit-labeling of the symbols. A different architecture, which is primarily used in this dissertation, is the bit-wise AE [67, 129]. Instead of representing the symbol with a one-hot vector, its index is

directly represented by their bits. Each of these bits is directly reconstructed by the AE. By minimizing the binary cross entropy between input and output bits, Log-Likelihood Ratios (LLRs) are produced that represent the certainty of each bit's value.

The AE architecture forms the foundation for E2E-optimized communication systems [107, 129–131]. The systems can be flexibly extended with arbitrary functionality implemented using classic operations (thanks to AutoDiff, see Sub-sec. 2.2.4) or learned AI/ML-based operations [132, 133].

3 Dissecting Artificial Intelligence-enabled Technology on the Edge in Space

This chapter targets to shine light on the challenge of realizing AI on-board spacecrafts. Based on the example of current telecommunication satellite platforms and payloads, we infer the necessity to involve completely new hardware technology to achieve this goal. A review of currently available and future state-of-the-art AI inference platforms relevant for space deployment and a comparison of their Key Performance Indicators (KPIs) is given. This is followed by a detailed analysis of AMD-Xilinx's VERSAL AI CORE platform in [Pub3.A], the platform that is at the core of next-generation spacecraft processors at Airbus DS and central to subsequent chapters. A discussion on future steps concludes this chapter.

3.1 On the Inevitable Shift in Hardware Technology

3.1.1 Introduction to Satellite Platforms and Payloads

A satellite's hardware can be coarsely separated into the satellite platform, also called satellite bus, and the payload. The satellite platform handles generic functionality, such as command and data handling, the operation of communication systems and antennas, electrical power distribution, propulsion, thermal control, attitude control, guidance, and navigation [134]. Individual mission-specific functionality, e.g., scientific instrumentation or communication-related analog or digital signal conditioning components, is realized in the satellite's payload, which sits "on-top" of the bus. The major benefit of this modular approach is the re-usability of the satellite platform for multiple missions, reducing overall development effort per mission.

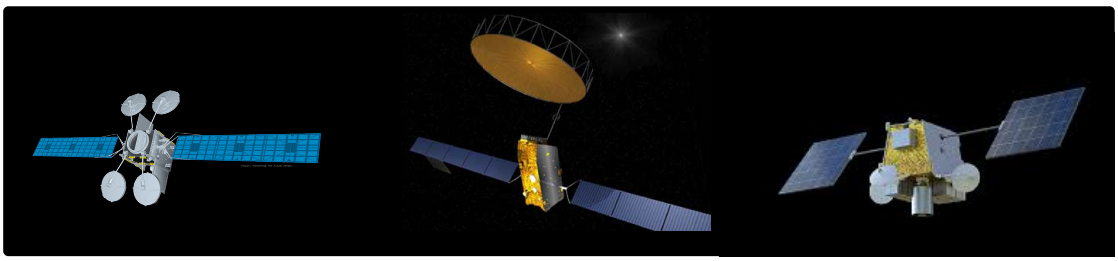


Figure 3.1: Images of satellite platforms: Boeing Space System 702 series (left), EADS Astrium's Eurostar series (middle), OneWeb's ARROW platform (right). Image credit: [135–137].

To give an intuition, Fig. 3.1 shows popular satellite busses, such as Boeing Space System's 702 series (left), EADS Astrium's Eurostar series (middle), and OneWeb's ARROW platform (right). For telecommunication satellites, platform capabilities vary significantly depending on the orbit. Geostationary Earth Orbit (GEO) typically hosts few but powerful HTSs that consume multiple kilowatts of power [138], provide 100s of Gigabits per second to Terabits per second of total data bandwidth [139], and weigh multiple tons in total with payload weights exceeding one tone [140]. The first two platforms mentioned above target this use-case. On the contrary, Low Earth Orbit (LEO) typically hosts much lighter satellites that can be multiple orders of magnitudes less powerful, but achieve similar or higher performance through a high satellite density. The ARROW platform targets such satellites and is used for the OneWeb [141] constellation.

Circling back to our original target of bringing AI processing capabilities to telecommunication satellites, multiple questions arise:

1. Should AI be integrated on the satellite bus or the payload level?
2. Should the overall AI processing capability be aligned with the requirements of HTS, i.e., GEO systems, or the comparably small LEO satellites?
3. Which AI hardware platforms (CPU, GPU, FPGA, ASIC, neuromorphic) are most promising for space deployment, and why?
4. [RQ-1] - How can AI-based processing capabilities be realized on regenerative satellite payloads under the constraints of the space environment?

For question 1 the answer depends on the concrete AI use case. In case the use-case is mission-specific and requires a specialized development effort (e.g. hardware) the standard approach is integrating it on the payload-side. However, very platform-near use cases, such as AI-augmented FDIR which works on a multitude of internal telemetry signals [142, 143], or AI-powered collision avoidance which requires control of the navigation unit, would profit from being integrated directly in the satellite bus' computer. For telecommunication use-cases any kind of processing capability is situated on the payload, rendering it regenerative.

Although the answer to question 2 seems trivial, i.e., always aligning the processing capabilities with the respective system requirements, the practical answer requires a concrete decision on system requirements and orbit, which in the European space sector is still unclear due to the current market situation (early 2025). To stay competitive and kick-off development, a practical approach to reduce financial risk and development burden is building one product that can be scaled to different situations in a later stage. To get the process started, knowledge of different space-enabled hardware technologies, as indicated by question 3, is indispensable. This thought process can be summarized under [RQ-1]. In the following, emerging AI processing technologies and their suitability to space deployment are discussed.

3.1.2 Emerging Hardware Technology and Computing Environments

The central KPIs in determining the suitability of AI accelerators for space deployment are

- NN capabilities: Architecture support, model size, inference and training capability
- Processing capacity: typically given in *Terra Operations per Second*
- Power efficiency: typically given in *Terra Operations per Second per Watt*
- Space-grade rating: Radiation resilience classification, operating temperature range
- Size, Weight, and Power (SWaP)
- Part costs
- Development effort

We want to note, that the space-grade rating, specifically with respect to the radiation resilience classification, is to be taken with a grain of salt. The use of Commercial Off-The-Shelf (COTS) components instead of qualified components has been heavily studied [144, 145] and is already successfully applied in the NewSpace era, however, mostly in LEO. GEO systems on the other side still demand a high level of failure resistance due to their longer lifespan and higher radiation levels which make space-grade components the preferred choice.

AI inference accelerators can be grouped by their underlying processing architecture, i.e., CPU-, GPU-, FPGA-, or ASIC-based. In the following we list relevant approaches and corresponding products that are currently highly considered in the space community:

CPU-based: The most straight-forward approach is to utilize generic processors, i.e., CPUs or Vector Processing Units (VPUs), that can run arbitrary programs. While providing the highest level of flexibility regarding NN capabilities, the strongest drawback is typically small computational capacity and power efficiency, specifically for space-grade processors.
Various optimized approaches have emerged: Klepsydra Technologies has developed an optimized software framework that reduces power consumption for CPU-only inference by up to 50 % and outperforms market-leaders like OpenCV and TensorFlow for embedded devices [146]. While the framework works on lots of different COTS processors, a collaboration with the leading provider of space-grace processors, Frontgrade Gaisler, aims at integrating the software framework into their radiation-hardened processors [147] to provide a space-grade solution. On the other side, new hardware has also emerged, such as Intel's Movidius Myriad VPU, which realizes a "neural engine" via a parallel CPU-near architecture for generic use cases.

GPU-based: GPUs are orders of magnitude more capable than CPUs. For operation in space, only embedded versions (due to missing air circulation) and soft-GPU implementations, i.e., emulations of GPU architectures, are relevant. Major advantages are huge compute capacities, high power efficiency, very low costs, and few development effort.
NVIDIA currently provides the state-of-the-art embedded GPUs with its Jetson Orin series which delivers up to 275 Terra Int-8 Ops/s [148]. By combining a general-purpose 2048-core Ampere GPU, up to 64 Tensor Cores, and a custom-silicon Deep Learning Accelerator into one System-on-Module (SoM), it can deliver remarkable power efficiency with up to 4.58 Terra Int-8 Ops/s/W [148]. Similar products from competitors, such as AMD, exist. Biggest drawback is the lack of radiation resistance by design which fuels hesitation by payload manufacturers, which sparked 3rd party radiation tests [149–152]. An alternative are Soft-GPUs that can be implemented, for instance, as FPGA-IP [153] on space-qualified hardware.

FPGA-based: Due to the long heritage of FPGAs as powerful processing unit in the space-domain, FPGA-based AI solutions are highly popular. The mode of operation can be divided into three categories: In *standalone* operation IP-cores that are either custom written or generated using frameworks based on high-level synthesis, such as HLS4ML [154], FINN [155], MATLAB's Deep Learning HDL Toolbox [156], etc., are deployed on the FPGA to realize a specific ML/DL task. In *emulation* operation, processing architectures targeting (generic) NN inference are emulated on the FPGA, such as soft-GPUs [157], soft-CPUs (e.g., Leon3 [158]), neural engines, or other inference engines like AMD-Xilinx's Deep Learning Processor Unit (DPU). In the third operation mode the FPGA is tightly integrated with additional peripherals, such as CPUs or DRAM, packaged together as SoC. SoC series like AMD-Xilinx's UltraScale+ and VERSAL enjoy remarkable popularity for AI processing due to AI-toolchains provided by the vendor, high performance (approx. 100 Terra Int-8 Ops/s on VERSAL AI-Core with Vitis-AI toolchain), and availability in radiation-tolerant formats.

ASIC-based: In the ASIC category falls anything that significantly differs from the above categories and utilizes a custom processing architecture. Due to very high capital expenses required for ASIC development and the comparably small space market, there exist only very few dedicated space-grade AI inference engines.
However, custom components can be integrated in other products, for instance as seen with the AI-Engines (VPUs optimized for efficient matrix multiplication) in the VERSAL SoC series. Other COTS components, such as Google/Coral's EdgeTPU with 4 Terra Int-8 Ops/s at 2 Terra Ops/s/W [159] or HAILO TECHNOLOGIES' Hailo-8 Edge-AI chip with 26 Terra Int-8 Ops/s at 2.5 W power consumption [160] have seen applications in NASA's SpaceCube Edge TPU [161].

A special sub-field that mostly builds on ASICs is neuromorphic computing. By using spiking NNs it aims to emulate the self-organizing and -learning nature of the brain, and allows to execute "traditional" ML operations as discussed in Sub-sec. 2.2.3 in an optimized way [162]. Multiple products have entered the market, such as Brainchip's

Akida chip (and corresponding FPGA-IP), Intel's Loihi chip, and IBM's TrueNorth. Missing radiation resilience by design is still the main drawback, which some vendors address by using the inherently resilient 22 nm Fully Depleted Silicium on Insulator (FD-SOI) process [163], or by utilizing external shielding.

Various options and their key advantages, disadvantages, and KPIs have been given. In this dissertation we select a hardware platform from the FPGA technology category that also integrates CPU and ASIC into one SoC: AMD-Xilinx's VERSAL AI CORE aSoC. The next section dives deeply into this platform's AI capabilities.

3.2 Pub3.A: Accelerated Deep-Learning Inference on the Versal adaptive SoC in the Space Domain

Authors: Michael Petry, Gabriel Wuwer, Andreas Koch, Patrick Gest, Max Ghiglione, Martin Werner

Publication Medium: European Data Handling & Data Processing Conference (EDHPC), 2023

Digital Object Identifier: https://doi.org/10.23919/EDHPC59100.2023.10396011

Thesis contextualization: This paper addresses this chapter's main question of identifying a suitable hardware platform that can deliver efficient, scalable, and flexible AI processing capabilities on-board spaceborne systems, and [RQ-2] of this dissertation. By taking a deep look into the components and their inter-operation principle of AMD-Xilinx's VERSAL AI Core aSoC platform, its processing capabilities and realistic performance is thoroughly analyzed for representative ML workloads as discussed in Sub-sec. 2.2.3. Furthermore, as the remaining hardware-related research in this dissertation is oriented on this platform, it serves as additional background context for succinct chapters.

CRediT author statement:

Michael Petry:	35 %	Conceptualization, Methodology, Software, Data curation, Formal analysis, Validation, Visualization, Writing
Gabriel Wuwer:	30 %	Conceptualization, Methodology, Software, Data curation, Formal analysis, Validation, Visualization, Writing
Andreas Koch:	15 %	Methodology, Investigation
Patrick Gest:	10 %	Software, Investigation
Max Ghiglione:	5 %	Conceptualization, Investigation
Martin Werner:	5 %	Conceptualization, Writing

Accelerated Deep-Learning Inference on the Versal adaptive SoC in the Space Domain

Michael Petry, *Graduate Student Member, IEEE, Student Member, Optica*, Gabriel Wuwer, Andreas Koch, Patrick Gest, Max Ghiglione, Martin Werner

***Abstract*—Artificial intelligence has found its way into space and necessitates a powerful and flexible hardware platform to keep up with the fast-paced AI domain. With the space-grade variant of the Versal, AMD-Xilinx offers one of the first space-ready AI accelerators that combine multiple compute paradigms, i.e., scalar processing (CPU), adaptive engines (FPGA), and vector processing (AI-Engine array) into an adaptive System-on-Chip. This paper provides a thorough analysis of its AI capabilities with respect to throughput and power efficiency for Multi-Layer Perceptrons and CNNs, and takes a look under the hood by profiling the system's efficiency on an architectural level based on the idea of the Roofline model. We believe that the gained insights ultimately help to design optimal NN architectures for deployment on the Versal.**

***Index Terms*—fpga, hardware accelerator, neural network, machine learning, AMD-Xilinx Versal, roofline model**

I. Introduction

Future telecommunications satellites are envisioned to seamlessly integrate into the mobile communication networks of the next generation. First signs towards the integration are already visible today with the ongoing normative activities on non-terrestrial networks in the current 5G New Radio standard [1]. This is a novelty for satellite manufacturers, as it marks the first time that satellites are explicitly included in a mobile communications standard [2]. Within the same scope, efforts are made to integrate Artificial Intelligence (AI) / Machine Learning (ML) techniques into orbit to serve various use-cases, such as Cognitive Radio, Earth Observation, Anomaly Detection, and many more. There is a major desire in the satellite industry for deploying NNs and similar algorithms directly on-board of the satellites in space. The main challenges associated with that desire are the limited power budget and computing resources of satellites. Furthermore, due to the requirement to operate in the harsh space environment, telecommunications satellites are typically restricted to the use of radiation-hardened hardware, and in particular space-grade field-programmable gate arrays (FPGAs) as the most potent parallel compute platform for AI inference in space [3].

With the Versal adaptive System-on-Chip (SoC) in the XQR variant, AMD-Xilinx offers a chip in a space-grade package that is particularly targeted for machine learning applications in space. By combining a scalar processing system (PS) with a high-compute-density FPGA fabric and ASIC-like vector engines, it promises more compute power at a higher efficiency than traditional single-technology systems [4], [5]. Its real strength, however, arises by utilizing these three compute paradigms in concert to perform ML operations in a deeply hardware optimized manner. To ease use for the developer, AMD-Xilinx provides an Intellectual Property (IP) block named Deep-Learning Processor Unit (DPU), which comprises an FPGA hardware image, instructions for the AI-Engines, and a data-handling and communication interface to the PS. An initial evaluation of the design flow and the basic capabilities of this framework on the Versal platform is investigated in [6].

Since the DPU is merely a black-box for the user, predictions on performance and suitability of specific NN architectures are hardly possible. Therefore, this work aims at providing transparency by investigating how different NN architectures impact the achievable performance on the Versal. By utilizing a grid-search we sweep through various network parameters to track throughput performance and power demands within the IP's core feature domain, i.e., Multi-Layer Perceptrons (MLP) and Convolutional Neural Networks (CNN). While this provides a broad superficial impression on the DPU's performance, we dive even deeper by profiling various architectural properties, such as hardware execution time, operational efficiency, and memory bandwidth utilization to study the internal reasons for the observed performance. This procedure aims at understanding how to design efficient NNs for hardware-accelerated scenarios, ultimately benefiting the complete range of ML application.

This paper is structured as follows: Section II provides a brief survey of state-of-the-art AI hardware accelerators for edge-deployment. Section III follows up on this by introducing the Versal hardware platform and the DPU, including a brief architectural review. The main contribution of this paper is given in section IV, which provides a throughput and power efficiency benchmark for several NN variations. Section V dives into the architectural level by computing a Roofline model, and investigates the origin of the observed behaviour. Lastly, a conclusion and outlook of future work is given in section VI.

This work is supported by the German Aerospace Center (DLR) under project Machine Learning on Telecommunications Satellites (MaLeTeSa) with Grant number 50 YB 2103. Corresponding author: Michael Petry (Email: michael.petry@airbus.com). M. Petry and A. Koch are with Technical University of Munich (TUM) and Airbus Defence and Space GmbH (ADS), G. Wuwer is with DHBW Ravensburg and ADS, Patrick Gest is with ADS, Max Ghiglione is with European Space Agency, and Martin Werner is with TUM.

II. State-of-the-Art AI Accelerators

AI applications have found their way into our daily lives by penetrating into multiple domains such as entertainment, industrial, and health care, for instance with speech-to-text translation on phones, on-the-fly video post-processing for conference video chatting, and much more. ML-models have been observed to increase in size and complexity by one order of magnitude per year [7], which renders deploying hardware-compatible updates extremely difficult or even impossible. This gave rise to huge research and development efforts in building long-term hardware platforms. The proliferation of resource-intensive ML applications has pushed the tech industry into developing a "smarter" hardware platform that provides higher adaptivity and reprogrammability while outperforming CPUs and GPUs in metrics such as power and throughput. As a result, a variety of edge-targeted AI hardware accelerators have emerged, whereas the most modern platforms will be briefly summarized here.

The most popular AI accelerators are GPU-based. In terms of AI edge deployment, NVIDIA leads the field with its novel 7nm Orin System-on-Module (SoM), which is a successor to the Jetson platform. The Orin architecture comprises an NVIDIA Ampere GPU architecture with up to 2048 CUDA-cores, a 12-core ARM Cortex-A78 application processor, and a silicon-based AI computation unit, called Deep Learning Accelerator (DLA). The interplay of these elements makes the module extremely capable, yielding 275 Int-8 TOPS theoretic peak compute power at 4.58 TOPS/W, delivering superior power efficiency.[1] Native in-silicon execution of standard ML operations, such as CNNs or pooling, paired with the flexibility of a GPU provides the highest level of efficiency and flexibility. However, optimal utilization of the GPU depends on the concrete NN architecture to accelerate, since architectural restrictions, such as memory bandwidth limitations or limited interconnections between CUDA-cores, can pose significant bottlenecks [8].

A similar approach is applied at Intel for their AI-optimized Stratix NX series, which builds on 14nm FPGA technology. By replacing standard Digital Signal Processing (DSP)-slices in the fabric with 3960 AI-optimized Tensor Cores, a $15\times$ higher Int-8 fixed point compute performance is achieved for matrix-matrix & matrix-vector multiplication, and element-wise operations, yielding a total compute power of 142.6 TOPS at around 1.5 TOPS/W. Alternatively, Intel pursues a second strategy with its Embedded Multi-Die Interconnect Bridge (EMIB) technology, which allows integrating external dies into the FPGA. One implementation that exploits this interface for accelerating deep learning operations are the Intel Stratix 10 FPGAs with TensorTiles [9].

Lastly, growing efforts go into designing custom application-specific integrated circuits for the AI domain. Google recently passed its fourth iteration for its Tensor Processing Unit (TPU), matching the Orin's compute capabilities at a reduced power efficiency of 1.62 TOPS/W. However, the system is primarily not intended for standalone deployment, since low-level communication interfaces allow for clustering multiple of these boards. Other approaches consider waver-scale ASICs that utilize a complete silicon waver as one module without slicing individual chips.

[1]We want to note, that the Orin platform supports sparse processing with a sparsity factor of up to 2. It's effective peak TOPS is therefore only half.

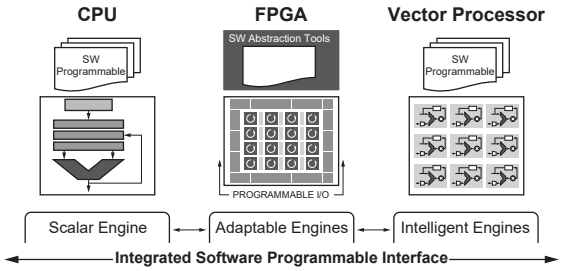


Fig. 1. Overview of different types of programmable compute resources integrated into the Versal's heterogeneous platform [10].

In conclusion, it can be observed, that all modern AI-targeted hardware platforms based on GPU or FPGA include an AISC-like computation unit that is designed for specific AI operations. This is a key enabler for power-efficient and high-performance ML inference. However, none of the above mentioned products or their competitors provide a space-ready variant in terms of radiation-hardness or -tolerance. One exception that follows this approach is the Versal adaptive SoC, which is a space-ready FPGA-based AI-optimized hardware platform. The Versal is introduced in the following section.

III. Versal Platform for DL inference

The previous section showed, that none of the prior mentioned technologies can be considered fully optimal, as they all excel in one or more specific characteristic, but do not provide a "one size fits it all" solution. The Versal AI Core series, introduced by AMD-Xilinx in 2019, is an approach to combat the existing problems by unifying multiple technologies in one SoC to extract and combine the best of all technologies. In the following, we will provide a detailed look into this system.

The Versal adaptive SoC is a fully software-programmable heterogeneous compute platform that combines scalar and vector processing elements tightly coupled to next-generation programmable logic, all interconnected with a high-bandwidth network-on-chip (NoC). This concept is shown in Fig. 1. Scalar processing, a domain where CPUs natively excel, is essential for algorithms that exhibit frequently occurring decision-based branches that control complex and nested program flows. Vector processing, as implemented in GPUs or DSPs, is optimal for domain-specific parallelization such as signal processing, e.g., complex math or convolutions. Lastly, adaptable hardware enables to operate on irregular data structures and workloads under tight latency constraints by providing flexible parallel compute and fast local memory, ultimately enabling real-time control. This hybrid architecture is promised to enable a performance increase through the

operation of multiple technologies in concert, outperforming single-technology-only solutions [10].

The major workhorse for AI computation is, similar as for the above technologies, an ASIC-like vector computation unit called AI-Engine Array, which is detailed in the following subsection.

A. AI-Engine Array

The intelligent engines, also called AI-Engines (AI-E) in the context of the Versal AI Core series, are an array of very long instruction word (VLIW) and single instruction, multiple data (SIMD) processing units combined with dedicated data and instruction memory. Each AI-E features dedicated fixed-point and floating-point vector units. The units support various combinations between eight 32-bit x 32-bit floating-point MACs or 128 8 x 8 bit MACs per clock cycle through a full permute unit with 32-bit granularity. Combining the VLIW and SIMD approach, two vector input streams can be concurrently loaded into the vector processing unit via a 256 bit-wide stream, and saved back to memory via a 256 bit-wide store unit, while performing additional functionality such as fix-to-float conversion or non-linear activation in silicon, all using a single 7-way instruction.

One of the strengths of the AI-E array lies in its extremely flexible interconnectivity. Thanks to various communication interfaces within the AI-E array, dataflow can be optimally routed either through a 1-dimensional cascade or a 2-dimensional dataflow. Furthermore, tight integration with the PL (0.96 TB/s) and the rest of the chip through the NoC (100 GB/s) avoids early memory bandwidth bottlenecks, although we show in Section V, that this limit can indeed be reached with certain NNs.

To utilize this array in concert with the PL, AMD-Xilinx provides the IP Deep-Learning Processor Unit, which is explained in the following.

B. Deep-Learning Processor Unit

The Deep-Learning Processor Unit is a configurable computation engine targeted for deep neural networks. AMD-Xilinx offers the DPU for multiple hardware platforms and with different optimization schemes, i.e., targeting throughput, latency, or scalability. Once selected, the DPU is deployed on the PL and the AI-Engines. It hence combines a synthesized code-block on the FPGA with a series of C++ programs deployed on the AI-Engines. Both parts together are denoted as a single IP block. It can be viewed as a microcoded co-processor with an instruction set optimized for Deep Neural Network (DNN) operations. Fig. 2 provides a system-model of the DPU. Various configurations, such as batch-sizes of up to six using Batch Handlers (BH), choosing 32 or 64 AI-Engines per BH, as well as running different NNs concurrently using multiple Compute Units (CU) are offered.

ML computation is split between FPGA and AI-E Array. Compute-intensive operations, such as MLPs, CNNs, and non-linear activations are computed on the AI-Engines, whereas data-handling, such as load and store operations, depth-wise convolution, pooling, and element-wise operations, are deployed on the PL. Hence, its theoretic maximum compute capability depends mainly on the AI-E array and is computed as [11]

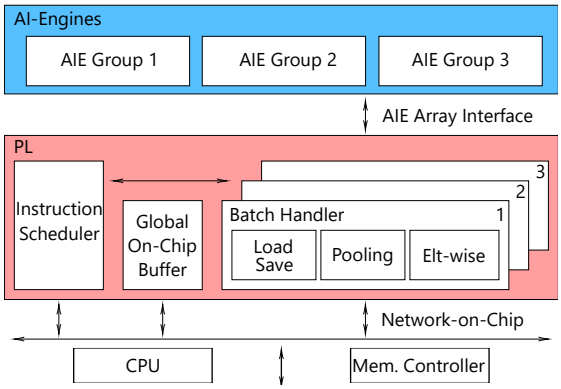


Fig. 2. Block diagram of the DPU architecture (DPUCVDX8G). Adapted from [12].

$$\text{DPU Perf.} = f_{\text{AI-E}} \cdot 256 \cdot \#\text{AI-E} \cdot \#\text{BH} \cdot \#\text{CU}. \quad (1)$$

The memory bandwidth available to the DPU depends on its configuration, specifically, on its number of batch handlers. It is calculated as

$$\text{DDR-BW} = (512 + 128 + 32 + 128 \cdot \#\text{BH}) \cdot f_{\text{PL}} \text{ bits/s}, \quad (2)$$

where 512, 128, 32, and 128 denote the bus widths for weight, bias, instruction, and data AXI memory mapped interfaces, respectively [11].

The native support of ML layers is limited to MLP- and CNN-like operations, however, custom operations of any kind can be integrated into the DPU dataflow either in a non-accelerated manner (PS) or on either PL and/or AI-E Array.

The DPU is benchmarked in three hardware configurations in this work, shown in Table I. We both analyze the impact of batch-parallelization, i.e., single-batch vs. multi-batch execution, and the trade-off between hardware resources and compute power, i.e. 32 vs. 64 AI-Engines per BH.

IV. Benchmark

This section discusses the benchmark strategy and its results. First, the challenge of selecting a representative set

TABLE I
Benchmarked DPU hardware configurations

Config	Freq.[a] [MHz]	FPGA [%]	AIE [#]	Batch [#]	Int8 Perf [TOPS]	Mem-BW [GB/s]
C32B1	333/1250	6%	32	1	10.24	33.3
C64B1	333/1250	7%	64	1	20.48	33.3
C32B5	333/1250	24%	160	5	51.20	54.6

[a]Frequency of PL / AI-Engines

TABLE II
SELECTION OF NN VARIATIONS FOR THE BENCHMARK

Property	Start	Stop	Step size
Multi-Layer Perceptron			
Num. Input	64	512	64
Num. Neurons	64	512	64
Num. Layers	1	9	2
Convolutional Neural Network			
Input Resolution	128x128	512x512	128
Channels	0[a]	256	64
Kernel-Size	1x1	8x8	2x2
Num. Layers	2	8	2
ResNet50			
Input Resolution	512x512[b]	2048x2048	512
Channels	3	-	-

[a] 1 Channel is used as start value.
[b] Resolution 32x32 is additionally provided.

of NN architectures that represent the current landscape of ML models is discussed. Second, the benchmark process w.r.t. strategy, performance metrics, measurement methods, and GPU-based reference platform is elaborated. Afterwards, the results are presented and a first conclusion is taken from a high-level perspective.

A. Selection of Representative NN Architectures

The goal of this benchmark is to provide transparency for key performance metrics of NN execution on the Versal platform using the DPU. Choosing a suitable set of representative NN architectures that allow to infer predictions for modern ML models, which resemble a combination of various ML operations and layers in a single model, is a challenging task. In this work, we decided to separately focus on single ML operations in their various configurations. This allows for a more fundamental understanding of how performance scales w.r.t. key NN properties, such as num. of layers, input resolution, etc. By covering a sufficiently wide range, this allows for a more general understanding compared to analyzing very specific architectures. However, variants of ResNet50 are also considered to provide a practical reference.

To analyze the DPU's performance while fully utilizing the AI-E array, we select the MLP and "standard" convolutional layer, as discussed in Subsection III-B. Each layer is followed by a ReLU activation function, which is considered state-of-the-art and hardware efficient. Table II summarizes the settings for generating the benchmarked NN architectures using an uniform grid-search-wise setting.

The NN models are transformed to their hardware-accelerated equivalent according to the workflow described in [6].

B. Benchmark Strategy, Performance Metrics, and Methods

The benchmark strategy utilized in this work targets two levels of abstraction. On the system-level we analyze the average model inference time and corresponding power consumption over ten runs. The power consumption is determined by reading the Versal's Core and PL Device Rail 'VCCINT' power sensor, which measures the effective power of the DPU (PL + AI-E array). These values provide primary information about the scaling of performance and power consumption as a function of NN parameters. On a deeper, architectural level, we concurrently measure the internal DPU execution time (without python overhead), computational efficiency, and occupied memory bandwidth for each model. This second-level analysis, discussed in detail in Section V, enables to understand the origin of certain hardware-related limitations and restrictions in achievable performance, ultimately benefiting the user in designing more hardware-suitable models for optimal performance.

All NN models are analyzed for all DPU configurations given in Table I. To provide reference performance values, the NNs' average inference times are additionally benchmarked on a consumer-grade NVIDIA Geforce RTX 4070TI GPU by utilizing TensorFlow's graph-execution with Just-In-Time (JIT) compilation.

C. System-Level Benchmark Results

Fig. 3 summarizes the system-level benchmark results for a selection (all combinations of minimum and maximum value for all variables) of the NNs analyzed. Multi-Layer Perceptrons, Convolutional Neural Networks, and ResNets are vertically arranged in groups. The first column denotes the NNs' architectural properties, the second column denotes avg. python inference time for Versal and GPU, and the third and forth columns denote avg. DPU runtime and DPU power consumption for the Versal only, respectively. Different hardware setups are denoted by colored bars. Blue, red, and yellow bars denote C32B1, C64B1, and C32B5 DPU configurations, respectively, whereas black bars denote the GPU's performance for single-batch configuration.

Intuitively, the python inference time grows with NN complexity for all models. A significant dynamic range of 0.1 ms - 0.5 ms for MLPs and 0.1 ms - 2.5 secs for CNNs is observed for the DPU in C32B1 configuration, whereas the GPU's overall dynamic range (0.25 ms - 0.4 ms) is much smaller. Extreme execution times in the order of seconds likely denote an internal anomaly, e.g., on-chip memory overflow for too big models. On average, the Versal's overall inference times for CNNs are approximately 2 orders of magnitudes slower than the GPU's, while they are equal for MLPs. Performance of multi-batch inference on the Versal shows an interesting behavior for MLPs and CNNs. By using a batch-size of five (C32B5 DPU configuration) on the Versal, nearly no python inference time increase is observed for any MLP variant, denoting an about $5\times$ higher throughput. This is not the case for all CNNs, as some execute with nearly the same inference time, while others reach $> 5\times$ higher times, voiding any possible benefit from processing batches in parallel. The reason for this phenomenon is the rather limited memory bandwidth which prevents data-intensive CNNs to utilize the multi-batch processing capabilities effectively, as discussed in detail in Section V. Moreover, in C64B1 configuration, while the MLP's inference times on average increase by a little,

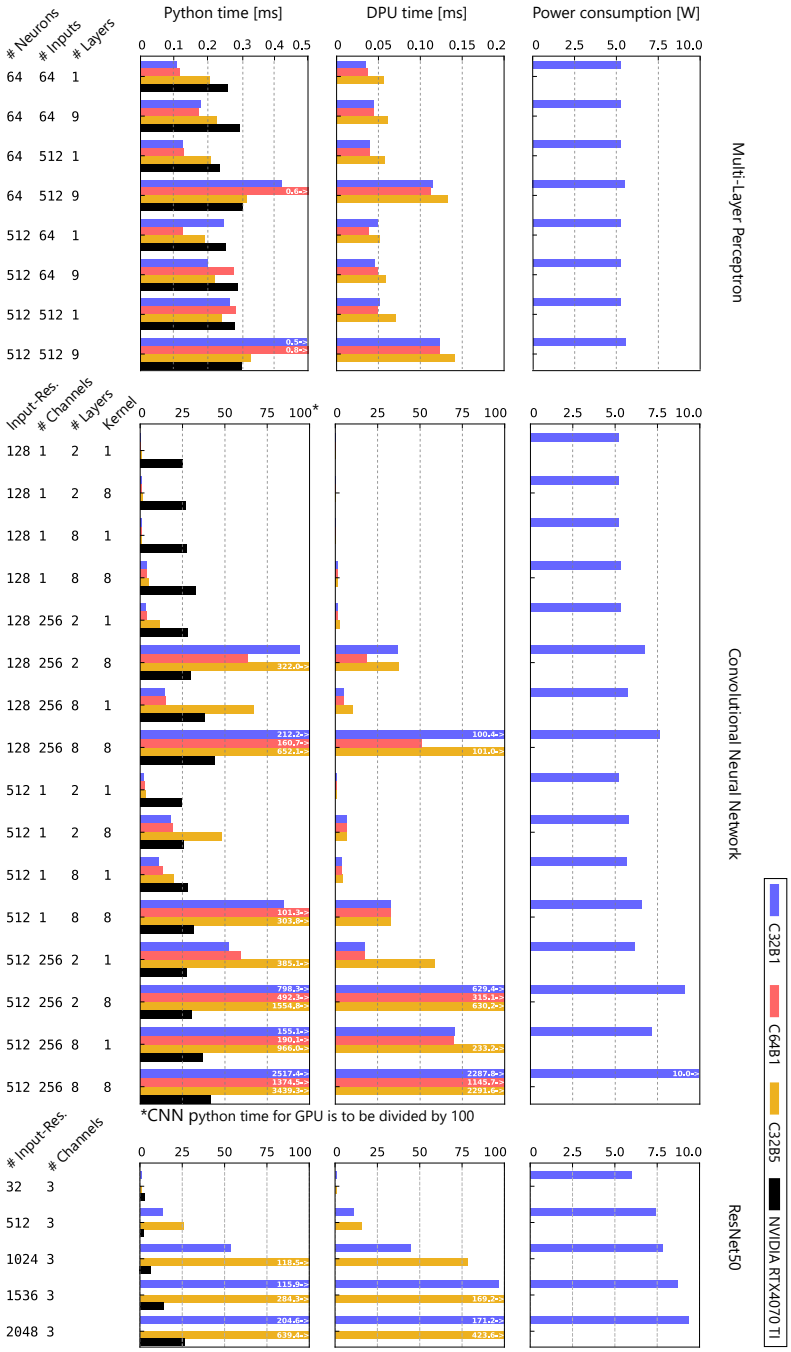


Fig. 3. System-Level benchmark results for MLPs, CNNs, and ResNet50s. Metrics: Python inference time (second column), DPU execution time (third column), power consumption (fourth column). Hardware configurations: DPU C32B1 (blue), C64B1 (red), C32B5 (yellow); GPU (black).

indicating a loss in throughput, the CNN's inference times on average decrease, which means they can efficiently exploit the additional compute resources.

The DPU run-time time (without python overhead) shows a similar dynamic behavior w.r.t. to model type and complexity, however, in terms of magnitude it is about 25% and up to 70% smaller than the corresponding python inference times for MLPs and CNNs, respectively. This indicates a significant overhead introduced in the Python-DPU interface and hints a major area of improvement. Possible solutions could be to utilize the C++-API or to directly control the DPU via its registers.

Lastly, column four denotes the power consumption of the DPU in C32B1 configuration. The power consumption depends heavily on the NN complexity and is approximately stable at 5.4 W for all MLPs and ranges between 5.2 - 10.0 W for CNNs.

V. Architectural Profile

This section provides an in-depth look "under the hood" of the DPU's performance by profiling it on an architectural level. To illustrate the low-level performance of the DPU on the Versal we adapt the Roofline Performance Model from high-performance computing (HPC) [13]. The core idea is that applications are either computationally bound by the maximal arithmetic performance of the system, or memory-bandwidth bound by the maximal speed of the data-transfer interface. This applies well to NNs, since they can be extremely computationally intensive and have to process large amounts of data. Here, the Roofline model offers insights on possible performance bottlenecks.

Fig. 4 summarizes the profiling results in the Roofline model. The y-axis denotes the performance in Giga operations per second of the DPU, which is measured by using AMD-Xilinx's profiling tools. The x-axis denotes the NN's *operational intensity* in terms of DPU operations per byte transfer on the DDR memory interface. This metric is computed as the fraction of the total operations performed for one NN forward pass and the total amount of bytes transferred between DPU and the DDR memory within the inference process. We want to clearly note, that the arithmetic intensity metric used in this work is not independent of the hardware architecture, since the DPU has full control over the weight and data buffering strategy (undisclosed) and, hence, utilization of the DDR memory interface. Therefore, measuring the arithmetic intensity using AMD-Xilinx's profiling tools is inevitable, since calculating the arithmetic intensity analytically is not possible.

We track this interface, since we identify it as the memory-wise weakest element in the system (up to 100 GB/s, ref. to Subsection III-A). The total bytes measured include everything necessary w.r.t. to NN execution, i.e., loading input data from DDR memory to DPU, loading instructions and weights for each layer, storing and loading the independent outputs (feature maps) to/from DDR memory, and transferring the final output result back from DPU to DDR memory.

Fig. 4(a) denotes the location for every NN variant generated in Subsection IV-A by a scattered point, which is rectangular and circular for MLPs and CNNs, respectively. Visual indicators, such as size, background color, border color, and label (only for CNN), denote the corresponding NN's architectural properties, i.e., num. of layers, num. of neurons for MLP (num. of channels for CNN), num. of inputs for MLP (input-resolution for CNN), and kernel size (only for CNNs), in this order. Their relations are visually shown in (c). All points are bounded by the maximum memory-bandwidth and the maximum compute capability of the DPU, which are indicated by a blue (grey) line for the DPU configuration C32B1 (C32B5). The following two sub-sections are limited to C32B1, while Sub-section V-C covers C64B1 and C32B5.

A. Performance of Multi-Layer Perceptrons

It can be seen, that the MLPs form a distinct group in the lower-left corner at operational intensities between 1.5-2 OPs/byte. Due to this low operational intensity, all MLPs operate in the memory-bandwidth bound domain. Depending on their specific architectural properties, MLPs are either primarily bandwidth limited (points near the blue line), i.e., they achieve their maximum theoretic performance possible, or are subject to additional bottlenecks (points below the blue line), which are not captured by the Roofline model. The primary property that decides between those two scenarios is the number of neurons per layer (blue background). A high neuron count (~ 500) leads to a performance near the theoretic limit, while a low neuron count (< 400) results in severely degraded performance. The second factor is the number of layers, with very few layers (2-4) performing worse than numerous (6+) layers. The number of inputs does not affect the performance in a noticeable way.

B. Performance of Standard Convolutional Layers

The performance of CNNs behaves in a more dynamic way compared to MLPs. The operational intensity of the CNNs benchmarked in this work stretches over 4 orders of magnitude (~ 1 to $\sim 10^4$ OPs/byte), hence, they operate in both memory-bound and compute-bound domains. In general, it can be observed, that well-performing models (near the blue line) follow the roof of the Roofline model closely, which indicates correct assumption of the architecture's theoretical limits w.r.t. memory bandwidth and compute performance.

The most important variable is the num. of channels, which primarily determines whether a model will be situated in the bandwidth-bound or compute-bound region. It can be observed, that CNNs with 1 or 64 channels are subject to extremely bad performance and are not executed efficiently w.r.t. to their theoretical maximal performances, i.e., they are far below the blue line. For a higher channel number a computational performance greater than 2 orders of magnitude is observed, shifting the models significantly closer to the system's memory and computational boundary. Second, the input resolution plays a non-negligible role in the models' performance. In both domains it can be observed, that a higher

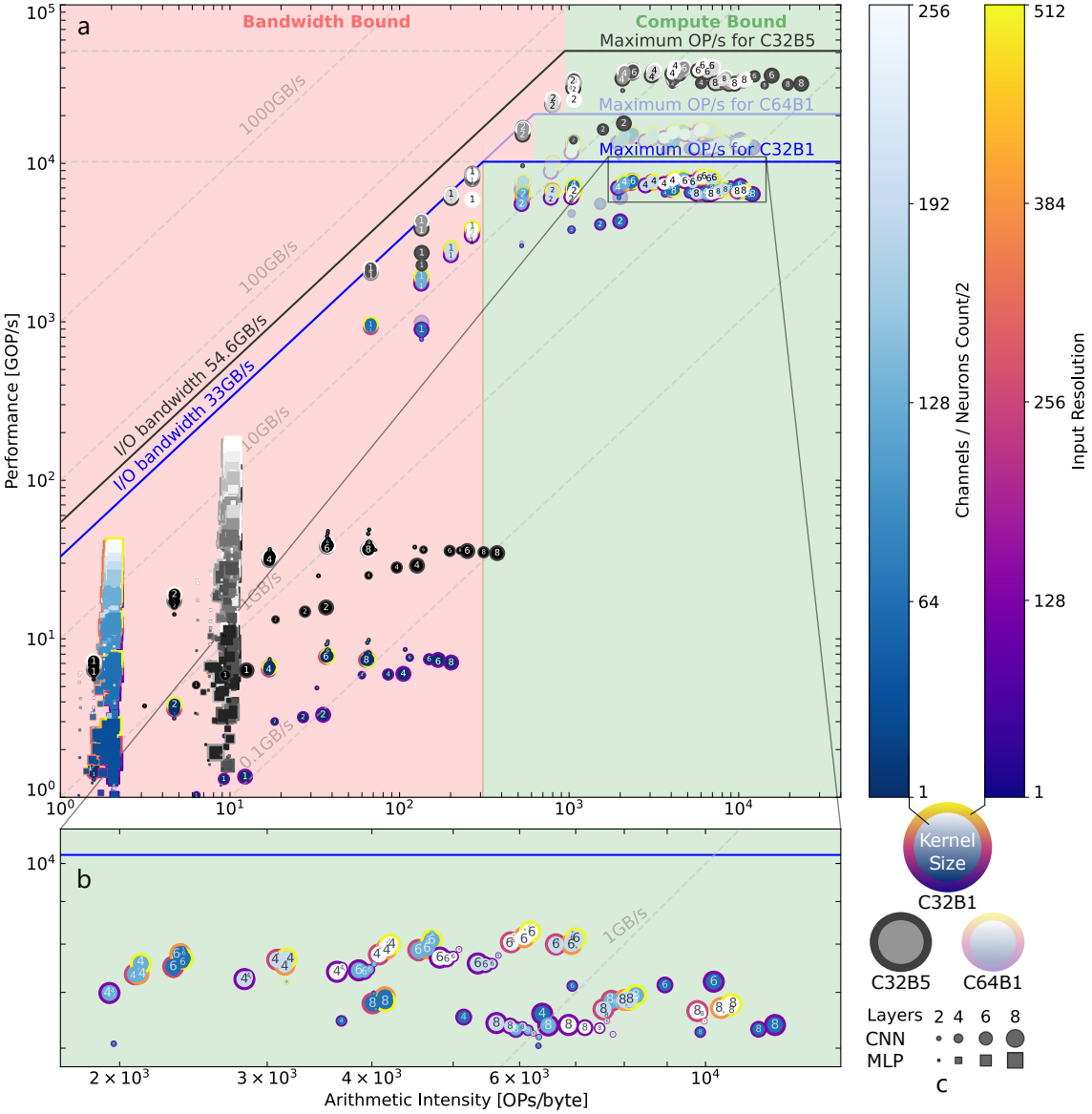


Fig. 4. Roofline model for DPU performance on the Versal, calculated for MLP and CNN NN architectures with various properties. Red area denotes memory-bandwidth bound region, green area denotes computationally bound region. The blue (grey) line denotes the maximum memory-bandwidth and the maximum compute capability for the DPU in configuration C32B1 (C32B5). Scattered rectangles and circles denote MLP and CNN architectures, respectively. Visual indicators for the points, i.e. size, background color, border color, and label (only for CNNs), denote the corresponding NN architecture as explained in (c). (a) sets the overall performance for both DPU architectures in perspective, while (b) provides a zoomed-version of the crowded computationally-saturated region of the Roofline model for the C32B1 DPU architecture, which is populated solely by CNNs.

input resolution generally raises the models' positions in the Roofline model quite significantly, although, a bigger impact is seen within the bandwidth-bound domain. The CNNs' performances are not impacted by the num. of layers.

Fig. 4(b) provides a zoomed-in view on the crowded computationally-saturated region of the Roofline model for the C32B1 DPU architecture. Within this region, all models reach between 6-8 TOPS, utilizing between 60%-80% of the DPU's maximal computational performance (10.24 TOPS). Interestingly, it can be seen, that models who only differ in their num. of layers reach higher arithmetic intensities for a lower number of layers, while the absolute computational performance remains the same. This is somewhat counter intuitive, as one might think, that the memory-wise overhead associated with one forward pass would impact the overall performance less for longer models.

C. Impact of Multi-Batch Inference and Additional AI-Engines

As stated in Subsection III-B, two additional DPU configurations, i.e., one batch handler (BH) with 64 AI-Engines, and five BHs with 32 AI-Engines per BH are analyzed. We want to provide a short summary what changes have been observed for those configurations:

a) Doubling Compute Power (DPU C64B1): For MLPs no difference at all is observed, since all models are memory-bound or suffer additional internal bottlenecks, voiding any possible benefit of additional compute capacity. We refrain from adding them to Fig. 4 for visibility reasons. CNNs show little improved performance for operational intensities $<\sim 500$ OP/s, however, after entering the compute-limited domain, additional compute capacity leads to similar relative maximal performance w.r.t. to the maximal limit (20.48 TOPS) as observed for C32B1 configuration. C64B1 CNNs are denoted by transparent points in Fig. 4.

b) Multi-Batch Inference (DPU C32B5): For all MLPs a shift of the operational intensities by about factor five is observed. This can be intuitively understood by the re-use of weights for the 5 BHs, resulting in more calculations without additional significant load on the DDR interface caused by data. Moreover, the available memory bandwidth is increased by 64% (ref. to Eq. 2), resulting in an overall higher compute performance.

CNN performance follows a different scheme. Since the memory interface is primarily occupied by intermediate data (feature maps) transfers for CNN inference and not for weights/biases, no change in operational intensity is observed, since the additional operations incurred by multi-batch execution require equally more data transfer. This also limits the performance increase for CNN models operating near the bandwidth limit. This is the reason, why certain CNN configurations have a significantly higher inference time, while others show no increase. However, in the compute limited domain, the performance rises up to a factor of five, exploiting up to 80% of the DPU's maximal compute power (51.2 TOPS).

VI. Conclusion and Outlook

In this work we performed an in-depth performance benchmark of the Deep-Learning Processor Unit on the Versal AI Core series platform. We provided a system-level analysis in terms of inference speed and power consumption for various MLP and CNN variants and ResNets, and performed an architectural profiling using AMD-Xilinx's profiling tools to understand the observed behavior better. Future work should consider analysing the so-called "Custom-Layer" feature w.r.t. overall performance, which allows to integrate unsupported operations (e.g., FFT, pre/post-processing, etc.) into the DPU dataflow for efficient execution.

References

[1] 3d Generation Partnership Project, "Release 17," https://www.3gpp.org/specifications-technologies/releases/release-17, 2023, [Online; accessed 22-January-2023].

[2] Fraunhofer Institute for Integrated Circuits IIS, "5g satellite integration," https://www.iis.fraunhofer.de/en/ff/kom/satkom/sat-5g.html, 2023, [Online; accessed 22-January-2023].

[3] M. Ghiglione and V. Serra, "Opportunities and challenges of ai on satellite processing units," in *Proceedings of the 19th ACM International Conference on Computing Frontiers*, ser. CF '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 221–224. [Online]. Available: https://doi.org/10.1145/3528416.3530985

[4] Advanced Micro Devices, Inc., "Xqr versal for space 2.0 applications," https://www.xilinx.com/content/dam/xilinx/publications/product-briefs/xilinx-xqr-versal-product-brief.pdf, 2022, [Online; accessed 22-January-2023].

[5] B. Gaide, D. Gaitonde, C. Ravishankar, and T. Bauer, "Xilinx adaptive compute acceleration platform: Versaltm architecture," in *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 84–93. [Online]. Available: https://doi.org/10.1145/3289602.3293906

[6] M. Petry, P. Gest, A. Koch, M. Ghiglione, and M. Werner, "Accelerated deep-learning inference on fpgas in the space domain," in *Proceedings of the 20th ACM International Conference on Computing Frontiers*, ser. CF '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 222–228. [Online]. Available: https://doi.org/10.1145/3587135.3592763

[7] M. Adhiwiyogo, R. D'Souza, S. Leibson, and R. Shak, "White paper: Pushing ai boundaries with scalable compute-focused fpgas," pp. 1–6, 2020.

[8] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, and A. Borchers et al., "In-datacenter performance analysis of a tensor processing unit," *SIGARCH Comput. Archit. News*, vol. 45, no. 2, p. 1–12, jun 2017. [Online]. Available: https://doi.org/10.1145/3140659.3080246

[9] E. Nurvitadhi, J. Cook, A. Mishra, D. Marr, K. Nealis, P. Colangelo, A. Ling, D. Capalija, U. Aydonat, S. Shumarayev, and A. Dasu, "In-package domain-specific asics for intel® stratix® 10 fpgas: A case study of accelerating deep learning using tensortile asic(abstract only)," in *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 287. [Online]. Available: https://doi.org/10.1145/3174243.3174966

[10] AMD-Xilinx, "Versal: The first adaptive compute acceleration platform (acap)(wp505)," https://docs.xilinx.com/v/u/en-US/wp505-versal-acap, [Online; accessed 05-July-2023].

[11] AMD-Xilinx Inc., "Dpucvdx8g for versal acaps product guide (pg389)," 2023, [Online; accessed 21-August-2023].

[12] AMD Xilinx, "Vitis ai user guide (ug1414)," https://docs.xilinx.com/r/en-US/ug1414-vitis-ai/Vitis-AI-Overview, [Online; accessed 12-July-2023].

[13] S. Williams, A. Waterman, and D. Patterson, "Roofline: An insightful visual performance model for multicore architectures," *Commun. ACM*, vol. 52, no. 4, p. 65–76, apr 2009. [Online]. Available: https://doi.org/10.1145/1498765.1498785

3.3 Discussion - Is Space Ready for AI?

In this section we address remaining challenges and clarify open points in bringing AI onto the satellite.

As already alluded to in [Pub3.A, P. 2], one major concern is the rapid pace in which AI technology changes in size and complexity [164]. This affects both hardware and software, since new NN algorithms often require adaptations to hardware to remain efficiently executable. With an average GEO satellite lifetime of 15 years [165], this discrepancy must be reflected in an option to reconfigure the hardware on a low-level, such as only FPGAs can offer. For average lifetimes of 2-4 years for modern LEO-based communication satellites, no special measures are required. From this perspective, the VERSAL platform is ideally positioned for both scenarios with the optimized but frozen AI-Engines, and reconfigurable Programmable Logic (PL) in terms of future-proof AI capabilities.

Another challenge that needs to be addressed to support both LEO and GEO deployments effectively is the discrepancy in resource requirements, which necessitates the scalability of hardware components and the efficient distribution of data flow and processing components. An important point to consider is the communication capability of the devices to exchange data among devices. Small-scale AI accelerators as listed in Sub-sec. 3.1.2, such as Brainchip's Akida, Intel's Loihi, Google/Coral's EdgeTPU, and the Hailo-8 Edge-AI chip are not intended to scale to huge workloads, since 100s of chips would be required to match the performance of GPU- and SoC-based platforms, workload distribution is not naively implemented, and communication overhead would overload communication busses. On the other side, GPUs enable distributed deployments by design and can combine available resources very effectively [166, 167]. Moreover, the I/O-interfaces on FPGAs-based SoCs provide massive bandwidth and efficient data handling across the device, such as up to 44 GTY transceivers with 32.75 GiB/s each on the VERSAL AI Core VC1902 chip, totalling 1.44 TiB/s in bandwidth.

Furthermore, one of the greatest challenges is ensuring confidence in AI/ML systems. Given the stringent verification processes outlined in standards like ESA's European Cooperation for Space Standardization (ECSS) [168] and NASA's Software Safety Assurance [169], ensuring trustworthy and certifiable AI in space is particularly challenging. These guidelines were designed for deterministic systems, making them difficult to directly apply to AI, which operates in a more opaque, partly black-box manner. One key issue is the need for full traceability and explainability of system behaviors, which is inherently at odds with how AI models function. For example, ECSS requires that all software behaviors be predictable and verifiable [168], but this is hard to achieve when the decision-making process of an AI model is not explicitly defined, often making its actions difficult to explain, especially in safety-critical applications.

Compounding this challenge is the space environment, where radiation can cause bit flips that alter the operation of AI systems, impacting both the reliability and the trustworthiness of the results [170]. These kinds of errors are particularly problematic for AI, as they can lead to unpredictable behavior and graceless degradation [171] that are difficult to test or detect in advance. Traditional verification processes are not designed

to handle the adaptive nature of AI or its susceptibility to radiation, which means we need new approaches to ensure fault tolerance. For instance, redundancy in AI processing units and fault-tolerant hardware might help, but these solutions come with added complexity and cost. From this we conclude the need for a tailored verification strategy.

Lastly, the most essential components for integrating AI on spacecrafts are appropriate ML/DL models which are designed and trained for the mission use cases. In the next chapter we explore new DL-based approaches to augment communication signal processing chains from the design process to hardware implementation and Over-the-Air testing with RF hardware.

3.4 Summary

This chapter discussed the integration of AI on spacecrafts with a focus on telecommunication satellites. It highlighted the necessity of new hardware technologies to enable on-board AI, given the limitations of current satellite platforms and payloads, which typically operate in transparent or bent-pipe modes without significant signal processing capabilities. The chapter reviewed emerging AI inference platforms suitable for space applications, including CPU-, GPU-, FPGA-, and ASIC-based solutions, and evaluated their key performance indicators such as processing capacity, power efficiency, and space-grade resilience. AMD-Xilinx's VERSAL AI Core platform is analyzed and identified as a promising solution for future-proof satellite payloads. The chapter concluded with an exploration of future challenges, including hardware scalability, data distribution, the necessity to rethink the verification process, and appropriate algorithm design.

4 Deep Learning-augmented Physical Layer Communication on the Edge in Space

In this chapter we explore how ML can be applied to communications for the task of physical layer signal processing while targeting practical realizations on AMD-Xilinx's VERSAL AI core hardware platform. We follow two approaches: First, [Pub4.A] proposes the concept of an on-board flexible and reconfigurable physical layer via means of an E2E "AI-designed" communication system. By realizing all signal processing steps, such as given in Sub-sec. 2.1.3, solely utilizing jointly trained NNs, we make the physical layer updatable and exchangeable via simply switching out weights or the NNs itself. An SDR-based demonstration with hardware-accelerated AI inference on the VERSAL validates the functionality, but also reveals shortcomings of the fully-AI-native concept, which are discussed in Sub-sec. 4.1.1.

This leads to an alternative approach: With a more efficient implementation in mind, [Pub4.B] shifts the focus to "AI-augmented" communication systems and investigates a hybrid approach that tightly couples classical- and DL-based components to address more selective tasks instead of taking on the whole workload. In [Pub4.B] we target the task of RF signal synchronization and propose a hybrid architecture that utilizes DL techniques to extract information, such as CFO, CPO, and STO from the received signal, while their compensation is performed using traditional methods.

Lastly [Pub4.C] explores ways to efficiently implement the hybrid architecture proposed in [Pub4.B] on the VERSAL by distributing processing steps to their optimal domain in this heterogeneous setup. It becomes clear that communication overhead between different computing domains becomes a major performance bottleneck if not properly handled, hence, a tailored data-flow and buffering system is devised.

A succinct discussion on both approaches introduced above reviews the practicability of realizing the proposed and other AI-oriented solutions on telecommunication satellite payloads and aims to generalize their compatibility to ongoing research in the communications domain.

4.1 Pub4.A: Envisioning Physical Layer Flexibility Through the Power of Machine-Learning

Authors: Michael Petry, Andreas Koch, Martin Werner

Publication Medium: IEEE Globecom Workshops 2023

Digital Object Identifier: https://doi.org/10.1109/GCWkshps58843.2023.10464871

Thesis contextualization: This paper initiates the first approach in this dissertation on forming a bridge between general research on AI/ML in wireless communications to their practical implementations on realistic hardware in the space domain. With the goal of performing a primary evaluation on the feasibility of bringing AI-heavy algorithms onto modern hardware platforms like the VERSAL, we leverage the concept of the bit-wise autoencoder [129] architecture to implement a simple E2E-trained communication system, addressing [RQ-1], [RQ-3], and [RQ-4]. Although an experiment validates the functionality of the concept, it reveals a lack in achievable throughput-performance and overall accuracy of the system which triggers a discussion on optimized approaches in this chapter and introduces a transition to AI-augmented systems.

CRediT author statement:

Michael Petry:	80 %	Investigation, Conceptualization, Methodology, Software, Data curation, Formal analysis, Validation, Visualization, Writing
Andreas Koch:	10 %	Investigation, Software, Writing
Martin Werner:	10 %	Investigation, Conceptualization

2023 IEEE Globecom Workshops (GC Wkshps): Artificial Intelligence Enabled Next Generation Wireless Networks

Envisioning Physical Layer Flexibility Through the Power of Machine-Learning

Michael Petry
Telecom Processing Germany
Airbus Defence and Space GmbH
Munich, Germany
michael.petry@airbus.com

Andreas Koch
Telecom Processing Germany
Airbus Defence and Space GmbH
Munich, Germany
andreas.c.koch@airbus.com

Martin Werner
Professorship of Big Geospatial Data Management
Technical University of Munich
Munich, Germany
martin.werner@tum.de

2023 IEEE Globecom Workshops (GC Wkshps) | 979-8-3503-7021-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/GCWKSHPS58843.2023.10464871

***Abstract*—This paper presents the vision of an adaptive radio frequency (RF) communication signal processing pipeline solely composed of machine learning domain operations, aiming to provide a fully hardware-accelerated alternative to dedicated RF chips. A hybrid architecture, comprising elements of classic signal processing and learnable algorithms trained in an end-to-end manner, is proposed, that is compatible with contemporary ML hardware accelerators. The RF-ML pipeline, including speed-up optimization modifications, are explained in detail, followed by a brief description summarizing the deployment workflow of the end-to-end system on a pair of AI-enabled space-grade FPGAs. Finally, a bit-error-rate performance study of the simulated system as well as a HW-deployed setup including software-defined radios (SDR) validates the concept, followed by a detailed throughput benchmark over multiple AI-accelerator hardware configurations. Finally, we raise questions regarding practical implementation, such as receiver synchronization, restrictions of the ML-accelerator feature space, and weight quantization, which are discussed at the end of this paper.**

***Index Terms*—signal processing, physical layer, machine learning, hardware acceleration, fpga, software-defined radio**

I. Introduction

Today's requirements on communication systems might not be the same as tomorrow's. This is best understood by the example of the 5G NR mobile communication standard. Starting from its initial version (Rel. 15) in 2017, new iterations are continuously passed that comprise changes/extensions to the physical layer. With Release 18 expected to pass at the end of 2023, it is clear, that at this rate of updates the broad majority of users cannot keep up with state-of-the-art equipment, hence, being "outdated" has become the new normal. This lack of technological advancement in the communications field can be traced back to hardware, specifically to specialized RF-chips or -ASICs not being upgradable w.r.t. physical layer modifications, making devices obsolete on a yearly basis. This conflicts heavily with the idea of sustainability. We envision to change this by combining the idea of a re-programmable physical layer hardware with the current mega-trend of machine learning and its available hardware accelerators. By not only exploiting the idea of augmenting the RF-pipeline with AI-algorithms, but also using the available AI-hardware accelerators for power-efficient and high-speed computation, we propose a system design that can combines various potential RF performance gains over traditional systems as shown in [1]–[4] with physical layer flexibility and adaptability unknown to compact mobile devices.

This work is supported by the German Aerospace Center (DLR) under project Machine Learning on Telecommunications Satellites (MaLeTeSa) with Grant number 50 YB 2103.

This paper builds on the idea of a learned communications system as proposed by O'Shea et al. [1] in 2017, which sparked the interest of applying machine learning in the RF communications field. Follow-up works primarily focused on extending the system's capabilities, e.g., bit-wise autoencoder [3], multiple antenna system [5], multi-carrier modulation schemes such as OFDM [6], neural equalization [7], graph neural network-based channel decoding [8], etc. However, all systems were either proposed in simulation or realized on impractical GPU-accelerated systems to fulfil computational demands, leaving the idea of AI-based communication practically infeasible for resource restricted platforms. The main contributions of this paper are the following: We aim at closing the gap between computer simulations and practical deployment by realizing a practically feasible non-GPU based hardware setup that builds on a power-efficient FPGA-based AI accelerator, but can be generalized to any compatible AI accelerator. Furthermore, we demonstrate continuous communication with our autoencoder implementation.

The rest of this paper is structured as follows: Section II details the designed RF-ML pipeline including the end-to-end training strategy for its learnable components. Section III briefly introduces the AI-optimized FPGA platform and its workflow including necessary modifications for transforming the generic NN model to a hardware compatible form. Validation of the proposed system, including a detailed BER- and throughput-performance benchmark, by using software-defined radios as RF-frontend to the FPGA platform is detailed in Section IV. Lastly, Section V concludes this work and discusses discovered challenges for future work.

II. System model

This work assumes a single-input single-output (SISO) system model with single-carrier modulation. The objective of the system is to transmit a *continuous* stream of binary data unidirectional from the sender to the receiver through a RF communications channel. This involves performing common signal

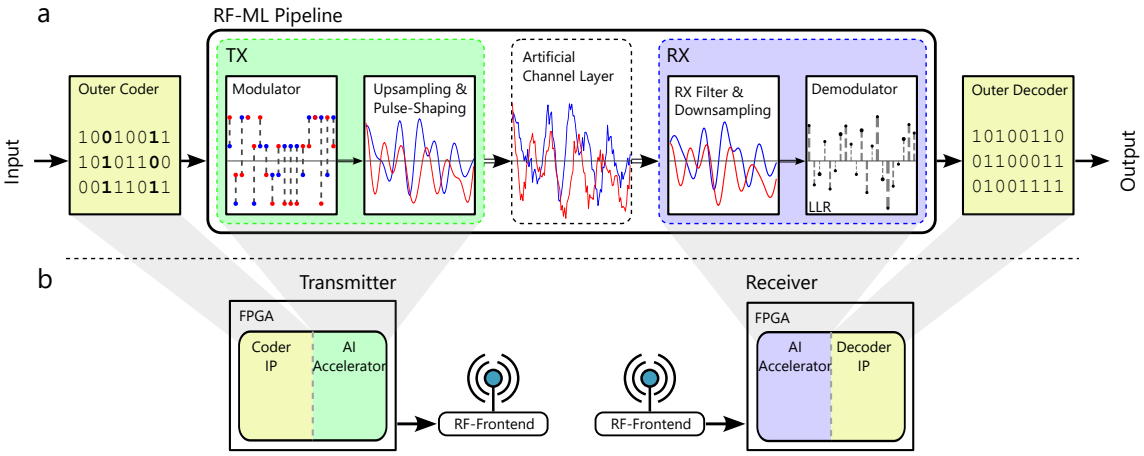


Fig. 1. System architecture of an AI-based hardware-accelerated (FPGA-based) RF-communication system. (a) Autoencoder-based NN implementation; (b) HW-accelerated deployment strategy. Dataflow from left to right. Transmitter-side: Binary data is encoded by an outer encoder implemented as soft-IP (yellow); encoded data is processed by HW-accelerated Transmiter-NN (green); output is propagated to RF-Frontend. Receiver-side: RF-Frontend sends received I/Q-samples to HW-accelerated Receiver-NN (purple); demapped LLRs are decoded by outer decoder implemented as soft-IP (yellow).

processing steps such as Coding, Modulation, Up-sampling, and Pulse-Shaping to generate the physical-layer baseband RF signal on the transmitter side, as well as their respective inverse operations to retrieve the transmitted information from the received signal on the receiver side.[1] Since this work envisions to perform all prior mentioned steps fully within the ML-domain, those steps are framed in the so-called "RF-ML Pipeline" as depicted in Fig. 1, which is a single neural network comprising different layers performing those tasks. We note, that coding and decoding is not implemented within the neural network as it suffers of the *curse of dimensionality* as discussed in [8]. Instead, it is implemented as pre- and post-processing steps by utilizing an Intellectual Property (IP) block[2] deployed on the Programmable Logic (PL). We differentiate between trained, i.e., (De-)Modulation, and non-trained, mathematical, i.e., Up-sampling and Filtering, components. The system is implemented using a bit interleaved coded modulation (BICM) auto-encoder, hence training is performed in an end-to-end (E2E) manner by minimizing the total binary cross entropy between input- and output-bits as follows, resulting in a Bit-Error-Rate (BER) minimization [3]:

$$L(\theta_M, \theta_D) := \sum_{j=1}^{m} \mathbb{E}[H(p_{\theta_M}(b_j|y), \tilde{p}_{\theta_D}(h_j|y))] \quad (1)$$

$$= \sum_{j=1}^{m} \mathbb{E}_{y,b_j}[-\log\, p_{\theta_M}(b_j|y)] \quad (2)$$

[1]Note: For our hardware experiment we assume a synchronized system by hard-wiring the local oscillators of both transmitting and receiving RF-Frontends, therefore synchronization procedures are out of scope for this work.

[2]LDPC Encoder / Decoder: https://www.xilinx.com/products/intellectual-property/ef-di-ldpc-enc-dec.html

$p_{\theta_M}(b_j|y)$ denotes the probability of the j^{th} output-bit, which is obtained by applying the sigmoid function to the corresponding logit. An artificial channel layer functions as a bottleneck by restricting the amount of information conveyed with one channel use (symbol), forcing the system to adapt it's properties (here: symbol constellation).

Following the training of the system, the channel layer is removed and the NN is split into TX (green) and RX (purple) parts as depicted in Fig. 1, which are then individually deployed on the ML hardware accelerator on both devices. Concatenation of outer (de)-coding completes the RF pipeline.

The following subsection explains the NN realisation of these computation steps in detail:

A. Neural Network RF-ML Pipeline Implementation

Fig. 2 visualizes the E2E data-flow through the processing chain from input bits to RF-signal and backwards.

1) First, the (coded) input bit-stream is modulated by mapping sub-vectors b of length m to a complex baseband symbol c by using a Dense Neural Network (DNN), also known as fully-connected dense layers. Hence, the DNN implements the parametrized function $f_M(\theta_M, b)$ performing the mapping $f_M : \{0, ..., 2^m - 1\} \mapsto \mathbb{C}$ with trainable parameters θ_M, resulting in modulation order m. In practice, two real-valued outputs are produced for one complex number. The required depth and inner size of the network depends on m and the constellation shape complexity. A shift from sequential to batch-wise modulation, e.g., to boost throughput, is achieved by formulating the DNN operations using equivalent convolutional (CNN) operations, which allows for spatial parallelism in two dimensions, resulting in a modulated output tensor $\mathbf{M} \in \mathbb{R}^{N_x \text{x} N_y \text{x} 2}$.

2) Up-sampling of the baseband symbols $\mathbf{M}$ with upsampling-factor u_{TX} in the dimension of sequential symbols (here: x-direction) is achieved by applying a transpose depth-wise 2D-convolution with one-hot kernel = $\{1, 0, ..., 0\} \in \mathbb{N}^{u_{\mathrm{TX}} \mathrm{x} 1}$ and stride u_{TX}, resulting in an output tensor $\mathbf{U} \in \mathbb{R}^{(N_x \mathrm{x} u_{\mathrm{TX}}) \mathrm{x} N_y \mathrm{x} 2}$.
3) Finally, transmit-pulse shaping is achieved by applying a depth-wise 2D convolution with "same"-padding and odd-length kernel $k \in \mathbb{R}^{l \mathrm{x} 1}$ (filled with filter coefficients, e.g., root-raised-cosine) on $\mathbf{U}$, resulting in the I/Q sample tensor $\mathbf{O} \in \mathbb{R}^{(N_x \mathrm{x} u_{\mathrm{TX}}) \mathrm{x} N_y \mathrm{x} 2}$.[3] N_x and N_y can be considered hyper-parameters to the RF-ML-Pipeline, since they both effect the numerical results of the filtering operation and also the underlying degree of compute parallelization, hence, achievable throughput.

Lastly, the tensor is flattened in the direction of sequential samples and then fed to the RF-Frontend.

On the receiver side, in principle the same operations are performed in an inverse manner. RX processing works on a sampled signal tensor $\mathbf{R} \in \mathbb{R}^{(N_x \mathrm{x} u_{\mathrm{RX}}) \mathrm{x} N_y \mathrm{x} 2}$ using an up-sampling factor of u_{RX} in the receiving RF-Frontend.

1) First, the received samples will be filtered using a matched-filter by applying the same operation as in 3) in the transmitter-side on $\mathbf{R}$.
2) The filtered samples are down-sampled by factor u_{RX} by applying a depth-wise 2D-convolution with stride u_{RX} and one-hot kernel of length u_{RX}, with the 1 at the position corresponding to the symbol-centered sample.
3) Demodulation from a symbol-centered I/Q-sample s to a Log-Likelihood-Ratio (LLR) vector l of length m is performed using a DNN implementing the function $f_D(\theta_D, s)$, with $f_D : s \mapsto \mathbb{R}^m$. Similarly as in 1) in TX-side, a CNN is used to perform the same operation in parallel, resulting in a LLR tensor $\mathbf{L} \in \mathbb{R}^{N_x \mathrm{x} N_y \mathrm{x} m}$.

The LLR tensor is then flattened and fed to the decoder.

B. Training

The auto-encoder system is trained using a variant of stochastic gradient descent (Adam). The training parameters are summarized in Tab. I.

TABLE I
TRAINING PARAMETERS

Parameters	Value
Loss	Binary Cross Entropy
Num. Epochs	25.000
Batch-Size	1000 bits
Training Algorithm	Adam
Learning rate	starting at 10^{-1} and decreasing by /5, /10, /100, /1000 every 5000 epochs
Training SNR	Fix per constellation
Weight Initialization	Glorot

[3]We want to note, that pulse-filtering using a 2D-convolution does not exactly replicate the original operation working on a 1D data array, since the convolution is not performed in one run, but on N_y slices of the data. This results in small dissimilarities in the neighborhood of every N_y samples whose consequences are discussed in chapter IV.

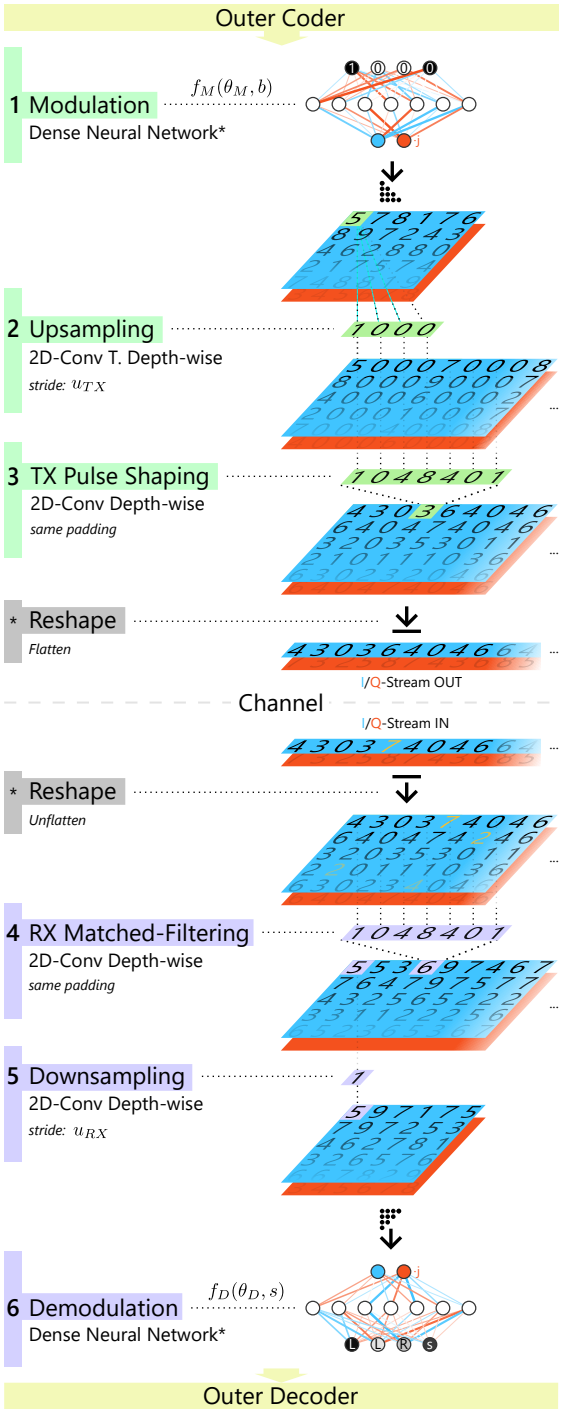


Fig. 2. Detailed implementation of the RF-ML-Pipeline NN.

Constellation energy normalization is performed after the modulator and is not inherently learned by the DNN, since no additional constraint (e.g., impact of energy on the loss) is applied. For long-term training this leads to an exploding output-range of the modulator. Similarly, symbol centering is applied outside the DNN, leading to a non-negligible offset to the DNN's output. These effects have to be taken into account for quantization and HW-model compatibility, as described in Sub-section III-D.

III. Algorithm Deployment

This section provides an overview of the utilized AI-accelerated hardware platform and its deployment process on which the prior introduced RF-ML pipeline is later benchmarked. Remarks regarding compatibility issues and necessary modifications conclude this section.

A. AI-Accelerator overview: AMD-Xilinx Versal

The Versal platform of the manufacturer AMD-Xilinx presents a powerful series of embedded System-on-Chips (SoC) that target a multitude of different computational-demanding embedded tasks, with a particular focus on high-speed ML inference [9]. The Versal AI Core series is one of the state-of-the-art ML hardware accelerators, featuring a novel architecture by combining three different computing paradigms, i.e., processing system / CPU (PS), programmable logic (PL) / FPGA, and a third class of vector compute engines. In the context of ML, these compute engines are referred to as AI-Engines and carry the main computational workload. This work utilizes the VCK190 development board, which ships with 400 AI-Engines. Fig. 3 provides a coarse overview over its computing resources.

To exploit full capabilities of all computing resources in concert, the manufacturer provides a generic inference accelerator in the form of an IP block to be synthesised and deployed on the Versal, as well as a tool-chain to compile deep-learning models from standard formats such as HDF5 and ONNX to hardware-suitable instructions. The interplay of both tools and their deployment workflow is described in the following subsection.

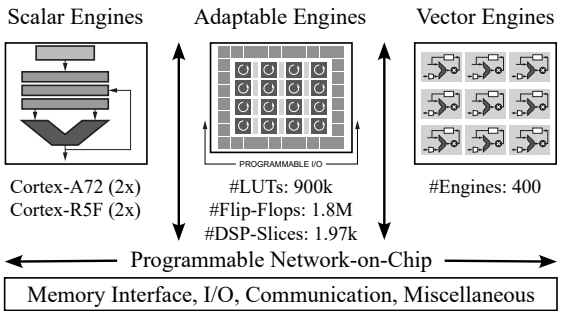


Fig. 3. Computing resources on the Versal platform, comprising scalar (CPU), adaptable (FPGA), and vector (AI-optimized) engines.

B. Generic Inference Accelerator

The generic inference accelerator, from now on also referred to as Deep-Learning Processor Unit (DPU), can be described as a microcoded co-processor with an instruction set optimized for NN inference. The IP is able to execute various ML-architectures dynamically by loading compiled model-files and updating the computation pipeline on PL and AI-Engines on-the-fly without any reprogramming. Configuration options exists that allow trading of its compute power (e.g., through the underlying level of parallelism) vs. hardware resource utilization. In this work, various configurations regarding hardware utilization of the FPGA, num. of AI-Engines, and batch parallelism are analyzed regarding achievable performance. Tab. II summarizes the considered configurations.

TABLE II
Analyzed DPU hardware utilization configurations.

Configuration	Freq.[a] [MHz]	FPGA [%]	AIE [#]	Batch [#]	Int-8 Perf [TOPS]
C32B1	325/1250	6%	32	1	10.24
C32B3	325/1250	15%	96	3	30.72
C32B5	325/1250	24%	160	5	51.20
C64B5[b]	325/1250	26%	320	5	102.4

[a]Frequency of PL / AI-Engines
[b]Maximum Performance Configuration

C. Deployment Workflow

The Deep-Learning Processing Unit expects a hardware-compatible model for inference. This subsection summarizes the steps for transforming a generic ML model to a hardware executable one using the tool-chain Vitis-AI. The deployment workflow can be divided into three categories.

a) Compatible ML Application: The DPU executes ML operations jointly on PL and AI-Engines in a highly efficient and hardware-tailored manner, hence only a subset of operations is supported [10]. To guarantee computation fully within the DPU, it is strictly necessary to ensure compatibility of the designed AI application to the DPU. Both operation type and parameters (e.g., kernel size, padding) are limited. However, unsupported features can also be executed on the PS, which in turn requires the exchange of intermediate data between DPU and PS, imposing a noticeable performance penalty to the application.

b) Model Quantization: DPU inference is based on 8-bit fixed-point computation, since FPGA-based systems benefit from a lower power-consumption, latency [11], and hardware complexity by realizing fixed-point units compared to floating-point units. Even though the AI Engines support floating-point based vector operations, fixed-point computation is more efficient due to their particular high Int-8 compute performance [12]. Vitis-AI performs 8-bit quantization of inputs, weights, biases, and activations after training using a power-of-two quantization scheme, a special form of quantization well-suited for FPGAs [13]. While this scheme is referred to as Post-training quantization (PTQ), it is also possible, though not used

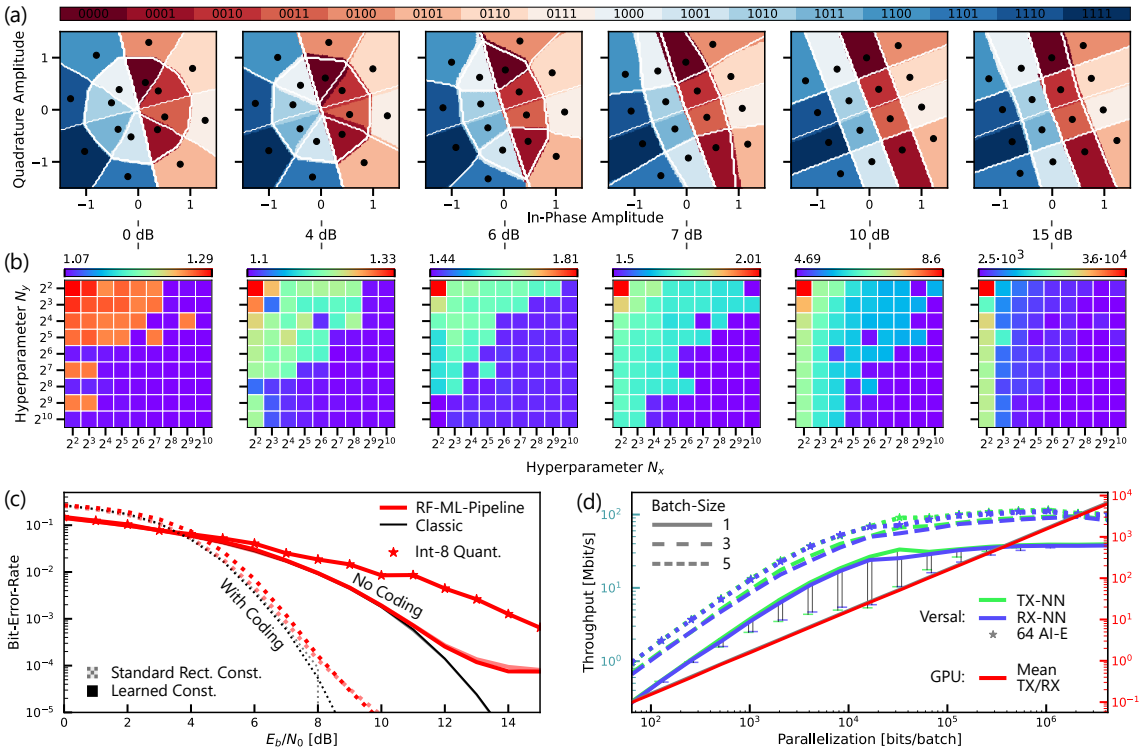


Fig. 4. BER- & Throughput-Performance of the proposed RF-ML-Pipeline. (a): Learned constellations (dotted points) and corresponding demapper decision regions for selected BERs for hardware quantized (shaded area) and Float-32 (white lines) computation. (b): BER ratio of uncoded HW-deployed RF-ML-pipeline vs. Float-32 "classic" computation for all power-of-two parallelization combinations in [2; 10]. (c): Simulated BER performance of coded (dotted) and un-coded (solid) system configurations with classic (black) or 256x256-RF-ML-Pipeline (red, thick) unquantized computation and learned (opaque) or standard rect. 16-QAM (semi-transparent) constellations. Line with star-marker denotes HW-measured BER performance. (d): Throughput analysis of RF-ML-Pipeline (blue: TX-NN, green: RX-NN, left y-axis) on Versal hardware platform for batch-sizes [1,3,5] vs. GPU throughput (red, right y-axis) for batch-size 1.

in this work, to perform Quantization-aware training (QAT), where the model is further trained by emulating inference-time quantization, creating a more robust model.

c) Model Compilation: Finally, the quantized NN is compiled into DPU-compatible instructions by invoking the Vitis AI Compiler. This generates a program consisting of quantized data and microcode for the DPU, which can be loaded from the file system into memory and transferred to the co-processor.

D. Model compatibility adaptation

The proposed RF-ML pipeline NN implementation as depicted in Fig. 2 is operation-wise fully compatible with the feature support of the DPU. However, limitations w.r.t. parameters, e.g., kernel dimensions of $w, h \in \{1, ..., 15\}, wxh \leq 64$ for 2D-convolutions, apply, which can result in degraded system performance (here: restricting filter taps to a maximum of 15). Circumventing these limitations can pose significant implementation challenges. In this work, the kernel length limitation has been overcome by splitting the original filter into multiple, smaller filters and concatenating the partial results Moreover, quantization strategies can require zero-centered outputs to exploit full Int-8 bit-width. Hence, the mapper DNN's output offset, as described in Sub-section II-B, is compensated by manually adjusting the bias of its last layer.

IV. System Validation and Performance Analysis

In this section we validate the functionality of the proposed RF-ML pipeline in simulation and on hardware through comparison to its "equivalent" classic implementation for various combinations of both hyper-parameters N_x and N_y. Additionally, the processing capabilities and implementation efficiency of the NN on the Versal platform w.r.t. throughput are benchmarked for a variety of hardware configurations and compared to a modern GPU-based setup.

To analyze the BER performance of multiple configurations of the RF-ML-Pipeline over a suitable signal-to-noise power range, we implement both AI-based and classic baseline systems with *modulation order* $m = 4$, wrap them inside a Low-density parity-check *(LDPC)* encoder-decoder pair with

code-rate $c_r = \frac{1}{2}$ and codeword length $c_l = 80$ if applicable, and train them to specific SNR values each. For pulse-shaping and matched-filtering a root-raised-cosine *(RRC)* filter with roll-off-factor $\beta = 0.22$ and odd length spanning 22 symbols is used with up-sampling factors $u_{\text{TX}} = u_{\text{RX}} = 2$. Mapper and demapper are 1-hidden layer DNNs with 128 neurons each. The classic systems are modeled using the Sionna framework [14]. The results are displayed in Fig. 4.

Fig. 4(a) visualizes the learned constellations (black dots) and the corresponding demapper decision regions (area: quantized; white line: unquantized) as a function of the SNR. It can be seen that the match of quantized and unquantized decision regions is good, however, differences exist that will result in misclassification, degrading quantized performance slightly.

Fig. 4(b) denotes the BER ratio of the uncoded HW-deployed RF-ML-Pipeline (with two Ettus X300 USRPs over a coaxial-cable channel) vs. simulated "classic" baseline as a function of the hyperparameters N_x and N_y and the SNR. While the HW-deployed performance is always worse than the simulated classic implementation, the ratio shows a strong dependency on the hyperparameters, decreasing approximately symmetrically with the increase of either one. This effect is caused by the data anomaly introduced by the "row-wise" filtering and concatenation operation in the TX- and RX-NN, introducing a systematic numerical error compared to classic filtering, leading to a decreased BER. This impact shows a high correlation with the SNR. Moreover, for high SNRs N_x has a stronger impact than N_y, which is simply caused by the fact that the numerical error can be eradicated by increasing the filter-row's length, i.e., N_x, to infinity.

Fig. 4(c) compares the BER-curves of *simulations* for various system configurations that can be divided in three categories: No-coding (solid) vs. coding (dotted), RF-ML-Pipeline computation ($N_x = N_y = 256$, red, thick) vs. classic computation (black), and learned mapping/demapping (opaque) vs. standard 16-QAM (semi-transparent). Firstly, the choice of constellation does not noticeable impact the performance, verifying the learned constellation's quality. Secondly, one can observe the RF-ML-Pipeline's BER degredation for high SNRs caused by the above described systematic error. Coding mitigates this issue only partially as visible for SNRs between 8-10 dB. This might be caused by the distribution of bit errors caused by the systematic numerical error, which is not uniform but exhibits spikes at a repeating rate determined by N_x. Lastly, the BER curve of the HW-deployed system is denoted by the star-marked line for comparison.

Finally, Fig. 4(d) shows throughput performance in MBit/s of the RF-ML-Pipeline implementation on the Versal AI-accelerator for multiple hardware configurations as a function of the number of bits processed per batch. Since multiple combinations of N_x and N_y can yield the same total parallelization, only the best performing value is chosen. The vertical lines for batch-size 1 configuration denote the respective throughput ranges for a given parallelization factor. A strong correlation to the degree of parallelization is observed. Both TX- and RX-NN perform similar, as they feature the same type of operations. RX-NN is slightly slower, which can be traced back to different data load and store speeds of the DPU. An increase in the batch-size is reflected on the throughput with a mean increase by factor 2.3 for $1 \rightarrow 3$ and 3.0 for $1 \rightarrow 5$. Surprisingly, increasing the AI-Engines from 32 to 64 for the batch-size 5 configuration only yields a 1.4% throughput gain. In general, throughput saturates for high parallelization values, topping out at 116 MBit/s. For comparison, GPU throughput performance of a NVidia V100 (AWS ml.p3 instance) for batch-size 1 is denoted by the red line, and exhibits a linear performance with a maximum of 6.8 GBit/s.

V. Conclusion and Future work

In this work we proposed and implemented our vision of a flexible, reconfigurable physical layer for RF communications through the use of generic AI hardware accelerators. The performance analysis w.r.t. BER shows promising results but hints improvements both for NN implementation and hardware-tailored deployment (e.g., quantization-aware training), indicating the necessity for follow-up work. Moreover, extending the RF-ML-Pipeline's capabilities, e.g., synchronization procedures, channel equalization, etc., is required for practical operation. Lastly, exploring the landscape of RF-targeted AI-accelerators, e.g., the Versal RF SoC, is subject of future work.

References

[1] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563–575, 2017.
[2] S. Dörner, S. Cammerer, J. Hoydis, and S. t. Brink, "Deep learning based communication over the air," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 132–143, 2018.
[3] S. Cammerer, F. A. Aoudia, S. Dörner, M. Stark, J. Hoydis, and S. ten Brink, "Trainable communication systems: Concepts and prototype," *IEEE Transactions on Communications*, vol. 68, no. 9, 2020.
[4] A. Felix, S. Cammerer, S. Dörner, J. Hoydis, and S. Ten Brink, "Ofdm-autoencoder for end-to-end learning of communications systems," in *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2018, pp. 1–5.
[5] S. Dörner, S. Rottacker, M. Gauger, and S. t. Brink, "Bit-wise autoencoder for multiple antenna systems," in *2021 17th International Symposium on Wireless Communication Systems (ISWCS)*, 2021.
[6] M. B. Fischer, S. Dörner, S. Cammerer, T. Shimizu, H. Lu, and S. T. Brink, "Adaptive neural network-based ofdm receivers," in *2022 IEEE 23rd International Workshop on Signal Processing Advances in Wireless Communication (SPAWC)*, 2022, pp. 1–5.
[7] W. Xu, Z. Zhong, Y. Be'ery, X. You, and C. Zhang, "Joint neural network equalizer and decoder," in *2018 15th International Symposium on Wireless Communication Systems (ISWCS)*, 2018, pp. 1–5.
[8] S. Cammerer, J. Hoydis, F. Aoudia, and A. Keller, "Graph neural networks for channel decoding," 12 2022, pp. 486–491.
[9] Advanced Micro Devices, Inc., "Versal architecture and product data sheet: Overview," https://docs.xilinx.com/v/u/en-US/ds950-versal-overview, 2022, [Online; accessed 25-May-2023].
[10] AMD-Xilinx, "Dpucvdx8g for versal acaps," https://docs.xilinx.com/r/en-US/pg389-dpucvdx8g/Introduction, 2022.
[11] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, and M. van Baalen, "A white paper on neural network quantization," 2021.
[12] A. Core and A. K. Gupta, "Architecture apocalypse dream architecture for deep learning inference and compute-versal ai core," 2020.
[13] D. Przewlocka-Rus, S. S. Sarwar, H. E. Sumbul, Y. Li, and B. D. Salvo, "Power-of-two quantization for low bitwidth and hardware compliant neural networks," 2022.
[14] J. Hoydis, S. Cammerer, F. Ait Aoudia, A. Vem, N. Binder, G. Marcus, and A. Keller, "Sionna: An open-source library for next-generation physical layer research," *arXiv preprint*, Mar. 2022.

4.1.1 Discussion on Fully AI-Native Physical Layers

The integration of AI/ML in the physical layer of communication systems is particularly driven through advancements of current research on 5G and 6G [172]. Since AI in the space-domain, particularly for high-throughput communication satellites, is still in its baby shoes, we align our discussion on the current state of research in 5G and 6G. Since one of the main objectives of 6G is to eradicate the boundary between Terrestrial Networks (TNs) and NTNs completely and realize one joint network [173], this makes a contextualization highly relevant, since in such case the air interface must be harmonized. As already discussed in Sub-sec. 2.3.1, AI is applied across all layers in the communication stack. Particularly in the physical layer the level of AI integration follows a pattern. Based on a vision for 6G, three stages of integration can be identified [174]:

1. AI-supported air interface: Enhancing or replacing individual classical processing blocks with ML.
   Example: Replacing a diversity combining, channel equalization, or classical demodulation algorithm with separate NNs.

2. AI-augmented air interface: Replacing groups of adjacent classical processing blocks jointly with ML.
   Example: Replacing the above mentioned blocks with one jointly trained NN.

3. AI-native air interface: ML designs parts or the full air interface itself.
   Example: Replacing the full transmitter and receiver chain by two jointly optimized and trained NNs.

To classify the systems developed in this dissertation more precisely, we extend this concept by considering a one-sided or two-sided use of ML, i.e., only on RX or TX, or on both sides.

The RF-ML-pipeline presented in [Pub4.B] is an example of a fully AI-native physical layer implementation (when neglecting the classical coding component). More precisely, the autoencoder is essentially comprised of one NN on TX- and RX-side, which itself contain different layers, such as dense and convolutional layers.

Although this approach brings in distinct benefits, such as the combination of complete reconfigurability [Pub4.A] while simultaneously running hardware-accelerated due to advancements in hardware technology [174], it also exhibits several shortcomings as discussed now.

A fully-learned architecture inherently lacks the ability to directly integrate expert domain knowledge in the form of algorithms or information into the model. Only via indirect methods, for instance by applying custom loss functions, can a model be forced to mimic certain behavior or store information. For RF signal processing this is a particular burden, since lots of algorithms utilize essential building blocks, such as interpolation, filtering, arithmetic with complex numbers, software PLLs, etc., which must be emulated by the NN often multiple times. As shown in [Pub4.A], even for native operations such as

filtering using time-domain convolution, this can lead to degraded performance compared to standard implementations.

A more practical restriction is introduced by the AI-accelerators' limitations on supported layers and maximum model size, which can easily limit the models' expressiveness, introducing blocking points when addressing certain tasks. For instance, AI-based Error Correction Coding (ECC) for practical codes requires sophisticated architectures, such as GNNs (not natively supported on the VERSAL), to tackle the curse of dimensionality [129]. Furthermore, an exponential training complexity also leads to integrating NNs into existing decoder structures [108, 175]

Lastly, the processing capacity and power efficiency of modern AI-accelerators is not as mature as telecommunication-optimized ASICs and DSPs (e.g., comparing the VERSAL's AI-engines with Satixfy's SX4000 SDR ASIC), which renders AI-accelerators too insufficient and power-hungry to take on the full DSP workload. The insufficient throughput is clearly observed in [Pub4.A, P. 5].

Overall, for this disruptive vision of replacing the complete physical layer with NNs to become practical, AI-hardware-accelerators and -models need to mature further. Hence, this integration level is unlikely to happen in the near future [176]. Instead, a more nuanced architecture that allows to combine domain knowledge, e.g., classical algorithms, with selected ML components into the physical layer, is appropriate to address the above shortcomings.

In the next section [Pub4.B] realizes this approach.

4.2 Pub4.B: Auto-Regressive RF Synchronization Using Deep-Learning

Authors: Michael Petry, Benjamin Parlier, Andreas Koch, Martin Werner

Publication Medium: IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN), 2024

Digital Object Identifier: https://doi.org/10.1109/ICMLCN59089.2024.10624754

Thesis contextualization: In this paper we take a more refined approach in utilizing AI/ML for wireless communications based on the learnings from [Pub4.A]. Instead of realizing the complete signal processing chain with Neural Networks, we apply them selectively for specific steps to address the problem of RF signal synchronization. Based on the Radio Transformer Network (RTN) architecture [24, 177] we pair domain knowledge in the form of classical algorithms, such as Fast Fourier Transform (FFT), complex multiplication, and matched filtering (among others), with small Multi-Layer Perceptrons to realize End-to-End (E2E) optimized estimate-and-compensate blocks to address fine Carrier Frequency Offset (CFO) (i.e., Constant Phase Offset (CPO)) and Sampling Time Offset (STO) compensation and demodulation with soft-information ([RQ-3], [RQ-4]). This forms the foundation of a hybrid processing algorithm which is brought to hardware in the subsequent publication [Pub4.C].

CRediT author statement: This publication includes partial results of the Master's Thesis by the co-author Benjamin Parlier [178], which has been supervised by Michael Petry.

Michael Petry:	40 %	Investigation, Conceptualization, Methodology, Software, Data curation, Formal analysis, Validation, Visualization, Writing
Benjamin Parlier:	40 %	Investigation, Conceptualization, Methodology, Software, Data curation, Formal analysis, Validation, Visualization
Andreas Koch:	10 %	Software, Writing
Martin Werner:	10 %	Investigation, Conceptualization, Methodology

2024 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN)

2024 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN) | 979-8-3503-4319-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICMLCN59089.2024.10624754

Auto-Regressive RF Synchronization Using Deep-Learning

Michael Petry*,†, *Student Member, IEEE,* Benjamin Parlier*,‡, Andreas Koch*,†, Martin Werner†
*Airbus Defence and Space GmbH, †Technical University of Munich, ‡RWTH Aachen

***Abstract*—This work presents a novel pilot-less Deep-Learning-based synchronization mechanism that seamlessly integrates within state-of-the-art auto-encoder-based end-to-end communication systems. By re-using the idea of Radio Transformer Networks, an auto-regressive strategy is designed that learns to estimate and mitigate synchronization-related perturbations for arbitrarily modulated continuous communication, i.e., sample time offset (STO) and carrier frequency offset (CFO). A performance gain of 0.6 dB in the high-SNR regime compared to classic synchronization techniques is demonstrated. The strength of this approach is a shift from sample-by-sample to batch-wise processing according to the ML paradigm, which enables efficient and fast computation required for practical deployment scenarios using hardware-accelerated ML inference engines.**

***Index Terms*—RF synchronization, algorithm, auto-regressive, machine learning, sample time offset, center frequency offset, RF front end**

I. Introduction

”Let’s assume the system is synchronized” is a phrase one cannot avoid when educating oneself on signal processing algorithms within the domain of wireless communication [1]. Whether it’s the subject of channel equalization, error correcting codes, signal detection, or decryption, these and much more take a synchronized system for granted. This becomes clear when trying to operate even the simplest digital communication system on real hardware. Without synchronization nothing works in practice, while everything works flawlessly in theory. Unarguably, synchronization is an indispensable component of every digital communication system, and with the introduction of Artificial Intelligence (AI) techniques into next-generation communications systems, the issue of synchronization demands to be re-explored.

Seven years ago, the hype of Machine Learning (ML) sparked over to the RF domain [2] and resulted in a staggering echo within the community. This resonance has manifested itself in several ways, such as the creation of a dedicated Emerging Technology Initiative at IEEE ComSoc [3], the introduction of various AI/ML study items at 3GPP for 6G technologies [4], and in the creation of dedicated ML-tailored venues[1]. Related research is spanning across the RF domain, ranging from neural equalization [5] and graph NN-based channel decoding [6] in SISO systems to advances in cell-free Massive MIMO [7], capacity-orchestration and -scheduling techniques [8], data-aware re-transmission [9], and much more.

Although most research is simulation-based, recent over-the-air communication experiments were inevitably exposed to the issue of synchronization, which created awareness of the algorithmic gap of ML-based synchronization strategies.

In 2016, O’Shea *et al.* performed the first attempt of explicit ML-based synchronization on recorded IQ-samples by leveraging learned attention models and appropriate domain transformations, framed in the so-called ”Radio Transformer Network” (RTN) [10]. However, this strategy was not targeted for communication but for signal classification. In 2017, Dörner *et al.* trained an external frame-synchronization module based on sequence decoding to enable continuous over-the-air transmission [11]. However, their setup is based on block-wise auto-encoders, which is known to suffer of performance degradation compared to its bit-wise counterpart, introduced 3 years later [12]. Other attempts exploit either preambles [13] or carrier aggregation [12] in the OFDM modulation scheme. Lastly, special circumstances, such as burst-wise very short packet communication, allows successful reception despite intentionally neglecting synchronisation mechanisms [14].

Up to date, there is a lack of ML-based strategies that solve this issue *explicitly* and therefore a pillar that enables success in conventional systems is missing. With this work we want to give an initial impetus to strengthening this foundation.

The main contribution of this work is the proposition of a novel ML-based auto-regressive synchronization strategy that operates on the physical layer, is pilot-less, supports arbitrarily-shaped modulation schemes, and integrates natively into state-of-the-art AI-based communication systems.

This paper is structured as followed: We revisit the problem of synchronization in Sec. II by reviewing the sources of signal perturbation and their impact on signal integrity, and cover popular (classic) mitigation strategies. We present our novel AI-based synchronization mechanism as well as its training strategy in detail in Sec. III. A first evaluation on its performance, robustness, and competitiveness to classic counterparts is provided in Sec. IV. Sec. V concludes this work and hints possible algorithm improvements. This is followed by an outlook on remaining challenges for making ML-based synchronization practically useful.

This work is supported by the German Aerospace Center (DLR) under project Machine Learning on Telecommunications Satellites (MaLeTeSa) with Grant number 50 YB 2103. Corresponding author: Michael Petry (e-mail: michael.petry@airbus.com).

[1]For a non-exhaustive list please see: https://mlc.committees.comsoc.org/workshops-tutorials-symposia/.

2024 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN)

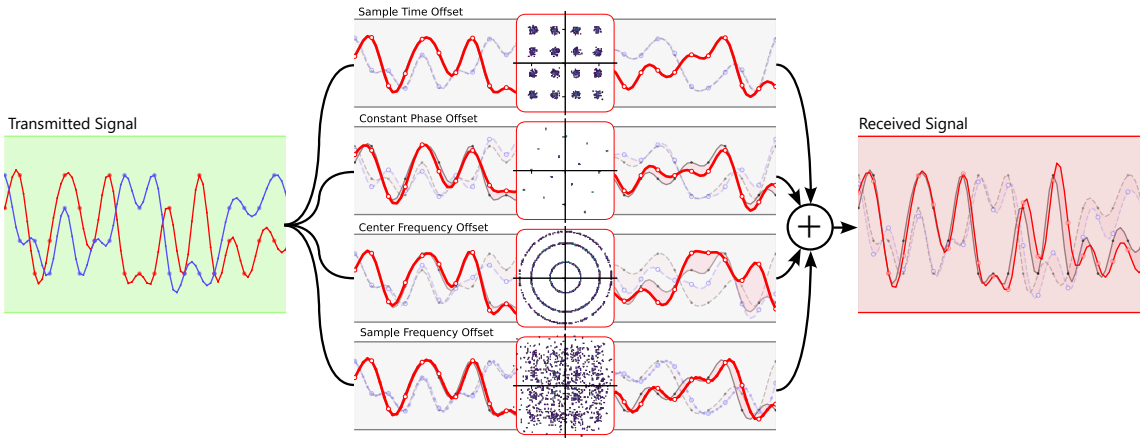


Fig. 1. Visualization of synchronization requirements in the baseband for a 16-QAM modulated signal. Left: Transmitted signal. Right: Received Signal. Red and blue colored lines denote the in-phase and quadrature components of the baseband signal, respectively. Center: Sample Time Offset, Constant Phase Offset, Center Frequency Offset, and Sample Frequency Offset individually illustrated on the full signal. The insets visualize individual impacts on the constellation diagram.

II. The Synchronization Problem - Source, Impact, and Classic Mitigation Strategies

Synchronization is of utmost importance for wireless communication as it is the key-enabler that allows communication between distant devices operating on practical, non-ideal hardware. Synchronization is necessary mainly for two reasons.

Firstly, the properties of a received radio wave depend on various environmental conditions, such as the propagation channel and the mobility of transmitter (TX) and/or receiver (RX) (Doppler effect). While the first effect can be mostly compensated using filtering strategies, the Doppler effect imposes a dynamic change in frequency and phase on the received signal, which ultimately leads to a mismatch between the analog RF chain and succeeding digital processing.

Secondly, due to tolerances in the manufacturing process of analog components (e.g., local oscillator (LO), mixer, Phase-locked loop (PLL), etc.), hardware imperfections are inevitably present in the RF front end. On top of this, time-variance of the components, such as the frequency dependency of the LO on environmental conditions like temperature, or stability issues like phase jitter, impact the transmitted and received signal fundamentally. Together, the propagation characteristics and hardware imperfections perturb the received IQ-samples in a multitude of ways, which can be summarized in four critical impacts that are explained in the following subsection.

A. Critical Signal Perturbations

Fig. 1 summarizes the four impacts. The left and right graphs show the transmitted (ideal) and the received (perturbed) RF signal, respectively, using 16-QAM modulation in baseband. The four centered graphs visualize each impact as they accumulate from start to end of the signal in an isolated fashion, with the insets denoting the changes observed in the constellation diagram. The red shaded region denotes the difference between the ideal and the perturbed signal. The effects and their origins are briefly explained in the following, ordered from simple to hard w.r.t. mitigation strategy complexity.

a) Sample Time Offset (STO): Time offset of the RX's Analog-to-Digital converter's (ADC) sampling times w.r.t. the matched-mfiltered "pulse-centered" IQ-samples. Static STO is caused by the ADC's random initial sample time and dynamic STO is caused by a sample rate offset (see d) below).

b) Constant Phase Offset (CPO): Quasi-static phase rotation within the observed IQ-samples. CPO is caused both by the phase rotation imprinted on the signal due to propagation distance and time, as well as by the LO's random phase in both TX and RX front end.

c) Center Frequency Offset (CFO): Frequency difference between the center frequencies of the TX's and RX's RF front end. CFO is primarily caused by precision tolerances and stability issues within LOs, however, mobility scenarios invoke the Doppler effect and directly impact signal frequency.

d) Sample Frequency Offset (SFO): Fractional mismatch between the receiver ADC's sample frequency and the instantaneous rate of received symbols. SFO has a similar origin as CFO.

As shown in the constellation diagrams in Fig. 1, these impacts either deteriorate signal quality severely or even make classic signal demodulation impossible, necessitating mitigation strategies as outline in the following.

B. Classic Synchronization Strategies

Synchronization strategies have to be individually tailored to the specific properties of the communication system at hand, with primary factors being the modulation type, transmit or

receive diversity (i.e., SISO or MIMO), interference considerations (e.g., single/multi-user systems), and the RF front end architecture. This has led to a vast proposal of DSP algorithms, from which we want to recall the most popular and fundamental software-only algorithms (e.g., no software-controlled in-hardware Phase-locked loops) to facilitate understanding of the rest of this paper.

From a high-level perspective, all synchronization strategies perform the following sub-tasks in the following or similar order.

1) Coarse carrier frequency recovery - Correct (mean) CFO on a large timescale to a few kHz precision to enable accurate IQ-sample processing on a small timescale: Popular algorithms are the Frequency Locked Loop using band-edge filters, which converts the residual energy in the modulation's excess bandwidth to a control signal, or the "Multiply-filter-divide" method, which recovers the carrier frequency using a non-linear frequency multiplier operation.
2) Symbol timing recovery - Remove time offset in sampled IQ data to achieve optimal sampling positions: Algorithms typically resemble a "software"-PLL, such as Mueller and Muller's timing recovery scheme [15], early-late or Gardner timing error detectors, and M-path Polyphase filter banks [1]. DSP-based algorithms might use interpolation to resolve inter-sample IQ-values.
3) Fine carrier frequency and phase recovery - Remove dynamic small-scale CFO and unknown phase offset to resolve ambiguity for demodulation: Closed-loop algorithms like the Costas-Loop [16] utilize a feedback-based PLL to eradicate unwanted phase offsets on a sample-by-sample basis.

The effectiveness, computational efficiency, robustness to dynamic behavior, and acquisition time (if applicable) of these procedures highly depend on the selected DSP algorithms, whose analyses are out of scope for this paper. However, various properties allow for further optimisation, e.g., increasing spectral efficiency by removing pilot symbols or decreasing acquisition time and enhancing robustness to abruptly occurring propagation effects. To address these issues we propose a novel ML-based synchronization procedure in the next section.

III. DL-based Auto-Regressive Synchronization Strategy

In this section we propose a novel pilot-less auto-regressive ML-based synchronization strategy that mitigates STO, CFO, and CPO. For simplicity we assume uni-directional continuous communication in a SISO scenario, although generalization to full-duplex MIMO systems is discussed in Sec. V.

The strategy exhibits three distinct characteristics:

- Batch-wise processing: To facilitate computationally efficient operation, IQ data is processed in batches to conform with the ML paradigm. This is in contrast to most classic algorithms as shown in Sub-sec. II-B.

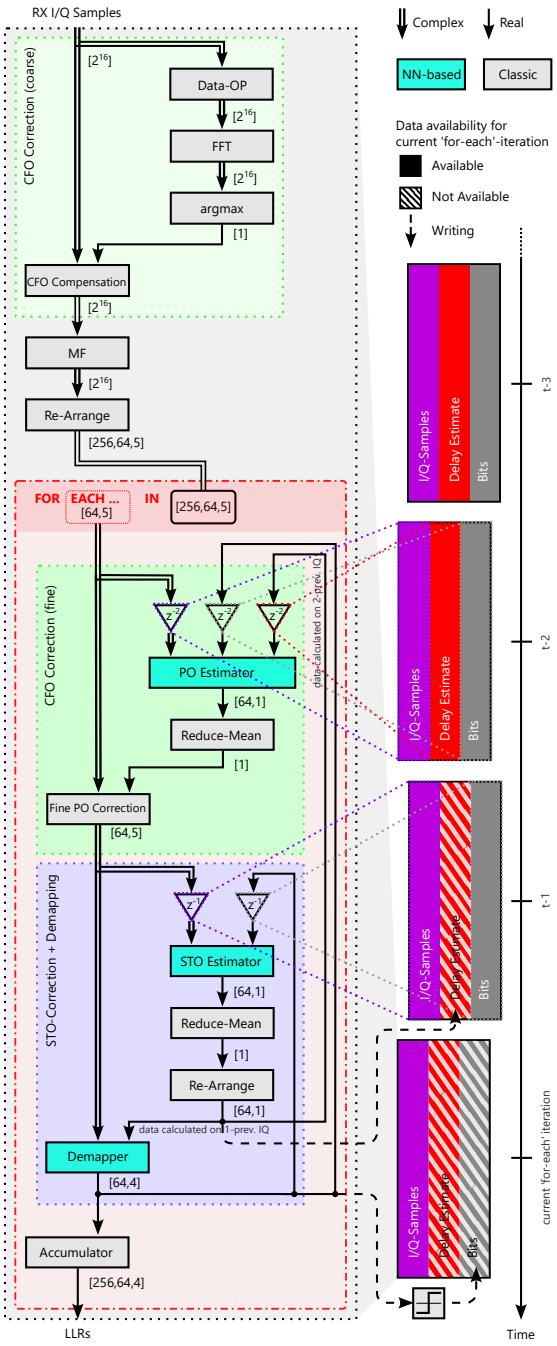


Fig. 2. Visualisation of the data flow (left) and intermediate data availability (right) in the receiver with ML-based synchronization.

- Pilot-less: The transmission of any form of known symbols (pilots) is not required, hence, spectral efficiency remains untouched.
- Auto-regressive information re-use: Information extracted from previously processed data is integrated into future processing steps.

In the following subsection we elaborate its core functionality, design choices, and data flow.

A. Algorithm Functionality and Data Flow

In a nutshell, the algorithm solves the sub-tasks listed in Sub-sec. II-B in a different order using the idea of RTNs to estimate signal properties with Neural Networks (NNs), and integrates domain knowledge by performing signal transformations with mathematical "layers". Fig. 2 visualizes the individual processing steps in a graph-like manner for transforming received IQ data to information bits. Data flows from top to bottom with the grey shaded region containing the processing pipeline for handling one batch of IQ data. The right-aligned timeline details the availability of inferred data from prior IQ mini-batches (explained below) w.r.t. time. Groups of operations that perform a distinct function, e.g., STO compensation, are highlighted by different background colors, while individual operations are either colored turquoise for ML-based operations or grey for mathematical operation. Fig. 2 assumes an IQ-batch size of 2^{16} complex samples, a mini-batch size of 256, and a receiver up-sampling factor of $\mu_{\mathrm{RX}} = 4$. In the following we outline the implemented data flow.

The algorithm utilizes a three-step strategy to process an unsynchronized signal. In the first step (see light green shaded region), a coarse CFO compensation is performed by estimating the CFO using Fast-Fourier-Transform (FFT) on modulation-removed input data (implemented by raising the signal to the power of m), followed by a compensation via a mathematical transformation. Although FFT is used for simplicity here, convolutional neural network (CNN)-based CFO estimation is equally applicable with similar precision. Although not utilizing ML in this implementation, this step can be interpreted as an RTN with a classic feature extractor.

Before the next synchronization step, the coarsely CFO-corrected data is matched-filtered. Filtering after CFO estimation has proven to result in a higher precision for coarse CFO estimation. This is attributed to the fact that filtering of frequency-shifted data would result in an asymmetric spectrum w.r.t. the center frequency.

From this point on ML comes into play. The IQ-batch is split into equisized mini-batches which are being processed sequentially (denoted by the red shaded "for each" loop in Fig. 2). First, the phase offset in the mini-batch is estimated via means of ML and corrected using mathematical transforms. Afterwards, STO is estimated and supplied to the Demapper, both operating using NNs. The Demapper outputs Log-Likelihood-Ratios (LLRs) to allow further processing (e.g., decoding) with soft information.

B. Crucial Points of our Strategy

From an information theoretic point of view, estimation of the phase offset solely based on IQ-data is not possible due to the lack of knowledge on what happened to the signal beforehand, which is classically derived from pilot symbols. STO estimation faces a similar issue when the sampling times are centered between two symbols, leading to an ambiguity in whether to select the left or right symbol. To solve this issue we apply two creative tricks:

a) Trick 1: Both Dense Neural Network (DNN)-based estimators necessitate some kind of reference data. Providing the originally transmitted unperturbed IQ-samples as reference data enables both estimators to successfully learn producing the desired quantities; however, this data is obviously not available at the receiver. A qualified substitute is required, for which we identified the corresponding bits as an excellent alternative. By exploiting deep neural networks, feature extraction of any kind can be learned by pairing the received IQ-data with the corresponding demodulated bits. Working with these unknown bits leads to trick 2:

b) Trick 2: Although the corresponding bits are unknown for the current processed mini-batch (otherwise the task of communication would become redundant), they are acquired for previous mini-batches (assuming a running system), hence, both estimators operate on previous data, making this algorithm auto-regressive[2]. This necessitates similarity between adjacent mini-batches, which leads to trick 2: The crucial point of this strategy is to split the IQ-batch into equisized mini-batches with lengths small enough that allow assuming a similar phase offset and STO for adjacent mini-batches.

In simpler terms, we estimate CPO and STO on previously processed IQ-data combined with demodulated bits, and apply corrective transforms on the current data under the assumption of quasi-static signal characteristics.

CPO and STO are estimated symbol-wise, followed by a combine-operation (here: mean) to enhance estimation stability. Demapping is also performed symbol-wise using the "window" of corresponding IQ-samples ($\mathrm{ups}_{\mathrm{RX}} + 1$ to handle STO) together with the estimated STO. The final data is accumulated.

The following sub-section elaborates the training strategy for this algorithm.

C. Genie-Aided Training Strategy

Although this strategy gives rise to various hyper-parameters and architectural data handling options that require tuning, which includes but is not limited to IQ-batch size, mini-batch size, estimator output handling, characteristics of RF components, propagation properties, and mobility scenarios, we restrict our focus on the training strategy of the NNs within the algorithm for one selected operation environment only for complexity reasons.

To simplify and speed-up training we utilize a genie-aided training strategy that resolves looping over mini-batches and

[2]The phase estimator is additionally supplied with the STO estimated in the previous batch, which boosts its accuracy but leads to a $2\times$ delay.

Algorithm 1 Genie-Aided NN Training

Input: Number of Epochs E
Batch size $B := 1$
Num. of IQ-Symbols per Batch N_I
Num. of IQ-Symbols per Mini-Batch N_O
Max-Frequency Offset $f_{\mathrm{CFO,max}}$
Modulation-Order m
Training SNR SNR
Output: Trained NN

repeat for E Epochs
  `Initialization:`
  $\mathbf{B} \leftarrow \mathrm{randBinary}([B, N_\mathrm{I}, N_\mathrm{O}, m])$
  $\vec{\tau}_\mathrm{sto} \leftarrow \mathrm{randUniform}(-0.5, 0.5, [N_\mathrm{I}]) \cdot T_\mathrm{samp}$
  $\phi_\mathrm{po} \leftarrow \mathrm{randUniform}(0, 2\pi)$
  $f_\mathrm{cfo} \leftarrow \mathrm{randUniform}(-f_{\mathrm{CFO,max}}, f_{\mathrm{CFO,max}})$
  $\mathbf{S} \leftarrow$ **NN-Modulator(B)** $\in \mathbb{C}^{B\times N_\mathrm{I}\times N_\mathrm{O}\times 1}$
  $\vec{U} \leftarrow$ Flatten and Upsampling by μ_TX of $\mathbf{S}$
  `De-Synchronizing TX signal:`
  for $i \leftarrow 0$ to N_I **do**
    $\vec{IQ}^{\mathrm{sto}}_{(i:(i+1))\cdot N_\mathrm{O}} \leftarrow \vec{U}_{(i:(i+1))\cdot N_\mathrm{O}} \circledast g_\mathrm{rcc}(t - \vec{\tau}_{\mathrm{sto,i}})$
  end for
  `Apply AWGN to set SNR:`
  $\vec{IQ}^{\mathrm{Noise}} \leftarrow \vec{IQ}^{\mathrm{sto}} + \mathcal{CN}(0, \sigma^2)$
  $\vec{IQ}_\mathrm{RX} \leftarrow \vec{IQ}^{\mathrm{Noise}} \times e^{j\cdot 2\pi \cdot f_\mathrm{cfo}\cdot \vec{t} + \phi_\mathrm{po}}$
  `Receiver Procedure:`
  $\hat{f}^{\mathrm{coarse}}_\mathrm{cfo} \leftarrow \underset{f}{\mathrm{argmax}}(\mathtt{FFT}(\vec{IQ}^{m}_\mathrm{RX}))/m$
  $\vec{IQ}^{\mathrm{a}} \leftarrow \vec{IQ}_\mathrm{RX} \times e^{-j\cdot 2\pi \cdot \hat{f}^{\mathrm{coarse}}_\mathrm{cfo}}$
  $\vec{IQ}^{\mathrm{a}} \leftarrow \vec{IQ}^{\mathrm{a}} \circledast g_\mathrm{rcc}(t)$
  $\hat{\boldsymbol{\phi}} \leftarrow$ **EstimatePhaseOffsets**$(\vec{IQ}^{\mathrm{a}}, \mathbf{B}, \vec{\tau}_\mathrm{sto})$
  for $i \leftarrow 0$ to N_I **do**
    $\vec{IQ}^{\mathrm{b}}_{(i:(i+1))\cdot N_\mathrm{O}} \leftarrow \vec{IQ}^{\mathrm{a}}_{(i:(i+1))\cdot N_\mathrm{O}} \times e^{-j\cdot \mathtt{angle}(\hat{\phi}_\mathrm{i})}$
  end for
  $\hat{\boldsymbol{\tau}} \leftarrow$ **EstimateSampleTimeOffsets**$(\vec{IQ}^{\mathrm{b}}, \mathbf{B})$
  $\mathbf{IQ}^{\mathrm{b}} \leftarrow \mathtt{Re\text{-}Arrange}\ \vec{IQ}^{\mathrm{b}}\ \mathtt{to}\ [B, N_\mathrm{I}, N_\mathrm{O}, m+1]$
  $\hat{\mathbf{B}} \leftarrow$ **NN-Demodulator**$(\mathbf{IQ}^{b}, \hat{\boldsymbol{\tau}})$
  $l_\mathrm{bce} \leftarrow \mathtt{BinaryCrossEntropy\text{-}Loss}(\mathbf{B}, \hat{\mathbf{B}})$
  $l_\mathrm{custom} \leftarrow l_\mathrm{bce} + 5 \cdot \mathtt{MSE}(\vec{\tau}_\mathrm{sto}, \hat{\boldsymbol{\tau}})$
  `Update all` **NNs** `using` l_custom
until

handling of previous data, while maintaining the algorithm's auto-regressive character. The training strategy is detailed in Algorithm 1. Algorithms 2 and 3 denote the CPO and STO estimation steps, respectively.

In summary, the algorithm is trained end-to-end within an auto-encoder-based setup. Signal perturbations are introduced on the transmitted IQ samples, i.e., CFO with random intensity per epoch sampled between $\pm f_{\mathrm{CFO,max}} = 50$kHz, CPO with random initial phase per epoch, and STO with random offset

Algorithm 2 EstimatePhaseOffsets

Input: IQ-Samples $\vec{IQ} \in \mathbb{C}^{B\cdot N_\mathrm{I}\cdot N_\mathrm{O}\cdot \mu_\mathrm{TX}}$
Bit-Tensor $\mathbf{B} \in \{0,1\}^{B\times N_\mathrm{I}\times N_\mathrm{O}\times m}$
Sample-Time-Offsets $\vec{\tau}_\mathrm{sto} \in \mathbb{R}^{B\times N_\mathrm{I}}$
Output: Estimated Phase Offsets $\hat{\phi} \in \mathbf{R}^{B\times N_\mathrm{I}}$

$\mathbf{IQ} \leftarrow \mathtt{Re\text{-}Arrange}\ \vec{IQ}\ \mathtt{to}\ [B, N_\mathrm{I}, N_\mathrm{O}, \mu_\mathrm{TX}+1]$
$\boldsymbol{\tau} \leftarrow \mathtt{Repeat}\ \vec{\tau}_\mathrm{sto}\ \mathtt{to}\ [B, N_\mathrm{I}, N_\mathrm{O}, 1]$
$\tilde{\boldsymbol{\phi}} \leftarrow \mathrm{NN}(\mathtt{stack}\ \mathbf{IQ}, \mathbf{B}, \boldsymbol{\tau}) \in \mathbb{C}^{B\times N_\mathrm{I}\times N_\mathrm{O}\times 1}$
$\tilde{\boldsymbol{\phi}}^0 \leftarrow \tilde{\boldsymbol{\phi}}/|\tilde{\boldsymbol{\phi}}|$
return `Mean on last 2 dim of` $\tilde{\boldsymbol{\phi}}$

Algorithm 3 EstimateSampleTimeOffsets

Input: IQ-Samples $\vec{IQ} \in \mathbb{R}^{B\cdot N_\mathrm{I}\cdot N_\mathrm{O}\cdot \mu_\mathrm{TX}}$
Bit-Tensor $\mathbf{B} \in \{0,1\}^{B\times N_\mathrm{I}\times N_\mathrm{O}\times m}$
Output: Estimated Sample Time Offsets $\hat{\tau} \in \mathbf{R}^{B\times N_\mathrm{I}}$

$\mathbf{IQ} \leftarrow \mathtt{Re\text{-}Arrange}\ \vec{IQ}\ \mathtt{to}\ [B, N_\mathrm{I}, N_\mathrm{O}, m+1]$
$\tilde{\boldsymbol{\tau}} \leftarrow \mathrm{NN}(\mathtt{stack}\ \mathbf{IQ}, \mathbf{B})$
return `Mean on last 2 dim of` $\tilde{\boldsymbol{\tau}}$

per epoch. Phase coherency is guaranteed between epochs. Additionally, additive white Gaussian noise with constant $E_b/N_0 = 10$ dB is applied. The total training time on an NVIDIA Tesla V100 (AWS Cloud instance) for 100k epochs is approx. 5 hours. The general training mechanism is similar as in [17].

IV. Evaluation

To verify functionality and performance of the proposed algorithm we perform an extensive study which is summarized in this section.

Fig. 3 provides BER curves of the developed system (red), a classically synchronized pendant using techniques listed in Sub-sec. II-B (yellow), and the theoretic baseline for a

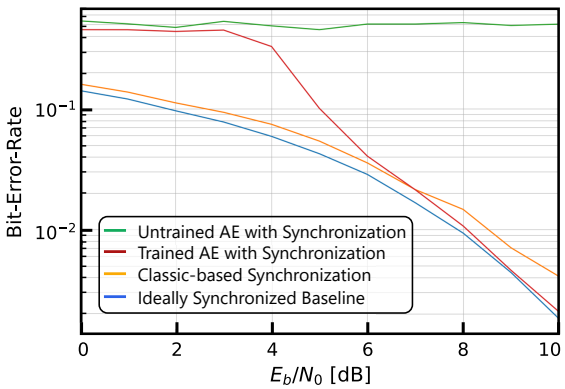


Fig. 3. Bit-Error-Rate curves for the untrained AE-based system (green), trained AE setup with AI-based synchronization (red), communication using classic synchronization techniques (yellow), and theoretic maximum performance for an ideally synchronized system (blue). A dynamic CFO with $f_{\mathrm{CFO,max}} = 50$kHz and varying STO is applied.

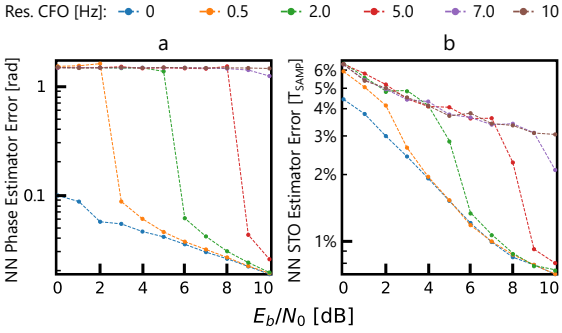


Fig. 4. Visualisation of (a) absolute NN-based phase estimation error and (b) relative NN-based STO estimation error w.r.t. the RX ADC's sampling period as a function of residual CFO and SNR. Additionally a dynamically varying STO is applied on the transmitted signal.

perfectly synchronized system without overhead (blue) as a function of E_b/N_0. It assumes 16-QAM-based transmission without coding in a realistic environment (STO and CFO vary dynamically during transmission, similarly as implemented for NN-training). It can be seen that the classic approach exhibits a constant performance gap of approx. 0.7 dB, while the AI-based synchronized setup reaches a negligible gap in the high SNR region. Nevertheless we note, that communication in the low SNR region is error-prone due to bit error propagation in the auto-regressive algorithm structure, as explained in the following.

To understand the above observed behavior and provide a more detailed analysis on the performance of the synchronization algorithm and its individual components, we study NN-based STO and CPO estimation precision as a function of CFO intensity and SNR while applying a dynamically varying STO. To simplify understanding we excluded the FFT-based coarse CFO compensation from this analysis and adapted the applied CFO range to match the residual CFO after this step.

Fig. 4(a) shows the phase estimator error, and Fig. 4(b) shows the STO estimator error. It is evident that both estimators strongly depend on both residual CFO and SNR, and that higher CFOs necessitate higher SNRs to uphold precision, which is intuitively expected. We could observe that when the SNR is low, too high residual CFOs lead to mis-estimations that fuel a chain-reaction of bit error propagation, prohibiting further communication.

V. Conclusion and Future Work

In this work we proposed, implemented, and evaluated a novel pilot-less auto-regressive RF synchronization strategy using Deep-Learning. By extending a state-of-the-art bit-wise auto-encoder-based setup we demonstrated continuous unidirectional communication by simulating a realistic environment, perturbed by hardware imperfections in the RF front end and propagation perturbations, leading to STO and CFO. Although low-SNR communication is highly error prone due to auto-regressive bit error propagation, we achieved a gain of approx. 0.6 dB in the high-SNR region.

Future work might consider improving this algorithm, e.g., by minimizing error propagation using soft information (LLRs instead of bits) within the NN-based parameter estimators, and extending this strategy to MIMO communication by deploying the algorithm on every RF path by utilizing a batch-size > 1. Although we believe that this work can serve as an introduction to ML-based synchronization, more questions have been raised than answered. To reach maturity some issues remain, such as developing a start-up procedure when no prior data is present, and extending this work by adding Sample Frequency Offset compensation.

Acknowledgment

We would like to thank Marc Lichtman for his work on PySDR and the resulting fruitful discussion.

References

[1] f. harris, *Let's Assume the System Is Synchronized*, 01 2011, pp. 311–325.
[2] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563–575, 2017.
[3] I. C. Society, "Machine learning for communications emerging technologies initiative," https://mlc.committees.comsoc.org/.
[4] X. Lin, "Artificial intelligence in 3gpp 5g-advanced: A survey," 2023.
[5] W. Xu, Z. Zhong, Y. Be'ery, X. You, and C. Zhang, "Joint neural network equalizer and decoder," in *2018 15th International Symposium on Wireless Communication Systems (ISWCS)*, 2018, pp. 1–5.
[6] S. Cammerer, J. Hoydis, F. Aoudia, and A. Keller, "Graph neural networks for channel decoding," 12 2022, pp. 486–491.
[7] T. T. Vu, D. T. Ngo, N. H. Tran, H. Q. Ngo, M. N. Dao, and R. H. Middleton, "Cell-free massive mimo for wireless federated learning," *IEEE Transactions on Wireless Communications*, vol. 19, no. 10, pp. 6377–6392, 2020.
[8] H. Xing, O. Simeone, and S. Bi, "Decentralized federated learning via sgd over wireless d2d networks," in *2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2020, pp. 1–5.
[9] D. Liu, G. Zhu, J. Zhang, and K. Huang, "Wireless data acquisition for edge learning: Importance-aware retransmission," in *2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2019, pp. 1–5.
[10] T. J. O'Shea, L. Pemula, D. Batra, and T. C. Clancy, "Radio transformer networks: Attention models for learning to synchronize in wireless systems," in *2016 50th Asilomar Conference on Signals, Systems and Computers*, 2016, pp. 662–666.
[11] S. Dörner, S. Cammerer, J. Hoydis, and S. t. Brink, "Deep learning based communication over the air," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 132–143, 2018.
[12] S. Cammerer, F. A. Aoudia, S. Dörner, M. Stark, J. Hoydis, and S. ten Brink, "Trainable communication systems: Concepts and prototype," *IEEE Transactions on Communications*, vol. 68, no. 9, 2020.
[13] J. Liu, K. Mei, X. Zhang, D. McLernon, D. Ma, J. Wei, and S. A. R. Zaidi, "Fine timing and frequency synchronization for mimo-ofdm: An extreme learning approach," *IEEE Transactions on Cognitive Communications and Networking*, vol. 8, no. 2, pp. 720–732, 2022.
[14] J. Clausius, S. Dörner, S. Cammerer, and S. T. Brink, "On end-to-end learning of joint detection and decoding for short-packet communications," in *2022 IEEE Globecom Workshops (GC Wkshps)*, 2022, pp. 377–382.
[15] K. H. Mueller and M. Muller, "Timing recovery in digital synchronous data receivers," *IEEE Trans. Commun.*, vol. 24, pp. 516–531, 1976.
[16] J. P. Costas, "Synchronous communications," *Proceedings of the IRE*, vol. 44, no. 12, pp. 1713–1718, 1956.
[17] M. Petry, A. Koch, and M. Werner, "Envisioning physical layer flexibility through the power of machine-learning," in *2023 IEEE Globecom Workshops (GC Wkshps)*, "in press".

4.2.1 Discussion on AI-Augmented Physical Layers

[Pub4.B] proposes an AI-augmented communication system architecture (ref. to level 2 of the categorization established in Sub-sec. 4.1.1) to specifically optimize RF signal synchronization in a single-carrier single-input single-output system. As denoted in Fig. 2 in [Pub4.B, P. 3], roughly a quarter of operations are realized using means of DL (blue), while the rest builds on classical algorithms (grey).

This approach resolves a majority of the limitations noted in Sub-sec. 4.1.1. First of all, it successfully integrates domain knowledge algorithms, i.e., FFT, FIR-filters, and various elementary operations, such as phase offset correction using complex multiplications and modulation removal via complex exponentiation, into the processing chain, circumventing NNs that emulate known behavior. Simultaneously, more throughput performance potential is opened up by reducing the overall ML workload. Despite a seeming disadvantage with respect to E2E optimization of the complete chain by introducing static, non-learnable classical components, the learnable components' parameters are still optimized in an E2E fashion by building on a fully differentiable implementation and utilizing AutoDiff as described in Sub-sec. 2.2.4. Finally, these changes deliver a BER performance that is not only on par with the classical baseline, but exceeds it and approaches the theoretical limit in the operating regime of interest.

Despite these optimizations, there is also a caveat which requires further investigation: While the AI-native architecture implies small deployment complexity due to only a single sequential NN to execute on both TX and RX, the AI-augmented architecture requires the tight integration of classical and ML components with respect to data flow and buffering of intermediate values, sequencing and timing, and distribution of the workload on a block-level to different processing domains to be efficiently executable. While on standard terrestrial hardware like GPUs a variety of frameworks and drivers exist to realize this procedure fairly easily, available processing domains in space, i.e., CPU, FPGA, DSP, and ASIC, render this a non-trivial and tedious venture. Depending on the complexity of the processing chain and the target hardware platform, a manually tailored implementation might be required.

In the following section, [Pub4.C] investigates and sketches such a manual implementation for realizing the hybrid processing chain for DL-augmented RF signal synchronization, as proposed in [Pub4.B], on the VERSAL hardware platform by considering and trading-off the above discussed factors.

4.3 Pub4.C: Zero-Copy AI-augmented Signal Processing Pipeline on Heterogeneous Satellite Processors

Authors: Michael Petry, Andreas Koch, Martin Werner

Publication Medium: Asilomar Conference on Signals, Systems & Computers, 2024

Digital Object Identifier: https://doi.org/10.1109/IEEECONF60004.2024.10943093

Thesis contextualization: Building on the hybrid architecture proposed in [Pub4.B], this paper follows the conclusion in 4.2.1 and investigates the efficient implementation of its receiver-side architecture on the VERSAL. We can achieve this goal by employing a manual deployment strategy that allows full control over the complete inference process, i.e., distributing workload, implementing an optimized zero-copy data buffering strategy, and managing data flow and access schemes. While it raises open points, most importantly regarding generalization to other algorithms and architectures, we ultimately confirm the feasibility of realizing and combining hybrid (AI/ML and classical) processing workloads on future spacecraft payloads, addressing [RQ-1] - [RQ-5].

CRediT author statement:

Michael Petry:	75 %	Investigation, Conceptualization, Methodology, Software, Data curation, Formal analysis, Validation, Visualization, Writing
Andreas Koch:	15 %	Investigation, Software, Writing
Martin Werner:	10 %	Conceptualization, Writing

Zero-Copy AI-augmented Signal Processing Pipeline on Heterogeneous Satellite Processors

Michael Petry*,† , *Student Member, IEEE,* Andreas Koch*,† , *Student Member, IEEE,* Martin Werner†
*Airbus Defence and Space GmbH, †Technical University of Munich

***Abstract*—We present a novel signal processing concept that natively integrates artificial intelligence (AI) processing capabilities into satellite platforms. The processing workflow on high-end, resource-constrained platforms is typically Field Programmable Gate Array (FPGA)-dominated which renders the efficient integration of Machine Learning-based components a challenge. We address this by leveraging a novel System-on-Chip (SoC)-based heterogeneous compute platform and propose a deployment concept that allows for a flexible and dynamic arrangement of conventional and AI-based processing steps. After a comprehensive review of the hardware components we identify efficient inter-component data streaming as the primary implementation challenge, which we address using a zero-copy strategy. The complete deployment process is outlined for an AI-augmented telecommunications processing scenario. A succinct performance analysis of the critical inter-component data flow concludes this work, identifying potential bottlenecks and discussing avenues for future enhancements.**

***Index Terms*—Satellite processing, artificial intelligence, heterogeneous computing, zero-copy, wireless communication**

I. Introduction

Artificial intelligence (AI) has become ubiquitous in terrestrial applications, offering innovative solutions for rapid and efficient problem solving across various domains, including image and speech recognition, behavioral observations, autonomous reactions, and many more. While AI has seen widespread adoption on earth, its potential for space-bound systems, particularly satellites, has just recently begun to receive significant attention [1].

Possible benefits of on-board AI affect the whole ecosystem within and ”outside” of satellites. This includes removing the necessity of human-based intervention in spacecraft control by utilizing self-supervision and -recovery mechanisms through AI-powered behavior analysis [2], alleviating bandwidth-intensive raw data transfers for computer vision-based earth observation (EO) applications by processing data directly on-board [3], and increasing overall system performance by exploiting the strengths of Machine Learning (ML) in digital signal processing (DSP)-intensive tasks, such as telecommunications [4]. Ultimately, AI can completely transform how satellites interact with terrestrial systems, transforming them from mere data collectors with relay functionality to autonomous datacenters in space.

This work is supported by the German Aerospace Center (DLR) under project Machine Learning on Telecommunications Satellites (MaLeTeSa) with Grant number 50 YB 2103. Corresponding author: Michael Petry (e-mail: michael.petry@airbus.com).

Key enabler is the integration of generic AI-centric processing capabilities into the satellite platform and ensuring a native integration into the conventional processing flow. While terrestrial systems benefit from a wide landscape of specialized hardware, such as Graphics Processing Units (GPU), Tensor Processing Units (TPU), Digital Signal Processors, etc., which can efficiently work in concert due to standardized communication protocols and drivers, space-bound solutions lack this diversity. Amongst multiple restrictions, such as the limitation in available hardware technology due to harmful radiation, a tight power budget due to solar energy sources as well as thermal management, or weak general computational power due to heritage-proofed processors, the major challenge is the extreme reliance on Field Programmable Gate Array (FPGA)-devices, especially for high-performance satellites, which renders the seamless integration of AI into the processing chain cumbersome.

This challenge has initiated a race amongst both established companies as well as newly formed start-ups. Amongst many proposed solutions, a straightforward approach is to utilize standalone modules designed for AI computation, such as NVIDIA’s embedded GPU module series [5] and Intel’s Movidius™ Myriad™ X Vision Processing Unit, or neuromorphic architectures, such as Brainchip’s akida IP [6] and Intel’s Loihi chip [7], accepting the penalty of missing radiation compliance and its associated problems of runtime faults and a shortened device lifespan. A different approach is the concept of General Purpose Computing on Graphics Processing Units (GPGPU) using FPGA-accelerated soft-GPU Intellectual Property (IP)-cores, effectively realizing a GPU on the FPGA [8]. Furthermore, purely software-based solutions that target execution on Central Processing Units (CPU) and microcontrollers are in active development, such as Klepsydra AI’s high efficiency inference framework [9].

However, the prevailing approach, among most solutions, is to treat the AI application as a separate entity by either managing it as an isolated software application or separating it into a distinct hardware component. The lack of native integration and the resulting suboptimal interfaces impede the effectiveness for high-performance applications.

This paper takes a different approach by integrating, and more importantly, interconnecting AI capabilities within the heart of the satellite. Its main contributions are the following: Based on a detailed component-level analysis of our next-generation satellite processor, we derive a deployment strategy

that prioritizes efficient data flow and effective hardware utilization based on a future reference scenario from the telecommunications domain, proposed by Petry *et al* [10]. To highlight relationships between both works, a coherent color-scheme is used throughout this work.

The rest of this paper is structured as follows: Sec. II introduces the telecommunications satellite processor design and reviews the reference scenario to be implemented. Sec. III proposes a deployment concept after analyzing the relevant hardware resources and their interconnectivity capabilities and describes the mapping procedure from algorithm to hardware. A brief performance analysis of the data flow is performed in Sec. IV, followed by an analysis of the system's behaviour and a discussion of possible improvements. Lastly, Sec. V concludes this work. This is followed by an outlook on the remaining challenges towards achieving powerful AI integration.

II. AI-augmentation of the Telecom Processor

In this section we describe the processor design and its main components towards AI integration, and introduce the reference scenario which is later implemented on this processor.

A. Overview Telecom Processor Architecture

The next-generation telecommunications processor architecture, developed as part of the European Space Agency (ESA)-funded Theia project, consists of multiple functional components on dedicated hardware boards, also known as slices, interconnected by space-grade communication links such as Space-Wire and optical links. Fig. 1 provides an overview of these slices. Among a control unit that supervises the spacecraft's health, and a stand-alone FPGA that enables in-flight re-programmable classic signal processing capabilities, the key slices w.r.t. telecommunications-related applications are the RF frontend, the signal regeneration (Re-Gen), and the AI-& DSP-Accelerator slices. While the RF slice typically functions as simple interface to the radio spectrum (although comprising certain integrated static processing features), the Re-Gen slice offers application-specific integrated circuit (ASIC)-based processing capabilities, such as protocol-specific (de-)coding and (de-)modulation. The main innovation is the first-of-its-kind AI- & DSP-Accelerator slice built around AMD-Xilinx's Versal series [11], as explained in subsection III-B. This slice is designed to serve as the main workhorse for tightly integrated AI-augmented processing scenarios.

B. Reference Scenario: AI-augmented Communications System with Deep Learning-based Signal Synchronization

Machine Learning (ML) has expanded into various aspects of wireless communication, including Integrated Sensing and Communications (ISAC) [12], differentiable Ray-Tracing-based digital-twin networks [13], [14], physical layer transceiver design [15]–[19], and more. The telecommunication satellite industry has recognized the potential of ML in its DSP-heavy applications, including beamforming and beam management, network orchestration and inter-satellite routing,

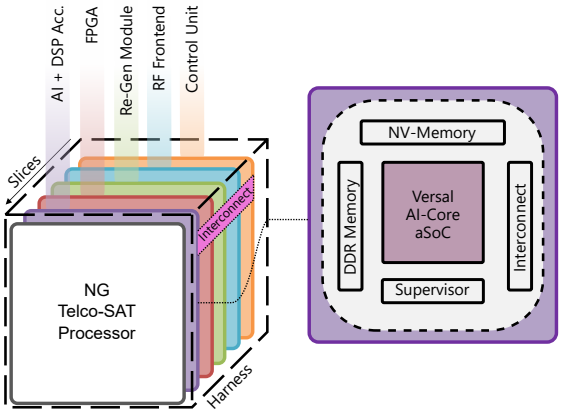


Fig. 1. Overview of the telecommunications processor design and its components. Detailed look into the AI- & DSP-Accelerator slice.

and radio access network (RAN) processing. Additionally, it has acknowledged the necessity of ML in future communication networks, such as 5G Non-Terrestrial-Network (NTN), as indicated by most 3GPP study items [20], [21], on satellite functionality.

To illustrate the presented deployment concept we select a reference scenario from the telecommunications domain that represents the key concept of future AI-augmented processing chains well: Combining conventional (classical) and AI-based processing blocks in a hybrid processing strategy. We realize a novel AI-augmented RF transceiver design from the receiver's perspective. The receiver integrates an auto-regressive signal synchronization procedure that estimates and compensates for coarse and fine center frequency offset (CFO) and sample time offset (STO) solely from the received IQ-samples. The "estimate-and-transform" strategy re-uses the concept of a Radio Transformer Network [22], utilizing mostly ML-based blocks to estimate certain signal parameters, followed by classical blocks to perform compensating signal transformations based on those estimations.

Fig. 2 visualizes the processing pipelines of the transmitter and receiver in analogous colors as originally presented in [10]. Small solid-border boxes denote the individual processing steps (turquoise: ML-based, gray: classic, red: RTN) and their surrounding shaded dotted-border boxes indicate their respective functionality. By focusing only on the receiver algorithm, it is evident that a complex computational graph which interleaves ML- and classically-based operations is utilized. Without delving too deeply into the functionality of the individual components, two RTNs are employed. The lower one utilizes a CNN to estimate synchronization-related properties, such as the phase offset from the RF signal, followed by the corresponding signal transformation using classic operations.

The utilization of a mixture of classic and ML-based processing is essential for high-performance and high-throughput

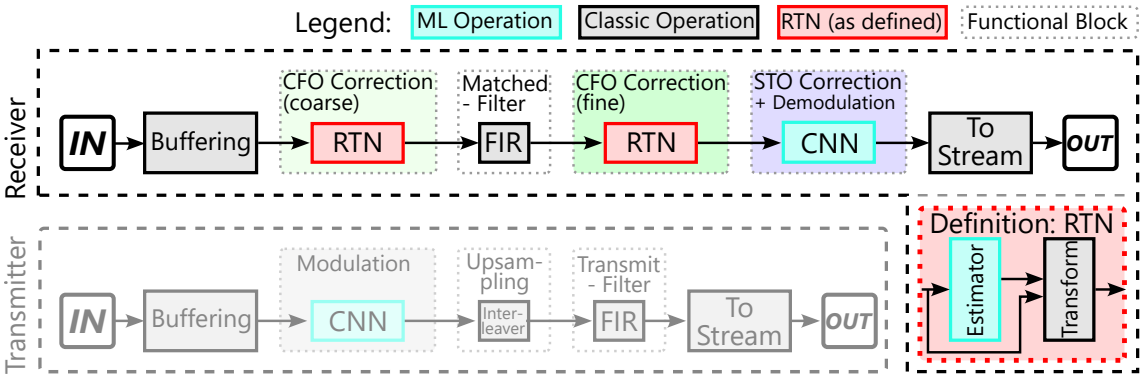


Fig. 2. Visualization of the transceiver processing chain. Machine Learning-based, classical, and RTN-based operations are shaded with turquoise, gray, and red colors, respectively. The surrounding dotted-border boxes indicate the processing blocks functionality.

DSP applications. Both types offer distinct advantages over each other in terms of functional performance and computational complexity depending on the specific task. A hybrid architecture that can not only support both ML and classical operations, but also interconnects them in a highly flexible manner to allow a complex data flow, is required to realize such AI-augmented concepts effectively. Therefore, the chosen application serves as a challenging deployment scenario which is detailed in the next section.

III. Deployment Strategy and Implementation

In this section, we outline the deployment strategy for the reference scenario on the AI- & DSP-Accelerator slice. We begin by examining the challenges associated with transitioning from a single-component to a multi-component processing system. Subsequently, we explore the capabilities of each component and their interconnectivity. From this we derive a mapping strategy that associates algorithm parts with hardware, and define a corresponding control strategy.

A. Transitioning to a Modular Computational Architecture

Porting an algorithm from a single-component to a multi-component compute platform bears a huge potential towards overall system performance, including throughput performance, power efficiency, scalability, resource optimization, and fault tolerance. The downside is typically an increased system complexity. New issues emerge, such as workload distribution and balancing, data buffering and synchronization, inter-component communication, supervision and control mechanisms, among others. These aspects are crucial and must be carefully addressed to achieve maximum effectiveness.

Since addressing all of these issues in this work is not feasible, in the remainder of this work we want to focus on the most crucial two, which are workload distribution and device intercommunication.

a) Workload distribution: Splitting up an algorithm and distributing its parts to various components requires consideration of the following:

- Matching computational requirements with native hardware capabilities. For instance, performing matrix multiplications on optimized rather than general purpose hardware, while maintaining sufficient numerical precision (e.g., fix- or floating-point format, bit-width).
- Ensuring sufficient local data cache to avoid slow external buffering.
- Maintaining equal execution times to allow efficient superscalar execution (refer to the Pipeline-Problem [23]).

In summary, the primary goal is to distribute the parts to the most suitable hardware components while ensuring rapid data transfers and interconnectivity. This ensures that the primary bottleneck of the system is compute power rather than secondary effects like memory size and data access limitations.

b) Inter-device connectivity: Efficient communication between 'neighboring' distributed parts is essential and directly impacts the overall system throughput when neglected. Especially in data-heavy applications, such as image processing and raw radio signal processing, connectivity has shown to often limit the system performance [24], [25]. However, one needs to distinguish between global and local connectivity. Global refers to the external interface that lets the device communicate to the other devices, for instance, using a system-bus or in case of the Versal the Network-on-Chip (NoC). Local refers to the internal capabilities of the device to store and move data between its low-level processing elements, such as the FPGA's Block-RAM or the AI-Engines' (see Sub-sec. III-B) local memory.

To address both of these subjects accurately, a deep understanding of the involved components is paramount. Therefore, the following subsection comprises an individual component-wise analysis before a hardware mapping can be performed.

B. Overview and Analysis of Processing Resources

In the following, we will summarize the essential components on the AI- & DSP-Accelerator slice, whose core component is the Versal AI-Core adaptive System-on-Chip [11]. Integrated inside are the Network-on-Chip (NoC), the

Processing System (PS), an FPGA (here also referred to as DSP-Accelerator), and an AI-Engine (AIE) array (part of the AI-Accelerator).

1) Network-on-Chip: The Network-on-Chip (NoC) serves as a multi-terabit interconnect between the different compute resources, memory (e.g., DDR4, HBM), and peripheral interfaces (e.g., PCIe, 100G Multirate Ethernet). Data enters the NoC through either a memory-mapped or streaming Advanced eXtensible Interface (AXI) port, which is an interface protocol defined in ARMs Advanced Microcontroller Bus Architecture (AMBA) standard. The NoC allows for flexible and dynamic access from any point within the SoC, servicing all major data generating and consuming components. The NoC is the central interface to the DDR memory controllers, external memory, on-chip memory (OCM), and Direct-Memory-Access (DMA) unit.

2) Processing System: The processing system is based on a dual-core ARM Cortex-A72 application processor and a dual-core ARM Cortex-R5F real-time processor with a primary intention for control and supervision tasks. It hosts an operating system (here: AMD-Xilinx's Petalinux) containing a Hardware Software Interface (HSI) that provides the necessary infrastructure (drivers, interfaces, configurations) to communicate with the SoC's components, such as the FPGA and the AI-engine array. The PS directly integrates an exhaustive number of peripherals, such as Gigabit Ethernet, Serial Peripheral Interface (SPI), Controller Area Network Flexible Data-Rate (CAN-FD), etc., and is furthermore connected to the NoC.

3) AI-Accelerator: The AI-Accelerator is a hardware-software co-design based on AMD-Xilinx's Deep-Learning Processor Unit (DPU) IP-core. It implements a configurable computation engine optimized for deep neural networks (DNN) using a synthesized code-block on the FPGA with a series of C++ programs deployed on the AI-Engines. The DPU is a generic ML accelerator that loads pre-compiled DNN models dynamically at run-time and executes them as needed. It currently only supports 8-bit fixed-point quantization of weights and features. For a more detailed introduction, please see [26].

Although the DPU utilizes multiple computational and memory resources internally, it can be perceived as a holistic component from a global perspective, whose main data interface (input and output) is the RAM.

4) DSP-Accelerator: For this hardware platform, the DSP-Accelerator consists solely of the SoC's FPGA. This terminology is used to ensure the generality of our proposed deployment concept and avoid being limited by resource-specific details. DSP components, such as filtering, FFT, or custom operations like transformation blocks of RTNs (see Sub-sec. II-B), are implemented on the FPGA using pre-built IP cores or RTL-level code. The available local storage is limited by the FPGA's internal RAM blocks (i.e., Block-RAM, Ultra-RAM, and distributed RAM), which total to around 200 MB for the largest chip.

5) Random Access Memory: Random Access Memory (RAM) is the only component which is not integrated inside the SoC, but placed on the AI- & DSP-Accelerator slice as a dedicated component. It is physically connected to the SoC's DDR memory controller, and made available to the other resources through the NoC, as mentioned above. The RAM is an especially crucial component, since it serves as a central element for data sharing between all mentioned components. The hereby used memory is an 8 Gb PC4-3200 DDR4 module with a maximum bandwidth of 25.6 GB/s [27].

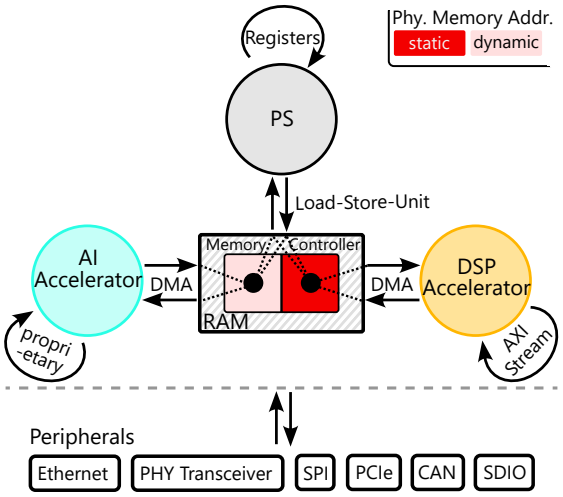


Fig. 3. Overview of the processing resources and their inter-communication using RAM. The PS utilizes its Load-Store Unit for memory access, while the AI- and DSP-Accelerator rely on DMAs, although their memory regions differ. Peripherals, such as Ethernet, Physical Transceivers, PCIe, etc., are available through one of the main processing resources.

After having introduced all relevant components, the following section elaborates on how they can be used in concert to enable a complex computation and data flow.

C. Inter-Component Communication

The interplay between various components is mainly defined by their ability to communicate with each other. On the one hand, this refers to the exchange of input, output, and intermediate data. On the other hand, this also refers to the ability to perform control and synchronization signalling to ensure smooth processing. For our further analysis we restrict our focus to data exchange only, since signalling does not present a significant bottleneck in our data-heavy scenario and is briefly handled in Sub-sec. III-E. Fig. 3 offers a glimpse into the main processing elements' data flow to and from one another. The RAM serves as a centralized storage hub, where all components obtain and store their processed data. This arrangement is inherently shaped by the centralized communication facilitated by the NoC and the individual components' lack of sufficient local storage. It therefore becomes clear, that the ability to exchange data between components is facilitated by the RAM and its access mechanisms.

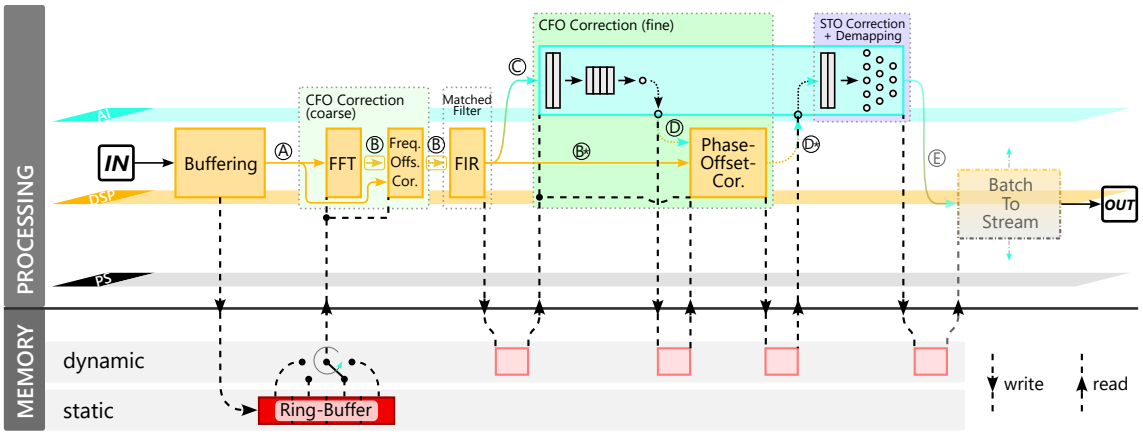


Fig. 4. Visualization of the hardware-mapping of the reference scenario's processing components and their respective data flow across the devices. Gray, yellow, and turquoise shaded colors refer to a mapping to the PS, DSP-Accelerator, and AI-Accelerator, respectively. Their memory access patterns are displayed on the bottom half of the figure, showing read and write operations into dynamic (light-red) and static (dark-red) buffer regions.

The PS has a rudimentary access mechanism solely through its Load-Store Unit and is therefore clearly not intended to actively participate in any data-intensive subtasks. This limitation extends to all peripherals that route data through the PS. Additionally, any RAM access, e.g., data copying between different regions (e.g., static and dynamic allocated buffers), is to be avoided. Since there exists no RAM-to-RAM DMA, this operation can sometimes be necessary, hence its performance is evaluated in sec. IV.

The AI-Accelerator uses separate buffers in the RAM for loading input data and storing the final result. Its access mechanism employs DMA for both directions. Due to the implementation of the underlying DPU, it is restricted to dynamically allocated buffers (i.e., accessible via virtual address), only. The internal mechanism to store intermediate results is proprietary and likely uses both RAM and FPGA-RAM.

Lastly, the DSP-Accelerator shares a similar interface like the AI-Accelerator, but it employs static memory-mapped input and output buffers configured during the operating system's build phase. AXI-Stream facilitates the transport of intermediate on-device data transfers, especially in scenarios involving buffering or a series of sequentially interconnected DSP blocks.

With the computational capabilities and data handling capacity of the components described, we are now ready to implement the reference scenario on hardware, as detailed in the next subsection.

D. Software-Hardware Mapping and Data Exchange

1) General Strategy and Data Fusion Approach: As of now, the software-hardware mapping is performed in a fully manual manner, since it enables the highest level of flexibility for exploring different design decisions. However, the mapping generally follows an intuitive and logical approach: ML-based blocks are always mapped to the AI-Accelerator if the operation is natively supported. Alternatively, the preferred fallback option is the DSP-Accelerator, followed by the PS. Classical processing blocks are preferably mapped to the DSP-Accelerator, with the PS as fallback option. It is crucial to understand that a mapping to the PS shall be avoided, even for low processing complexity steps, since its primary bottleneck is memory bandwidth, not computational power.

Equally important is to efficiently route the data between various processing components and domains. Key to efficient and fast data exchange is a zero-copy strategy. Zero-copy implies, that adjacent operations share the same memory interface, i.e., data buffer. In practise this requires aligning the memory addresses between adjacent input-output and output-input relationships. For instance, as visualized in Fig. 2, the Matched-Filter's output buffer address must equal the (fine) CFO Correction's input buffer address. Although a simple concept, aligning static (defined when building the operating system) and dynamic (defined during run-time) buffers in different memory regions and mapping virtual to physical memory address can be inconvenient, depending on the underlying drivers used.

2) Mapping of the AI-augmented Receiver Algorithm: As described in Sub-sec. II-B and shown in Fig. 2, the receiver algorithm consists of multiple interleaved classical- and ML-based processing blocks. The hardware mapping is visualized in Fig. 4. The processing flow is shown in the top half, the memory access patterns are shown in the bottom half. The same processing blocks as introduced in Fig. 2 are shown, with gray, yellow, and turquoise colors denoting their mapping to PS, DSP-, and AI-Accelerator, respectively.

It is evident, that the mixture of processing domains results in a variety of interconnections, which are labelled using letters surrounded with a circle. Implementing them efficiently

is the major challenge in a multi-component system. These interconnections are now described in detail:

- **A**: Data transfer between peripheral and DSP-Accelerator. Continuous data is buffered in a statically allocated ring buffer by the peripheral's driver, whose segments are then read by the DSP-Accelerator using its DMA.
- **B**: Data transfer between two sequential DSP-blocks. Data is directly streamed between both blocks using AXI-Stream. No significant local data storage is required, hence, RAM access is not necessary.
- **B***, **D**, and **D***: See below.
- **C**: Data transfer between DSP- and AI-Accelerator: The output of the DPS-Accelerator is stored via DMA in the dynamic input buffer of the AI-accelerator, which is then read via DMA by the AI-Accelerator.
- **E**: Data transfer between AI-Accelerator and DSP-Accelerator or PS: The output of the AI-Accelerator is stored via DMA in a dynamic buffer, which can directly be accessed by the DSP-Accelerator via DMA or by the PS using its Load-Store unit.

A significant complication is introduced within the AI-Accelerator, which features an RTN block whose transformation component (Phase-Offset-Correction) is classical, and hence deployed on the DSP-Accelerator. This effectively leads to a classical component enclosed within two ML components. While in general it is possible to deploy multiple separate neural networks (NN) on the AI-Accelerator to address such an issue, the approach of a single NN is pursued here. This is made possible by utilizing the DPU's Custom-Layer functionality, which allows to execute arbitrary code, and hence even a block on the DSP-Accelerator, in-between two layers of the NN. In our case, the following interconnections are required:

- **B***: Data transfer between a stand-alone and an interleaved DSP-block: Since the data cannot be directly stream between both DSP-blocks due to additional data dependencies on the AI-Accelerator, the input buffer is used as temporary storage, and simply read via DMA when required.
- **D**: Data transfer of an intermediate result between AI- and DSP-Accelerator: The intermediate result is automatically stored in a dynamic buffer by the AI-Accelerator and is then read via the DSP-Accelerator's DMA.
- **D***: Same as D, but in reverse direction and order.

Finally, after defining the hardware mapping, interconnections, and memory access patterns, we define a control procedure that supervises the complete computation process.

E. Process Control Procedure

The computation process is controlled by a Python application running on the PS. For simplicity, computation is executed in a one-shot, non-pipelined manner. Pipelining is discussed in Sec. V. The application has the following tasks:

- Wait asynchronously for a ring buffer segment to be filled with data from the RF frontend slice.
- Start FFT, frequency-offset correction, and FIR filter component on DMA-Accelerator, wait until completion.
- Start AI-Accelerator. Upon reaching the phase-offset correction block, the DPU's Custom-Layer API executes a custom script and supplies its dynamic input and result buffer addresses, which triggers the DSP-Accelerator's DMA for both dynamic buffers. Upon completion of the block, the result is written via DMA into the dynamic output buffer supplied by the Custom-Layer API, and control is returned back to the AI-Accelerator.
- Upon completion of the AI-Accelerators execution, the final result is available in the AI-Accelerators dynamic output buffer and can be further routed to its destination.

With the procedure defined, the reference scenario is fully implemented. We can now begin to evaluate the performance of this implementation, which is detailed in the next section.

IV. Performance Evaluation

This goal of this section is to explore the practical performance of our deployment concept. Since the AI-Accelerator's performance has already been sufficiently analyzed [25], [26], [28], [29], and the DSP-Accelerator's performance is highly predictable due to its reliance on the FPGA, we focus our analysis on the components' inter-communication capabilities. After reviewing the measurement methodology and system configuration, we summarize our findings and provide a critical discussion on possible improvements.

A. Measurement Methodology and System Configuration

As observed in Sub-sec. III-D, a complex computation graph necessitates various communication interfaces between the involved components. Only a single inefficient interface can have a significant impact on the overall system performance, which is why this section evaluates the achievable data transfer speeds between all involved components. In the following we list the related interfaces, calculate their theoretical maximum performance, and detail our measurement strategy to evaluate their realistic performance. For all measurements we use a buffer size of 64 MiB.

a) AI-Accelerator - RAM: The AI-Accelerator utilizes DMA to read and write its input and output data. Since the underlying DPU is a proprietary and encrypted hardware-software co-design, its realistic bandwidth to memory can only be estimated. Since running an "empty" model with zero layers (to remove any computation burden and hence time) is not supported, we can only measure the total inference time for valid neural networks. Therefore, we compile and run the execution time of multiple models that each contain the exact same layer (here: convolutional layer with resolution 512x512, kernel 7x7, 128 input- and output-channels, same padding) a different amount of times, which lets us determine the isolated layer run-time, and therefore the raw data transmission time by subtracting the layer execution time. Note, that the model loading time is (correctly) not included in this measurement, as model loading is performed prior to execution. This method

TABLE I
PERFORMANCE ANALYSIS OF THE PS, AI-, AND DSP-ACCELERATOR'S THEORETIC AND MEASURED COMMUNICATION CAPABILITIES WITH THE MEMORY. THE EVALUATED METRIC IS THE DATA RATE.

Operation	Route			Throughput [GiB/s]		
	Source	via	Destination	Forward	Backward	Theoretic Max.
Data Feed	AI-Accelerator	DMA	RAM *(dynamic)*	3.19		4.47 - 26.82
Data Feed	DSP-Accelerator	DMA	RAM *(dynamic)*	14.59	14.91	17.88
DSP Pipeline	DSP-Accelerator	AXI-Stream	DSP-Accelerator	-	-	17.88
Copy Buffer	RAM	PS Load-Store-Unit	RAM	4.34	4.29	8.94

computes the average transfer rate for reading and writing, the separate values cannot be identified.

The theoretical maximum rate can be computed based on the bus-width of the corresponding input and output data AXI-port[1] ($128 \cdot$ batch-size), for a batch-size up to six. For a clock frequency, which is typically set to $300\,\mathrm{MHz}$, this results in a maximum theoretic data rate of 4.47 GiB/s per Batch, totalling to 26.82 GiB/s for multi-batch execution. The configuration analyzed here utilizes single-batch execution only.

b) DSP-Accelerator - RAM: The DSP-Accelerator utilizes DMA to access the RAM. As a first-order approximation, the speed depends on the DMA's frequency and bus-width and can be calculated as

$$\text{Transfer Speed} = f_{\text{DMA}} \cdot \text{bus-width}_{\text{DMA}}. \tag{1}$$

Although f_{DMA} can reach values up to $500\,\mathrm{MHz}$ [29], a popular setting that avoids timing problems with most IP-cores (including the DPU) is $300\,\mathrm{MHz}$, which is also used here. Although the speed is expected to be similar in forward (reading from the RAM) and backward (writing to the RAM) direction, it is measured in both directions by implementing an "empty" receiving IP-core on the FPGA (to avoid transfer stalls caused by computation) when reading data, and a constant data source when writing data to the RAM. The DMAs are controlled via the python py-uio library [2].

c) DSP-Accelerator - DSP-Accelerator: Communication between two sequential DSP-blocks is implemented via AXI-Stream, which has an intrinsic deterministic performance. Its performance is calculated as

$$\text{Transfer Speed} = f_{\text{AXI-Stream}} \cdot \text{bus-width}_{\text{AXI}}. \tag{2}$$

With $f_{\text{AXI-Stream}} = 300\,\mathrm{MHz}$ and AXI-buswidth $= 512$, the interface speed results to $17.88\,\mathrm{GiB/s}$. No measurement has been performed, since the calculated speed is expected to be identical with the practical speed.

d) PS - RAM: The PS accesses the RAM via its Load-Store Unit, and hence can be easily evaluated by timing the corresponding system-native memory copy functions. Since our control program utilizes python, we use *ctypes*[3] system-native *memmove* method to evaluate the transfer speeds from a static to a dynamically allocated buffer and in reverse direction. On a first-order approximation, the theoretical limit is defined by the max. CPU clock frequency ($f_{\text{CPU,max}} = 1.2\,\mathrm{GHz}$) and the memory bus-width (64 bits), resulting to 8.94 GiB/s.

[1]https://docs.xilinx.com/r/en-US/pg389-dpucvdx8g/Interface-Ports ref. to port *Mxx_IMG_AXI*.

[2]py-uio library website: https://github.com/mvduin/py-uio

[3]ctypes python library: https://docs.python.org/3/library/ctypes.html

B. Measured Performance Results

Table I summarizes the benchmarked operations and their results. The PS-based buffer copy mechanism achieves a realistic rate of 4.3 GiB/s which is rougly identical in both directions, realizing 48% of its theoretical speed. The interface between AI-Accelerator and RAM achieves a 71% utilization with 3.19 GiB/s transfer speed. The DSP-Accelerator is able to access the memory with a much higher rate of 14.59 and 14.91 GiB/s in forward and backward direction, respectively, realizing 82 % of its maximum potential. Lastly, as mentioned before, the on-device data transfer of the DSP-Accelerator achieves 17.88 GiB/s.

C. Discussion and Future Improvements

The above results are promising, indicating efficient utilization of the overall system through our deployment approach. The AI- and DSP-Accelerators demonstrate effective coordination in exchanging process data. Notably, the DSP-Accelerator's utilization of multiple sequential processing blocks proves highly effective, achieving data transfers at theoretical maximum speeds.

Measurements confirm that the PS's access inefficiency to memory poses a potential performance bottleneck if utilized, for instance, when performing buffer copy operations. By utilizing cache coherent memory access (not used in this work), an increase in bandwidth is expected. Another reasons for the gap between theoretical and measured communication speed for the other components is the non-optimized configuration of DMA frequencies and bus widths.

Not analyzed in this work is the additional overhead incurred by data conversion or formatting (e.g., from INT8 to FLOAT32) between different processing components, which is expected to have a minimal impact on the overall performance. These conversions would typically be deployed on the DSP-Accelerator.

V. CONCLUSION AND FUTURE WORK

In conclusion, this paper has demonstrated the successful integration of artificial intelligence (AI) processing capabilities into high-performance satellite platforms, specifically within the context of telecommunications signal processing. By interleaving AI-based computation blocks within an FPGA-based

processing framework, we have shown the feasibility of enhancing traditional data processing workflows with advanced AI techniques. Through a thorough evaluation of hardware components and communication interfaces, we have achieved a seamless deployment of an AI-augmented signal processing pipeline on state-of-the-art telecommunications processor hardware. Our results highlight the potential for improving satellite platform performance and efficiency through the intelligent use of AI technologies.

Future work:

Moving forward, there are several avenues for future research and development in this area. Firstly, further optimization of communication interfaces and hardware configurations could lead to an enhanced system performance and efficiency. Moreover, pipelining the data flow bears significant overall performance potential, ideally keeping the DSP- and AI-Accelerator occupied at all times and removing the idle periods when waiting for prior computation to finish. This would require the introduction of additional buffer memory between each pipelined computation stage and a modification to the zero-copy strategy to preserve optimal communication. A further extension is to realize a multi-DPU-based AI-Accelerator to run multiple different neural networks simultaneously. Additionally, a highly relevant future feature could be to implement real-time adaptivity to the AI-related processing components, such as enabling in run-time weights update or neural network training. Lastly, the topic of redundancy, system behaviour in case of component faults, and options to restore service should be addressed.

References

[1] M. Ghiglione and V. Serra, "Opportunities and challenges of AI on satellite processing units," in *Proceedings of the 19th ACM International Conference on Computing Frontiers*, ser. CF '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 221–224. [Online]. Available: https://doi.org/10.1145/3528416.3530985

[2] A. Koch, A. Krstova, F. Hegwein, M. C. De Lera *et al.*, "On-Board Anomaly Detection on a Flight-Ready System," in *2023 European Data Handling & Data Processing Conference (EDHPC)*, 2023, pp. 1–4.

[3] G. Giuffrida, L. Fanucci, G. Meoni, M. Batič *et al.*, "The -Sat-1 Mission: The First On-Board Deep Neural Network Demonstrator for Satellite Earth Observation," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–14, 2022.

[4] G. Zhu, D. Liu, Y. Du, C. You *et al.*, "Toward an Intelligent Edge: Wireless Communication Meets Machine Learning," *IEEE Communications Magazine*, vol. 58, no. 1, pp. 19–25, 2020.

[5] N. Destrycker, W. Benoot, J. Mattias, I. Rodriguez, and D. Steenari, "EDGX-1: A New Frontier in Onboard AI Computing with a Heterogeneous and Neuromorphic Design," in *2023 European Data Handling & Data Processing Conference (EDHPC)*, 2023, pp. 1–6.

[6] Brainchip, "Benchmarking AI Inference at the Edge: Measuring Performance and Efficiency for Real-World Deployments," 2023, https://brainchip.com/wp-content/uploads/2023/01/BrainChip_Benchmarking-Edge-AI-Inference-1.pdf [Accessed: 10. March 2024].

[7] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya *et al.*, "Loihi: A Neuromorphic Manycore Processor with On-Chip Learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[8] G. Benelli, G. Giuffrida, R. Ciardi, D. Davalle *et al.*, "GPU@SAT, the AI enabling ecosystem for on-board satellite applications," in *2023 European Data Handling & Data Processing Conference (EDHPC)*, 2023, pp. 1–4.

[9] P. Ghiglino, M. Harshe, D. Stenaari, and L. Mansilla, "AI/ML Inference Engine Software for High-Reliability applications on Space Qualifiable Hardware," in *2023 European Data Handling & Data Processing Conference (EDHPC)*, 2023, pp. 1–11.

[10] M. Petry, B. Parlier, A. Koch, and M. Werner, "Auto-Regressive RF Synchronization Using Deep-Learning," in *IEEE International Conference on Machine Learning for Communication and Networking*, 2024, [accepted, pre-print available at https://www.bgd.ed.tum.de/pdf/2024_RFSynchronization_ICMLCN_Petry.pdf].

[11] B. Gaide, D. Gaitonde, C. Ravishankar, and T. Bauer, "Xilinx Adaptive Compute Acceleration Platform: VersalTM Architecture," in *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 84–93. [Online]. Available: https://doi.org/10.1145/3289602.3293906

[12] A. Bourdoux, A. N. Barreto, B. van Liempd, C. Lima *et al.*, "6G White Paper on Localization and Sensing," 06 2020.

[13] J. Hoydis, S. Cammerer, F. A. Aoudia, A. Vem *et al.*, "Sionna: An Open-Source Library for Next-Generation Physical Layer Research," 2023.

[14] A. Alkhateeb, S. Jiang, and G. Charan, "Real-Time Digital Twins: Vision and Research Directions for 6G and Beyond," *IEEE Communications Magazine*, vol. PP, pp. 1–7, 11 2023.

[15] S. Cammerer, F. A. Aoudia, S. Dörner, M. Stark *et al.*, "Trainable Communication Systems: Concepts and Prototype," *IEEE Transactions on Communications*, vol. 68, no. 9, 2020.

[16] M. B. Fischer, S. Dörner, S. Cammerer, T. Shimizu *et al.*, "Adaptive Neural Network-based OFDM Receivers," in *2022 IEEE 23rd International Workshop on Signal Processing Advances in Wireless Communication (SPAWC)*, 2022, pp. 1–5.

[17] S. Dörner, S. Rottacker, M. Gauger, and S. t. Brink, "Bit-wise Autoencoder for Multiple Antenna Systems," in *2021 17th International Symposium on Wireless Communication Systems (ISWCS)*, 2021.

[18] S. Cammerer, J. Hoydis, F. Aoudia, and A. Keller, "Graph Neural Networks for Channel Decoding," 12 2022, pp. 486–491.

[19] W. Xu, Z. Zhong, Y. Be'ery, X. You, and C. Zhang, "Joint Neural Network Equalizer and Decoder," in *2018 15th International Symposium on Wireless Communication Systems (ISWCS)*, 2018, pp. 1–5.

[20] 3GPP, ""3rd Generation Partnership Project; Technical Specification Group Radio Access Network; New SID on AI/ML for NR Air Interface"," 3GPP, Tech. Rep. RP-212708, 2021.

[21] ——, ""3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Study on Integrated Sensing and Communication (Release 19)"," 3GPP, Tech. Rep. TR 22.837, 2024.

[22] T. O'Shea and J. Hoydis, "An Introduction to Deep Learning for the Physical Layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563–575, 2017.

[23] J. L. Hennessy and D. A. Patterson, *Computer Architecture, Sixth Edition: A Quantitative Approach*, 6th ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2017.

[24] N. P. Jouppi, C. Young, N. Patil, D. Patterson *et al.*, "In-Datacenter Performance Analysis of a Tensor Processing Unit," *SIGARCH Comput. Archit. News*, vol. 45, no. 2, p. 1–12, jun 2017. [Online]. Available: https://doi.org/10.1145/3140659.3080246

[25] M. Petry, G. Wuwer, A. Koch, P. Gest *et al.*, "Accelerated Deep-Learning Inference on the Versal adaptive SoC in the Space Domain," in *2023 European Data Handling & Data Processing Conference (EDHPC)*, 2023, pp. 1–8.

[26] M. Petry, P. Gest, A. Koch, M. Ghiglione, and M. Werner, "Accelerated Deep-Learning inference on FPGAs in the Space Domain," in *Proceedings of the 20th ACM International Conference on Computing Frontiers*, ser. CF '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 222–228. [Online]. Available: https://doi.org/10.1145/3587135.3592763

[27] M. Technology, "MTA9ADF1G72AZ DDR4 SDRAM VLP UDIMM," 2024, https://media-www.micron.com/-/media/client/global/documents/products/data-sheet/modules/unbuffered_dimm/ddr4/adf9c1gx72az.pdf?rev=4d6f78da73bb4d5f89724be3e7073314 [Accessed: 09. March 2024].

[28] A. Al-Zoubi, G. Martino, F. H. Bahnsen, J. Zhu *et al.*, "CNN Implementation and Analysis on Xilinx Versal ACAP at European XFEL," in *2022 IEEE 35th International System-on-Chip Conference (SOCC)*, 2022, pp. 1–6.

[29] M. Wierse, "Bachelor Thesis: Evaluation of Xilinx Versal Device," 2023, https://doi.org/10.3929/ethz-b-000600880.

4.4 Discussion

The above publications present very different approaches on how DL can be applied to the physical layer ([RQ-4]), exploring an AI-native and an AI-augmented system. For both integration levels we've addressed benefits and disadvantages with respect to overall system performance (both "accuracy" and "throughput capacity"), implementation complexity, dependability on resource constraints, and on-board reconfigurability.

Based on these points the overall benefits can be attributed to Level-1 and Level-2 systems (AI-supported and AI-augmented). Even though specifically from a hardware deployment perspective we come to this conclusion, the current reality is that overall research is primarily driven from a software and/or algorithm perspective, and few emphasis is given on how practical the proposed solutions are. There is a gaping void between what's possible with respect to real-life constraints, such as the availability of layer types, restricted model branching complexity and size, resource limitations, and quantization, and what the proposed solutions demand. This gap might be one of the reasons why the wide-spread use of AI in communications has still not picked up for terrestrial systems [108].

Recent surveys on novel AI-based techniques that address current problems in wireless communications reveal that current models are still too complex or presuppose advanced functionality [107–109, 179]. For example, [96] proposes a fully AI-native communication systems with a complex frame synchronization mechanism that uses multiple NNs in parallel to perform various tasks, such as STO and CPO estimation, that produces a highly intertwined data flow, requiring a high level of flexibility on the inference platform (here: GPU). A different situation can be observed in [180] which proposes a very low-complexity solution utilizing only a single hidden layer feed-forward network to perform channel estimation and equalization to combat fading and high-power amplifier-caused non-linearities. Although the forward-pass is computationally viable, it presupposes an online-learning mechanism to dynamically adapt to changes, which is only rarely supported on platforms different than CPU and GPU.

However, more balanced approaches do exist. In [94, 181] Ait Aoudia *et al.* propose to augment an OFDM multi-user MIMO receiver architecture by only replacing two specific classical components, i.e., channel estimator and demapper, with DL-based pendants. From an implementation perspective, the sparse utilization of ResNet/CNN-based components forms a hybrid processing chain similar to the system proposed in [Pub4.B], which scores many points regarding possible realizability. Other approaches that selectively apply ML for communications build on attention mechanisms [130, 177].

Thanks to the generic nature of physical layer signal processing we could build the research in this dissertation on the foundations of AI for terrestrial communication technology. Although comparative approaches for or with applicability to the space domain are especially rare (e.g. an FPGA-based autoencoder implementation [182]), since the cost of evaluation hardware, a limited possibility to publish due to company IP ownership, and simply the small market size strongly limit open research, we can draw additional conclusions by shifting our focus from 5G/6G back to space.

The most prominent observation by different national space agencies and industry

practitioners is the general lack of sufficient processing resources due to the unavailability of suitable hardware [183–185]. The only resolve is to utilize COTS-components, which, although initially solving resource restrictions, introduce other problems, such as unreliability due to radiation-induced faults. In [186] Lange *et al.* argue that such issues of natural hazards are barely addressed in the space community due to common ML tool chains not offering native interfaces for directly inducing and studying radiation-related effects, and suggests a possible solution by incorporating resilience to these issues directly into NNs by inducing faults at training-time. Another problem is the general undersupply of raw training data gained in space which in some situations makes it difficult to both train NNs and validate their performance prior to deployment [187]. Often overseen is also the limitation in numerical precision or resolution due to quantization in both raw data and processing capabilities [188].

One topic which is mostly neglected (also in this dissertation) is the aspect of fairness in ML. While not directly applicable to the specific functionality implemented in [Pub4.A] and [Pub4.B], fairness can be a crucial concern for communication systems when multiple users with heterogeneous demands, devices, and personal data are involved [189]. One can easily imagine that ML models onboard satellites, for example when employed within 5G/6G NTNs, might unintentionally prefer users from certain regions.

To avoid unexpected problems during the due diligence phase (i.e., the standardized validation process for space applications as discussion in Sec. 3.3), it is encouraged to include the above named concerns as early as possible in the design phase.

4.5 Summary

This chapter investigated how DL can be applied to the physical layer while keeping in mind its practical realization on resource-constrained systems. We explored the different levels of how deep AI/ML components can be conceptually integrated into the signal processing chain by following a top-down approach. We began by proposing a fully AI-native communication system based on a bit-wise autoencoder that performs essential processing steps solely via means of NNs to allow unidirectional communication in [Pub4.A]. After identifying various shortcomings with the most profound being related to implementation feasibility, we transitioned to an AI-augmented integration level in [Pub4.B] by applying DL-based components only selectively to address specific tasks. Although the resulting hybrid processing pipeline implies much higher development effort, we showed that this refined approach, if implemented optimally, can improve functional system performance (here: lower BER), while simultaneously enabling higher data throughput. A contextualization within the current state of the research regarding 5G/6G concluded this chapter. In summary, we argue that AI-augmented systems, such as autoencoders, are no plug-and-play technology yet and demand specific hardware platforms and implementation effort [190]. For both terrestrial and spaceborne applications the practical realization of AI in the field of communications is still waiting to be picked up, with very first terrestrial products, such as Qualcomm's Snapdragon X80 modem with AI-support just getting out on the market.

5 Artificial Intelligence-capable 5G Non-Terrestrial Network Base Station Concept

One of the most promising novelties in communications with great potential to amplifying the necessity of deploying AI on satellites is the introduction of NTNs in the context of 5G. Serving connectivity from spaceborne or airborne stations directly to devices that support modern mobile cellular standards like Long Term Evolution (LTE) and 5G heralds a paradigm-shift in our capabilities to communicate. This chapter investigates solutions for satellite-based base stations from a payload perspective. After reviewing key aspects of 5G-NTN based on the current state of the standard we motivate the development of a fully on-board gNodeB for the regenerative architecture track. Afterwards, [Pub5.A] proposes and investigates a system concept for an AI-enabled fully-featured gNodeB on AMD-Xilinx's VERSAL AI Core platform and reveals challenges with respect to performance and scalability. Sub-sec. 5.2.1 delves into the inner workings of the gNodeB to discuss the primary shortcomings in an architectural context and proposes an improved concept by integrating additional hardware capabilities into the setup. Sec. 5.3 contextualizes the approach within current developments in research and industry and discusses how future AI demands foreseen within the standard are applicable to the introduced architecture. A summary concludes this chapter.

5.1 Advantages of Regenerative Payloads on Spaceborne Base Stations

The phrase NTN has evolved into an umbrella term involving any infrastructure that is not stationed on ground in the context of 5G. Although the primary focus in standard development by 3GPP is set on spaceborne stations in GEO and LEO, recent interest has also been sparked for airborne stations, such as balloons, airships, and High Altitude Platform Systems (HAPS). HAPS have even been listed in the Top 10 emerging technologies report by the World Economic Forum in 2024 [191]. In the rest of this chapter we restrict our focus to spaceborne systems. The introduction of NTNs is motivated by the following three main goals [192]:

- Roll out of 5G in unserved areas and areas that are not accessible via terrestrial means, e.g., islands, mountains, rural areas, and disaster-affected areas, including end-points in the air and the sea, i.e., aircraft and maritime vessels, respectively.
- Reinforce reliability and provide service continuity for Machine-to-Machine (M2M) communication, Internet of Things (IoT), and moving platforms, e.g., ships, high speed trains, busses, and cars.

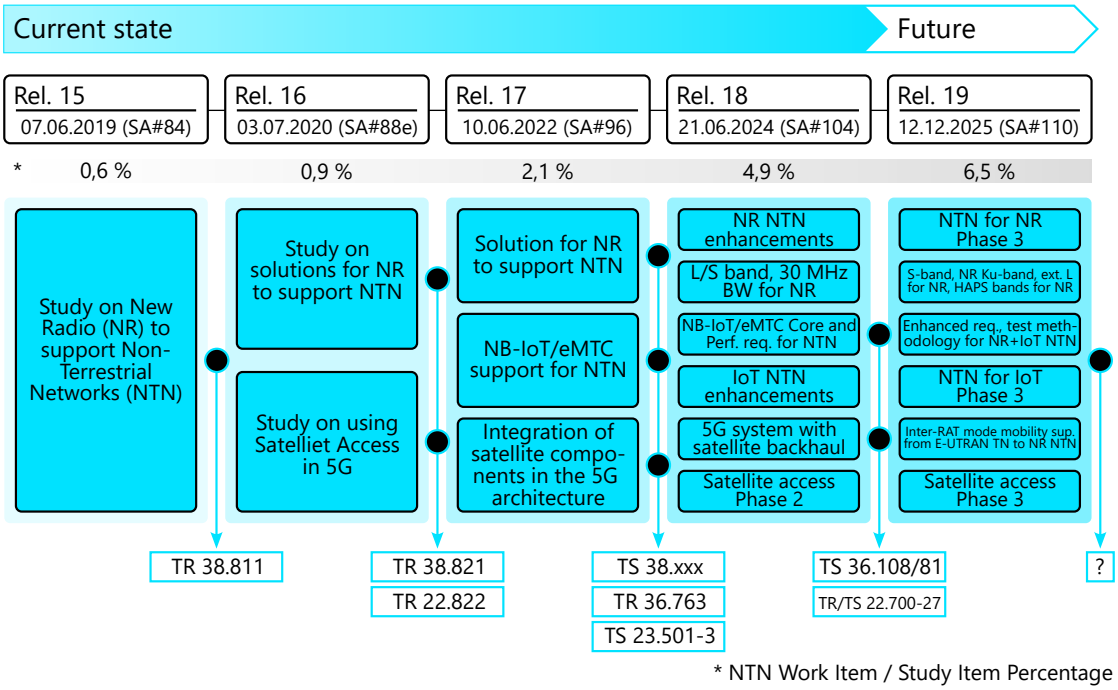
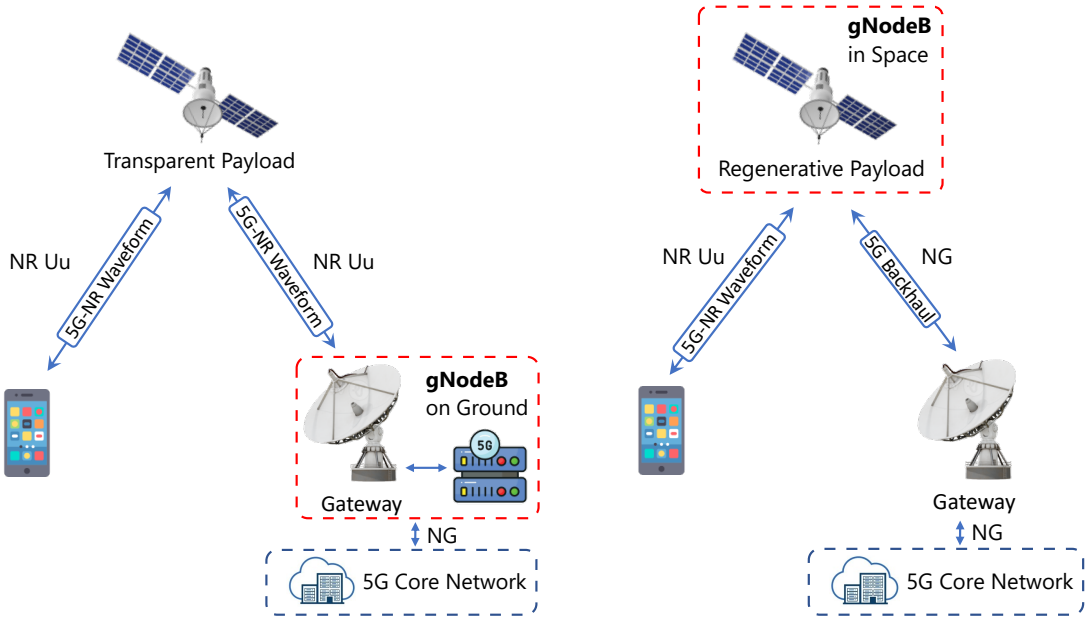


Figure 5.1: Visualization of the major study and work items regarding Non-Terrestrial Networks in the 5G standard development and the resulting technical reports and specifications. Adapted from [194].

- Enable network scalability towards the network edge by providing efficient multicast/broadcast through nodes with large coverage regions.

In summary, the integration of NTN into terrestrial networks is thought to improve multiple KPIs, including coverage, user bandwidth, system capacity, service reliability and availability, energy consumption, and connection density [193].

The topic of NTNs was first approached formally through 3GPP in Rel. 15 (2019). Fig. 5.1 gives an overview of the main milestones towards the integration of NTN within the 5G standard until today. Rel. 15 started the conversation by introducing a study item that investigates the feasibility of 5G-New Radio (NR) to support NTNs [192]. It defines the NTNs deployment scenarios and system parameters such as architecture, altitude, and orbits, and identifies key impact areas on the NR interface that need to be addressed to enable this feature, such as cell pattern, Doppler effect, propagation delay, and protocol-related issues. Among introducing various network architectures and different access schemes for end-users, most relevant for satellite payload design is the definition of a transparent (bent-pipe) and regenerative operation principle [192]. Fig. 5.2 depicts the two deployment schemes, with Fig. 5.2a showing the transparent architecture, and Fig. 5.2b showing the regenerative architecture. Both architectures have significant implications on network performance KPIs like Quality-of-Service (QoS), cell



(a) Transparent architecture **(b)** Regenerative architecture

Figure 5.2: Visualization of the two major 5G-Non-Terrestrial Network architectures. The transparent (bent-pipe) architectures operates a gNodeB on ground and relays its signals using a transparent satellite. The regenerative architecture operates a gNodeB directly on-board a regenerative satellite in space and facilitates connectivity to the 5G core network via the gateway.

coverage, and User Equipment (UE)-density, on the air interface like spectral efficiency, link budget, and waveform support, and finally on satellite capabilities like RF and on-board processing capacities, beam-forming, inter-satellite links, and the feeder-link. The key difference between both architectures is the location of the actual base station (and its corresponding processing hardware), which is either situated on the ground next to the gateway, or is partially or fully integrated on-board the satellite.

Rel. 16 picks up and consolidates the challenges revealed by Rel. 15 and investigates adaptions and solutions to mitigate these, primarily focusing on the physical layer, layer 2, and layer 3 [195]. Furthermore, a study on satellite access makes the first steps towards a unified TN-NTN by investigating service continuity when switching between both networks [196].

The resulting technical reports form the foundation for the actual specifications developed and integrated into the standard in Rel. 17. The recently standardized Rel. 18 incorporates enhancements for NR and IoT-NTN by focusing on mobility scenarios, such as TN-NTN mobility and conditional handovers, as well as infrastructure, such as satellite backhaul and L/S-band specifications.

Rel. 19 is currently ongoing and has a particular focal point on infrastructure, specifically on the regenerative architecture for NTN. Multiple features are being standardized, such as UE-satellite-UE connectivity which gives the gNodeB the capability to facilitate direct communication between two UEs without routing data over the core network, enhanced multicast/broadcast, store-and-forward operation for IoT, and inter-satellite links.

The benefits of regenerative architectures are recognized and consecutively standardized. However, as is common practice, the implementation of the standard is left to industry. In the following, we propose an implementation of a fully on-board gNodeB.

5.2 Pub5.A: AI-enabled 5G base-station with Hardware Acceleration for Non-Terrestrial Networks on a Space-Grade System-on-Chip

Authors: Michael Petry, Raphael Mayr

Publication Medium: The first joint European Space Agency SPAICE Conference on AI in and for Space, 2024

Digital Object Identifier: https://doi.org/10.5281/zenodo.13885600

Thesis contextualization: With this paper we aim to achieve two goals: First and foremost, fueled by the progress on regenerative architectures in the 5G standardization process, we propose and implement a design-partitioning for efficiently deploying OpenAirInterface's gNodeB on AMD-Xilinx's heterogeneous VERSAL AI Core aSoC platform. Second, by anticipating future AI workloads that are expected to be integrated into the gNodeB, we integrate AMD-Xilinx's generic AI inference accelerator on the same platform by efficiently sharing resources, answering [RQ-6]. This paper lays the foundations for an AI-capable gNodeB platform for in-orbit deployment which is further optimized in Sub-sec. 5.2.1.

CRediT author statement:

Michael Petry:	55 %	Investigation, Conceptualization, Methodology, Software, Data curation, Formal analysis, Visualization, Writing
Raphael Mayr:	45 %	Investigation, Conceptualization, Methodology, Software, Data curation, Validation, Writing

Proceedings of the 1st SPAICE Conference on AI in and for Space | Pages 308 - 312 DOI: 10.5281/zenodo.13885601

AI-enabled 5G base-station with Hardware Acceleration for Non-Terrestrial Networks on a Space-Grade System-on-Chip

Michael Petry[1,2] and Raphael Mayr*[1]

[1]*Telecom & Navigation Department, Airbus Defence and Space GmbH, Munich, Germany*
[2]*Professorship for Big Geospatial Data Management, Technical University of Munich, Munich, Germany*

Although 5G Non-Terrestrial Networks envision regenerative satellites with on-board 5G modems to offer the highest degree of user experience, current base-station technology prohibits practical deployment in space. We bridge this gap by presenting a shared platform that unifies gNodeB and Artificial Intelligence (AI) functionality on a hardware-accelerated, space-grade System-on-Chip. By porting OpenAirInterface's gNodeB onto this architecture and offloading DSP-intensive processing tasks of the Physical layer to its Field Programmable Gate Array, benchmarks indicate a reduction in resource requirements by over one order of magnitude, enabling in-space processing and parallel execution of future AI workloads. Moreover, hardware-accelerated support for the Linux Industrial I/O Subsystem interface enables efficient connectivity to space-ready RF frontends. A discussion on further steps towards core network integration concludes this paper.

1 Introduction

The expansion of 5G connectivity through Non-Terrestrial Networks (NTNs) marks a significant advancement in the pursuit of global communication coverage. By leveraging satellite and space-based platforms, NTNs promise to extend high-speed cellular networks to remote and underserved regions, facilitating critical applications in disaster recovery, maritime and aeronautical communications, and the Internet of Things (IoT). Among other use-cases, such as business-to-business (B2B) and machine-to-machine (M2M), NTNs initiated a race towards space-dominance that culminated in the creation of multiple satellite mega constellations [1, 2].

One of the key goals of 5G NTN is to enable unmodified smartphones to directly connect with satellite constellations, referred to as Direct-To-Cell (D2C) access. Primarily led by private companies, different approaches with respect to constellation type, and more importantly, on-board radio frequency (RF) technology are being developed. First successful field trials, such as Lynk's Sub-1-GHz GSM/2G-based test in 2020 [3, 4], or most recently SpaceX's LTE/4G-based demonstration in 2024 to D2C-modified satellites in their Starlink fleet [5], confirmed principle feasibility. Expansion into geostationary earth orbit (GEO) is also being explored [6].

Despite its potential, the deployment of 5G NTNs entails formidable challenges. These can be broadly categorized into wireless propagation-related issues, the integration of NTNs in existing Terrestrial Networks (TN), and computational-related challenges. Additionally, efforts for augmenting the 5G stack with Artificial Intelligence (AI) across various layers, especially the physical, MAC, and application layer, are gaining momentum, as evidenced by the deployment of embedded AI accelerators in state-of-the-art smartphones. This trend is expected to expand to the base-station side [7, 8], necessitating on-board AI execution capabilities. While the first two categories are being addresses in the 5G NTN standard development, practical realization of the gNodeB on space-grade hardware remains a largely unaddressed issue. Unique environmental challenges in space, such as lower power capacities, radiation, extreme temperatures, and distinct processing technologies, necessitate innovative solutions to transition from testbed setups to commercially scalable implementations.

This paper presents a processing solution by realizing a 5G gNodeB on AMD-Xilinx's space-grade Versal AI Core adaptive System-on-Chip (aSoC) series. The main contributions are the following: By utilizing the OpenAirInterface's (OAI) gNodeB implementation [1], the 5G stack is ported to the ARM-based Versal aSoC, while DSP-intensive components are offloaded via hardware acceleration to the Field Programmable Gate Array (FPGA). Furthermore, we extend OAI's radio head with support for the Linux In-

*Corresponding author. E-Mail: raphael.r.mayr@airbus.de

[1] https://openairinterface.org/

Proceedings of the 1st SPAICE Conference on AI in and for Space | Pages 308 - 312 DOI: 10.5281/zenodo.13885601

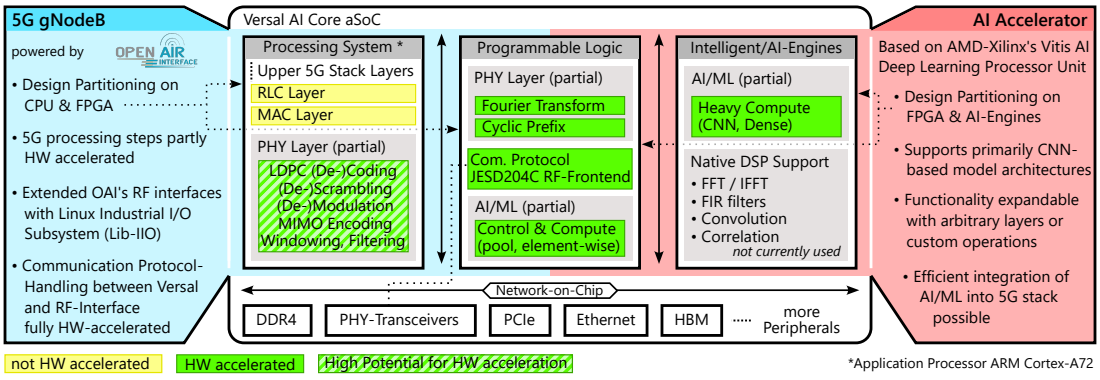


Figure 1: System Architecture for shared 5G gNodeB and AI/ML application on an AMD-Xilinx Versal aSoC.

dustrial Input/Output (IIO) Subsystem. Moreover we address future AI demands by implementing proof-of-concept (PoC) AI functionality on the platform.

The rest of this paper is structured as follows: In Sec. 2 we present the concept of the AI-augmented 5G base-station, outline design decisions w.r.t. processing offloading to the SoC's components, and introduce the PoC AI application. Sec. 3 analyzes the results of hardware acceleration in terms of speed and resource burden for three PHY configurations. Sec. 4 discusses the presented approach, discusses AI integration, elaborates on further optimization, and concludes this work.

2 AI-augmented 5G base-station

2.1 Concept and Target Hardware Platform

Artificial Intelligence is currently being explored on various fronts within cellular networks and is expected to benefit a variety of aspects, such as network and routing optimization, improved user experience and quality of service, energy efficiency, and more. While the integration of AI/ML computational capabilities into terrestrial base-stations is being investigated primarily with GPU-based approaches [9, 10], space-bound systems lack a hardware platform that can satisfy all demands.

With this work we present a shared system design between gNodeB and AI-Accelerator on the novel Versal AI Core System-on-Chip. The Versal facilitates heterogeneous processing with a state-of-the-art processor, hardware-accelerated DSP capabilities on programmable logic (FPGA), and a so-called AI-engine array, efficiently connected by a programmable Network-on-Chip (NoC). The central idea is that the gNodeB and AI Accelerator share this platform. The design split is visualized in Fig. 1, with the blue-shaded and red-shaded components denoting gNodeB and AI Accelerator, respectively. To exploit the heterogeneous architecture effectively, the gNodeB and AI Accelerator implement a design partitioning, distributing control and processing tasks to the most suitable hardware component. The detailed implementation of both components is discussed in the following two sub-sections.

2.2 Hardware-Accelerated 5G gNodeB

The 5G gNodeB's processing architecture leverages a Hardware/Software Codesign approach. Computationally intensive and repetitive tasks are offloaded to the FPGA, while tasks that require a more generic processor, such as control functionality and higher layers of the 5G stack are handled by the Versal's Application Processing Unit (APU). This functional split maximizes platform utilization and enhances performance and power efficiency, addressing a crucial issue in non-terrestrial environments.

2.2.1 Functional Split

Figure 1 illustrates the 5G stack's layers and their corresponding positions within the processing platform. As indicated by the green-shaded items, all major Physical-layer components, such as Coding, Modulation, Filtering, etc., and their inverse operations, are highly suitable for offloading. Currently, Discrete and Inverse Discrete Fourier Transforms (DFT/IDFT) including cyclic prefix handling are offloaded to the FPGA. Furthermore, OAI's data interface to the Radio Frequency (RF) frontend is extended with Lib-IIO capabilities, which utilize an FPGA implementation of the underlying JESD204C protocol driver, fully alleviating the CPU from any communication overhead.

Proceedings of the 1st SPAICE Conference on AI in and for Space | Pages 308 - 312 DOI: 10.5281/zenodo.13885601

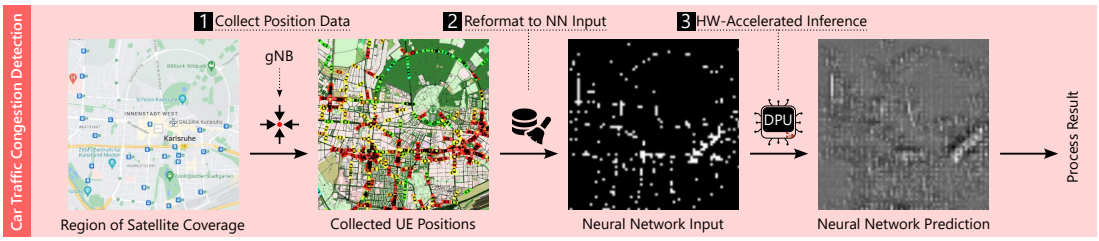


Figure 2: Visualization of the Proof-of-Concept AI car traffic congestion detection application dataflow.

These tasks are significantly more efficient and performant when executed on the PL section. Notably, the building blocks remain configurable via a bus interface from the ARM CPU, ensuring the flexibility of the gNodeB.

2.2.2 Hardware-Software Synchronization

Since the processing blocks on the FPGA have to be controlled differently compared to processing threads in software, a novel control and synchronization mechanism is introduced to the 5G Stack. This mechanism redirects the data flow from being written into Random Access Memory (RAM) to special buffers, which are directly accessible by the processing blocks on the FPGA via DMA. Additionally, a client thread manages the initiation and termination of processing on the PL and ensures synchronization between hardware processing and the calling thread through POSIX Inter-Process Communication (IPC) methods.

2.3 AI Application

As briefly mentioned in Sub-Sec. 2.1, the Versal AI Core series features an AI-Engine array which is a special hardware feature used to accelerate AI/ML inference. On 400 Very Long Instruction Word (VLIW)-Single Instruction Multiple Data (SIMD)-like vector processors, typical ML operations, such as matrix multiplication and convolution, can be performed in a massively data-parallel manner, reaching a performance of up to 100 Int-8 TOPS. The manufacturer's Intellectual Property (IP) core Deep Learning Processor Unit accelerates ML inference on generic CNN- and Dense-Layer-based model architectures by utilizing both AI-engine array and PL, as indicated by the red-shaded region in Fig. 1.

To demonstrate the simultaneous operation of gNodeB and AI Accelerator on the same platform we deploy a Proof-of-Concept AI Application in the Application Layer with the assumption of using metadata from the gNodeB as its input to solve a given task. Here, we decided for a car traffic congestion detection algorithm by using the position of connected (assumed vehicular) users. Fig. 2 visualizes the data-flow starting with position information collected from the gNodeB, reformatting the data, and finally processing it with a neural network (NN).

3 Results

3.1 Partial Physical Layer Offloading

In this work, the reduction in CPU workload is used as primary metric to measure the effectiveness of hardware acceleration. Three Physical layer configurations[2] with varying computational loads as summarized in Tab. 1 were implemented. Table 2 summarizes the utilized hardware resources for CPU, FPGA, and RAM for each configuration in HW-accelerated and non-accelerated mode.

A significant reduction in CPU load is evident for Config-1 and Config-2. On average, the CPU load is reduced by about 25 %. For the largest configuration (Config-3), no significant CPU reduction of is observed, which can be attributed to the saturation of the CPU *both* in the non- and HW-accelerated case. This indicates that further offloading of other processing steps is required to free up CPU resources. The memory consumption remains unchanged independent of offloading, since only the processing domain changes. Overall, the required PL hardware resources are below 5 %.

3.2 AI Accelerator Performance and Resource Utilization

In this sub-section we evaluate the achieved NN inference performance and resource utilization. The U-Net-based NN [11] with a relatively small input resolution

[2]The Physical layer test configurations can also be found at `https://gitlab.eurecom.fr/oai/openairinterface5g` in the leo-5g-ntn branch under gnb.sa.band66.fr1.25PRB.usrpx300.conf, gnb.sa.band78.fr1.24PRB.usrpb210.conf, and gnb.band78.tm1.106PRB.usrpx300.conf, respectively.

Proceedings of the 1st SPAICE Conference on AI in and for Space | Pages 308 - 312 DOI: 10.5281/zenodo.13885601

PHY Config	Resource Blocks [#]	Sub-Carriers [# Total]	Sub-Carrier Spacing [kHz]	(I)FFT Size [#]	Sample Rate [MSamp/s]	Band [Nr.]	RF Head
Config-1	25	300	15	512	7.68	66	RF-SIM
Config-2	24	288	30	512	15.36	78	RF-SIM
Config-3	106	1.272	30	2048	61.44	78	RF-SIM

Table 1: Key properties of the implemented Physical layer configurations.

PHY Config	HW Acc.	FPGA LUT	FF	BRAM	URAM	DSP	CPU [% total]	RAM [%]
Config-1	no	0	0	0	0	0	73 %	9.3 %
	yes	1.7 %	1.6 %	4.7 %	0	3.2 %	53 %	9.3 %
Config-2	no	0	0	0	0	0	75 %	9.6 %
	yes	1.7 %	1.6 %	4.7 %	0	3.2 %	55 %	9.6 %
Config-3	no	0	0	0	0	0	93 %	16.8 %
	yes	1.7 %	1.6 %	4.7 %	0	3.2 %	91 %	16.8 %
Resources Available	N.A.	899.000	1.799.000	34 MBit	130 MBit	1968	100 %	8 GB

Table 2: Summary of utilized hardware resources (CPU, FPGA, RAM) for each PHY configuration in HW-accelerated and non HW-accelerated mode.

DPU Arch.	#Batch	#AI-E	FPGA	Time
C32B1	1	32	9 %	1.36 ms
C32B3	3	96	23 %	0.73 ms
C64B5	5	320	40 %	0.61 ms

Table 3: Resource Utilization and Performance of the AI Accelerator.

of 128x128x1 pixels is evaluated on three DPU configurations, i.e., the smallest (C32B1), medium-sized (C32B3), and largest (C64B5) configuration, realizing a trade-off between performance and required resources, which can be observed in Tab. 3. In summary, an inference time of < 1 ms per batch is achieved, demonstrating the platform's AI capabilities. For a more detailed performance analysis w.r.t. different network architectures and sizes, please refer to [12].

4 Discussion

In this work we successfully implemented OAI's gNodeB and AMD-Xilinx's AI accelerator on a shared, embedded platform. This is a significant achievement, since OAI's official system requirements (OS Ubuntu 22.04., 8 cores x86_64 @ 3.5 GHz, 32 GB RAM) prohibit deployment on a practical space-grade system. As shown in Tab. 2, we reduced the required resources by over an order of magnitude. By using only a fraction of the PL and no AI-Engines for the gNodeB, sufficient resources are available for the AI-accelerator. However, as shown in Sub-Sec. 3.1, our implementation has reached the limit for computationally heavy configurations, necessitating further optimizations:

Further work The primary step is to offload more PHY components, such as Coding, Scrambling, Modulation, Filtering, and more, to alleviate the CPU from most DSP-based processing burden. As a consequence of realizing multiple components on the PL, an efficient CPU-free inter-communication and data buffering strategy is to be implemented. Afterwards, the focus might be shifted on higher layers.

Once the gNodeB itself is sufficiently HW-accelerated, its connections to other base-stations and the 5G core network must be taken care of. While this will definitely require a hardware-accelerated communication protocol implementation, the current Ethernet-based NG- (gNodeB to core network) and XN- (inter-gNodeB) interfaces must be replaced with a space-to-ground (ground station) or inter-satellite (relaying to ground station) link, respectively. This inherently requires a dynamic position-aware routing algorithm to be designed.

Proceedings of the 1st SPAICE Conference on AI in and for Space | Pages 308 - 312 DOI: 10.5281/zenodo.13885601

References

1. Kulu, E. Satellite Constellations - 2021 Industry Survey and Trends. *35th Annual Small Satellite Conference* (2021).
2. Knopp, A. *et al. HEUMEGA – Independent trend analysis on megaconstellations* tech. rep. (German Aerospace Center (DLR), Cologne, Germany, June 2021). https://www.dlr.de/de/ar/medien/publikationen/broschueren/broschuere-heumega/HeumegaStudie.pdf/@@download/file.
3. Jackson, Donny. *Lynk claims successful test of satellite-to-cell-phone communications, cites potential public-safety value* ,in Urgent Communications, https://urgentcomm.com/2020/03/19/lynk-claims-successful-test-of-satellite-to-cell-phone-communications-cites-potential-public-safety-value/. 2020.
4. Laursen, L. No More "No Service": Cellphones will increasingly text via satellite. *IEEE Spectrum* **60,** 52–55 (2023).
5. SpaceX. *SpaceX sends first text messages via its newly launched Direct-To-Cell satellites* , available at https://api.starlink.com/public-files/DIRECT_TO_CELL_FIRST_TEXT_UPDATE.pdf. 2024.
6. Kumar, S. *et al. 5G-NTN GEO-based over-the-air demonstrator using OpenAirInterface* in *39th International Communications Satellite Systems Conference (ICSSC 2022)* **2022** (2022), 110–114.
7. 3GPP. *"3rd Generation Partnership Project; Technical Specification Group Radio Access Network; New SID on AI/ML for NR Air Interface"* tech. rep. RP-212708 (3GPP, 2021).
8. 3GPP. *"3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Study on Integrated Sensing and Communication (Release 19)"* tech. rep. TR 22.837 (3GPP, 2024).
9. Villa, D. *et al.* An Open, Programmable, Multi-vendor 5G O-RAN Testbed with NVIDIA ARC and OpenAirInterface. *Proc. of the 2nd IEEE Workshop on Next-generation Open and Programmable Radio Access Networks (NG-OPERA)* (2024).
10. DeepSig & Intel. White Paper: Amplifying 5G vRAN Performance with Artificial Intelligence & Deep Learning (2022).
11. Ronneberger, O., Fischer, P. & Brox, T. *U-Net: Convolutional Networks for Biomedical Image Segmentation* 2015. arXiv: 1505.04597 [cs.CV].
12. Petry, M. *et al. Accelerated Deep-Learning Inference on the Versal adaptive SoC in the Space Domain* in *2023 European Data Handling Data Processing Conference (EDHPC)* (2023), 1–8.

5.2.1 Improvement: Advanced Disaggregation of the 5G Stack on an Extended Processing Platform

To the best of our knowledge, [Pub5.A] introduces the first 5G gNodeB tailored specifically to the VERSAL platform. Hereby the presented concept applies the principle of hardware acceleration for both 5G physical layer signal processing as well as AI inference based on the analysis performed in [Pub3.A], pursuing two goals simultaneously. While the proof-of-concept implementation in [Pub5.A] demonstrates the setup's feasibility, it raises concerns with respect to scaling the system up to higher demands on the 5G processing side, caused by utilizing waveforms with higher bandwidths and enabling multi-user access. While the major focus of [Pub5.A] is on the compute capacity bottleneck due to the intensive workload of the physical layer, the higher layers in the 5G stack also warrant sufficient resources. Fig. 5.3 gives a peak on the underlying layering scheme inside the gNodeB. To understand their processing demands, it is appropriate to review their functionality:

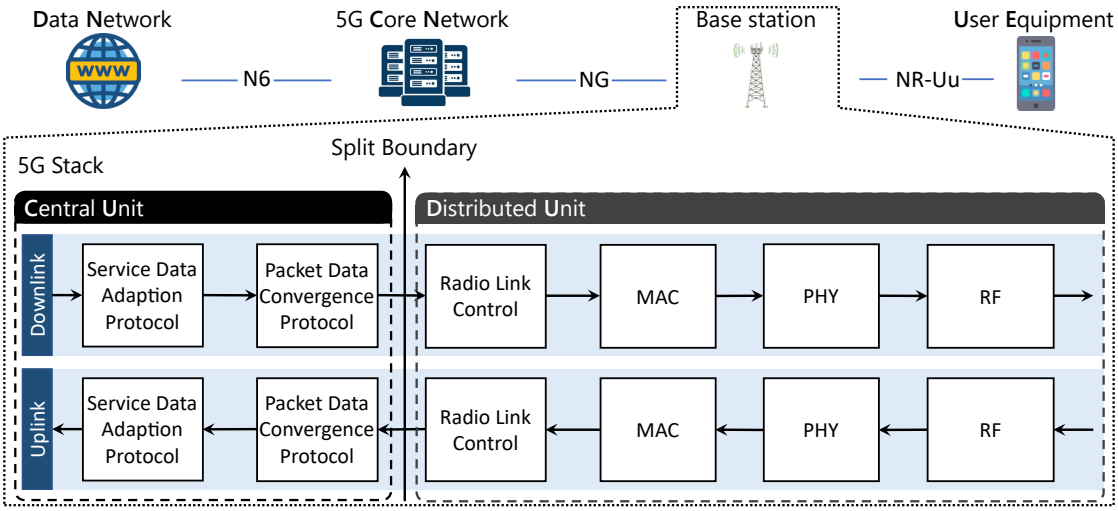


Figure 5.3: Overview of the processing layers inside a 5G gNodeB and their disaggregation into central unit and distributed unit (one option of many shown).

- MAC layer: Multiplexing between logical channels and transport channels, error correction through Hybrid Automatic Repeat Request (HARQ), priority handling of UEs by means of dynamic scheduling, random access procedures, beam management, and more.
- Radio Link Control layer: Reordering of 5G-RLC Protocol Data Units, duplicate detection, protocol error detection, and segmentation.
- Packet Data Convergence Protocol layer: Ciphering and integrity protection, transfer of user and control plane data, reordering and in-order delivery, and duplicate

discarding.

- Service Data Adaption Protocol layer: Management of QoS flows across the 5G air interface.

From the task descriptions it becomes clear that these layers consist mainly of not trivially parallelizable operations but instead benefit from a general-purpose architecture, such as provided by standard CPUs. The VERSAL platform does contain only a comparably weak dual-core A72 ARM core which does not provide sufficient resources. From this perspective we argue that the payload would benefit from the integration of general-purpose processing capabilities in the form of a multi-core CPU [197].

This raises the challenge of efficiently splitting the workload between the parallel VERSAL platform and the general-purpose CPU platform. The standardized solution to this problem is called Radio Access Network (RAN) disaggregation. It allows to split up the gNodeB in a central component, called CU, and a distributed component, called DU, as indicated by the two boxes in Fig. 5.3. Multiple positions in the 5G stack where the split boundary is positioned have been investigated by 3GPP [198–200]. A good overview is provided by [201].

When considering the hardware platforms at hand, a split between the lower and higher layers, i.e., between the physical layer and the MAC layer, is most effective. Based on 3GPP terminology, this corresponds to split option 6 (MAC-PHY split) [200]. Fig. 5.4 visualizes the design partitioning. The red-shaded region contains the components deployed on the VERSAL, i.e., the complete physical layer, and the green-shaded region contains the components deployed on the multi-core CPU, i.e., the rest of the 5G stack. The figure also denotes the interfaces between the components, which is Peripheral Component Interconnect Express (PCIe) for the communication between both devices.

Furthermore, the improved design ensures efficient connectivity of a dedicated space-grade RF frontend by routing the data directly to/from its end point on the VERSAL by utilizing an FPGA-accelerated implementation of the JESD204C protocol.

In conclusion, the revised architecture picks up one of the main open points revealed in [Pub5.A] by addressing the scalability of the concept. However, what has not yet been thoroughly discussed are the platform's AI capabilities and its use options with respect to 5G base station processing, such as the achievable integration level for physical layer signal processing (ref. to Sub-sec. 4.1.1), or possible applications in the higher layers. The next section elaborates on the current directions regarding the integration of AI capabilities on the base station side in the standard and assesses the future-proofness of the proposed approach.

5.3 Discussion on Future-Proofing to 5G-Advanced and 6G AI Requirements

The integration of AI/ML into 5G-Advanced (Rel. 18 - Rel. 20) marks a pioneering stride within 3GPP and lays the groundwork for potential trajectories in 6G [202]. In this section we look at the on-going standardization effort on the integration of AI into

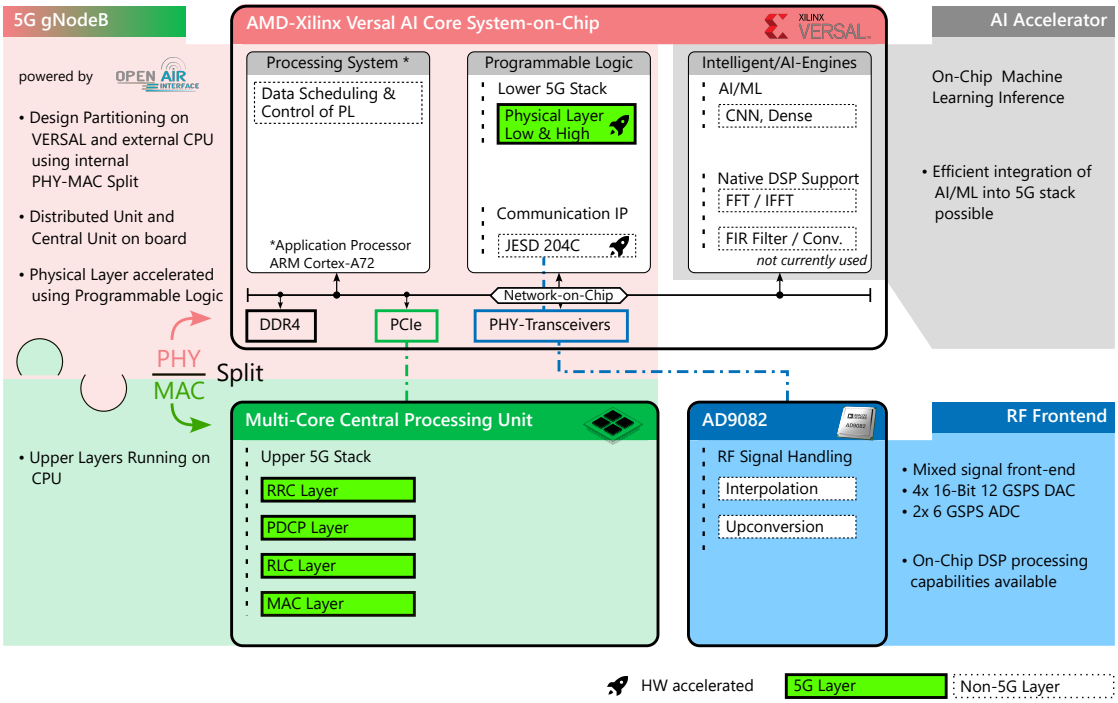


Figure 5.4: System design of a full 5G base station (gNodeB) including central and distributed unit realized on a heterogeneous processing platform. 5G layer processing (green-shaded boxes) is distributed between VERSAL (physical layer) and multi-core CPU (higher 5G layers), realizing a PHY-MAC processing spit. The AD9082 mixed signal front-end is connected to the VERSAL via the JESD204B/C procotol. AI inference capacity remains available trough the VERSAL's AI-Engines. Reproduced from [197].

the network from a conceptual perspective by focusing on the envisioned AI/ML model Lifecycle Management (LCM), deployment schemes, and base station-UE collaboration. This is followed by a feasibility analysis for supporting the key AI/ML functionality defined by 3GPP on the above presented hardware platform.

Within 3GPP, the applications of AI currently studied can be separated into the following four categories [203]:

1. AI/ML for the New Radio (NR) air interface [204]: To fully leverage AI/ML via cross-node collaboration on network- and UE-side in the air interface, including physical layer aspects, protocol aspects, and interoperability and testing aspects, three distinct use cases, i.e., channel state information feedback enhancement, beam management, and positioning accuracy enhancements are studied. The envisioned AI LCM is discussed in detail below.

2. AI/ML for mobility in NR [205]: The study targets two goals by means of AI: First,

to reduce the measurement efforts in the temporal, spatial, and frequency domain by using predictive measurements, and secondly, to perform mobility robustness optimization by primarily increasing handover performance, i.e., reducing ping-pong handovers, handover interruptions, handover failures, radio link failures, and short times of stay.

3. AI/ML for the next generation Radio Access Network (RAN) [206]: The RAN, which comprises all hardware and software components responsible for connecting individual devices to a network through a radio link, is currently studied for the use cases energy saving by recommending the activation or deactivation of cells and predicting energy efficiency, load balancing by predicting own and neighboring NG-RAN nodes' resource status, and mobility optimization by applying trajectory prediction to predict handover target nodes and resource reservation time windows. Future topics of interests are network slicing and coverage and capacity optimization.

4. AI/ML for management [207]: In this category the focus is on extending the Management Data Analytics (MDA) function with AI, which is considered a foundational capability for mobile networks to process and analyze data related to network and service events, such as performance measurements, KPIs, Quality-of-Experience, and more.

Interestingly, all of the above mentioned categories develop their own individual AI/ML model LCM flavors by individually defining data collection points, model training and inference locations, actor components that consume the results, and data paths to facilitate feedback for model updates. Depending on the category, additional aspects, such as data and model specifications, interfaces, cross-vendor interoperability, security, and trustworthiness are addressed as needed. In the following we take a look at the AI/ML model LCM in the air interface (category 1) since its different deployment schemes allow conclusions towards the future-proofness of our AI-enabled gNodeB concept.

In the NR air interface, the LCM of AI/ML models includes the following aspects: Data collection, model training, functionality/model identification, model delivery and transfer, model inference operation, functionality/model selection (including (de-)activation, switching, and fallback operation), monitoring, model update, and UE capability [204]. The essential functionality of this LCM and how these functions are interconnected is depicted in Fig. 5.5.

One highly interesting aspect are the collaboration levels between the network, specifically the gNodeB and the UE. The following levels have been found pertinent for the three selected use cases in this category [202, 204, 208]:

- Level x: No collaboration. AI/ML functionalities are implemented independently on either the UE or network side, using proprietary techniques. No changes to the air interface are required to support these operations, as there is no coordination or exchange of information related to AI/ML operations between the two sides.

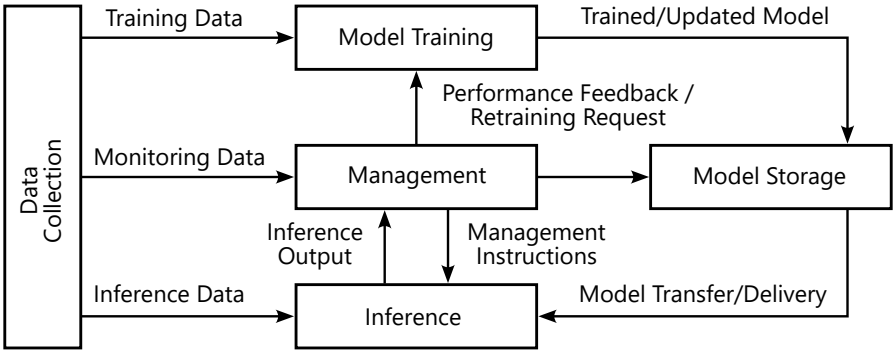


Figure 5.5: Lifecycle Management for Artificial Intelligence/Machine Learning models as defined by the 3rd Generation Partnership Project for the New Radio air interface. Reproduced with adaptions from [204].

- Level y: Signaling-based collaboration without model transfer. AI/ML processes rely on dedicated signaling introduced through modifications to the air interface. While collaboration between the UE and network is enabled through signaling, the transfer of AI/ML models over the air interface is not involved. This approach uses standardized enhancements to support AI/ML without requiring model exchanges.
- Level z: Signaling-based collaboration with model transfer. On this level, both advanced signaling enhancements and the capability to transfer AI/ML models over the air interface are integrated. The transfer allows the UE and network to exchange AI/ML models, enabling deeper collaboration and enhanced functionality through standardized mechanisms.

We now go individually through each functionality of the NR air interface AI/ML LCM and evaluate its practicality for implementing it on the presented hardware platform, and what potential challenges could arise with respect to the different integration levels, and how possible mitigation efforts could look like.

Data collection is the primary functionality on which model training, monitoring tasks, and model inference rely on. It can operate without or with latency constraints, for instance, for offline model training or for real-time performance monitoring, respectively. For standardized AI use cases 3GPP is expected to define interfaces for data collection and signaling, hence, access to the data is ensured. For proprietary use cases it could be required to implement interfaces manually, which is not expected to pose a problem in our flexible architecture.

Model inference is envisioned to be supported for both static (local) models, and models delivered over the air interface (ref. to Level z above). In our architecture as depicted in [Pub5.A] and Fig. 5.4, model inference is performed on the VERSAL. As described in [Pub3.A], the model's operations must be compatible with the feature set of the accelerator IP and it must undergo multiple transformations, including parameter quantization and compilation, prior to being run. Depending on the model size this can

require significant resources (computational and memory) and time. While for static models this can be done in advance, the preparation of recently transferred AI/ML models can only be done on the VERSAL's CPU or the multi-core CPU. Although this is possible in principle, it will negatively impact the time-span until the model is available for hardware-accelerated inference.

Model training raises similar issues. Although models can be trained on either of the CPUs in our proposed architecture, succinct post-processing as described above must be performed before inference can be run on the received / updated model. Furthermore, processing capabilities for model training are limited and must be shared with 5G processing. Additionally, overhead for transferring the training data, the raw model, and the compiled model within the architecture must be considered. In summary, although model training and the succinct post-processing operations are possible, the system is not optimized for this use-case and practical limitations to model size and training time are to be expected. This can severely limit Level z integration which might require real-time model training and inference upon delivery.

The management function oversees the operation, i.e., selection, (de-)activation, switching, and fallback of AI/ML functionalities and collects the inference output of the models. Based on the inference output and monitoring data it determines performance feedback and can issue retraining requests. It also issues model transfer and delivery requests to the model storage function. The nature of these operations warrants general-purpose processing capabilities, ideally being tightly connected with the AI inference engine. The VERSAL's embedded dual-core CPU satisfies those requirements. A potential minor challenge can be the additional communication effort between the 5G stack's higher layers, executed on the multi-core CPU, and the VERSAL for facilitating information transfer.

The last defined feature is model storage, which provides permanent and volatile model storage capabilities. As is a general issue in space, storage volume, especially of non-volatile nature, is highly valuable and severely limited due to additional precautions to protect of radiation-induced faults. While the proposed architecture primarily focuses on the definition of processing capabilities and distributing workload, we want to stress that additional storage capabilities for AI/ML models must be considered in future implementations.

Overall, we can conclude that the proposed platform concept supports the key functionality of the LCM of AI/ML functionality envisioned by 3GPP to a large extend, indicating general compatibility with the future demands of AI technology embedded in the gNodeB in space. However, the performance for complex use cases that involve in-orbit model fine-tuning or training and real-time inference of dynamically delivered AI/ML models cannot match terrestrial technology due to the lack of large-scale model training capabilities on embedded architectures compared to GPUs on earth. For these reasons, we encourage standardizing bodies to integrate AI processing demands in a light-weight manner into NTNs, but take the different processing capabilities between space- and earth-bound technologies into account.

5.4 Summary

In this chapter we took a look beyond the horizon and explored the growing demand for AI technology on-board spacecrafts from the perspective of NTNs introduced with 5G/6G technology. We see NTNs and the accompanying requirement for spacecrafts to be cell-towers in space as a key driver for bringing AI functionality, whose native integration into the 5G stack is currently under heavy investigation, into space. After introducing the transparent and regenerative network architectures, we proposed a payload design optimized for on-board gNodeB functionality in [Pub5.A] that shares resources on the same platform with an AI inference engine and implemented a proof-of-concept use case that demonstrates the synergy of this approach. A succinct optimization picks up shortcomings regarding processing capabilities for 5G's higher layers by expanding the platform with a multi-core CPU. Finally, we studied the future-proofness of this optimized design with respect to the current state of standardization efforts of AI/ML functionalities and their envisioned life cycle by 3GPP for the upcoming 5G/6G releases.

6 Conclusion

In this dissertation, we have investigated how the paradigm shift in the field of communications fueled by Artificial Intelligence (AI) will impact the operation and design of future telecommunication satellite systems. By operating in a special setting at the intersection between industry and research, we motivated this topic from different perspectives. While, on the one hand, the satellite payload manufacturer is primarily concerned with the question «How do we make communication payloads fit for the future?», standardization committees, on the other hand, ask «What AI/ML capabilities and functionality can we expect and demand from next-generation telecommunication satellites?», for instance in the frame of on-going standardization efforts within 3GPP's Non-Terrestrial Network (NTN) for 5G/6G. These questions in turn motivate a number of research topics which we have addressed using the following three-step methodology.

After identifying that these questions are fundamentally grounded on the hardware capabilities that enable processing of AI/ML workloads in space, our first step was marked by an investigation of AI-enabled hardware technologies for space. With the biggest discrepancy to terrestrial solutions grounded in Size, Weight, and Power (SWaP), withstanding the harsh space environment (specifically radiation), and limitations in AI/ML capabilities, such as model architecture and size restrictions, we performed an in-depth analysis of available hardware technologies and related products, providing clarity on contemporary capabilities. This led to the conslusion that the VERSAL AI CORE adaptive System-on-Chip (SoC) platform is the best-fit for future AI-heavy payloads by delivering a game-changing amount of user-controlled AI-native processing resources in a space-grade format. Furthermore, by joining forces with reconfigurable FPGAs and CPUs using the SoC concept, the technology paves the way for the paradigm shift from transparent to regenerative satellite architectures.

In the second step, we investigated the feasibility of applying AI to communication systems on the resource-constrained platforms as selected above. We did this in two ways, following an AI-native and an AI-augmented approach. In the AI-native approach we successfully realized a functioning unidirectional communication system based on a bit-wise Autoencoder (AE) that solely relies on operations performed within Multilayer Perceptrons and Convolutional Neural Networks, the two key model architectures natively supported on the VERSAL. In addition to being trained End-to-End (E2E) from modulator to demodulator, the concept of a Neural Network (NN)-defined physical layer inherently enables a high level of flexibility for fully replacing or upgrading the physical layer by simply switching out the NNs or only its weights, respectively. By bringing this concept to a non-GPU-based hardware platform, we demonstrate principle feasibility and functionality of this approach on realistic hardware and lay the first steps on supporting dynamic AI/ML inference functionality based on model transfers during

run-time.

However, we found that the major downside of the AI-native approach is limited scalability, both to more capable models and higher throughput, due to the limited amount of resources for AI hardware acceleration. For this reason and due to a lack in model performance, we transitioned to the AI-augmented approach by selectively augmenting expert domain knowledge in the form of classical signal processing pipelines with Deep Learning (DL)-based processing components. We chose to specifically address Radio Frequency (RF) signal synchronization procedures with DL due to its high relevance for satellite communications (mobility, Doppler effect) and for being generally underinvestigated in the field of communications. We addressed signal perturbations, such as Carrier Frequency Offset, Constant Phase Offset, and Sampling Time Offset, by adopting the Radio Transformer Network concept in a hybrid signal processing chain that utilizes DL-based components for estimating key characteristics of the aforementioned effects, followed by classic algorithms to perform the respective compensating transformations. With this approach we could address both shortcomings of the AI-native approach. We beat a classic baseline in Bit Error Rate (BER) performance for unsynchronized signals due to complementing existing domain knowledge with the power of E2E-optimized ML, and significantly reduced the overall workload on the AI hardware accelerator, enabling to scale the concept to higher demands.

The price paid for the aforementioned benefits is an increased system complexity when it comes to practical hardware deployment. Implementing such hybrid signal processing chains efficiently requires far more effort than simply deploying one AI-model for AI-native solutions. We developed a custom-tailored deployment strategy for the AI-augmented RF signal synchronization algorithm by splitting up the complete workload in parts, distributing them to their respective compute domain (FPGA for classic algorithms, AI-accelerator for DL), organizing the data flow between the different components in an efficient way using a zero-copy buffering strategy, and writing custom software for controlling the whole inference process. Therefore we demonstrated the feasibility of realizing more advanced processing schemes, which are expected to come in such hybrid forms, on future generation regenerative satellite platforms.

Lastly, we have evaluated the gained insights on future AI capabilities of telecommunication satellites in a bigger picture by contrasting them with the AI/ML-capabilities and features envisioned for future releases of 5G and 6G, which are highly relevant due to their applicability to the concept of NTNs. After setting a foundation by proposing and implementing an AI-enabled prototype of an on-board 5G base station (gNodeB), we evaluated its compatibility to the AI-related operations defined in 3GPP's AI Lifecycle Management (LCM) for the 5G air interface. We came to the conclusion, that key functionality is to a large extent supported, confirming the principle future-proofness of the hardware platform and the developed concept. However, we also highlighted open points, specifically regarding the highest AI integration level that requires real-time training and inference of models transferred over the network, that warrant modifications in the standard or additional efforts on the satellite payload side in the future.

Bringing Artificial Intelligence to satellites in space is no longer just a vision but an active field of development, pursued by government agencies and industries worldwide. As

global connectivity continues to expand, and as modern society becomes increasingly dependent on seamless communication, satellites - particularly those with high-throughput communication capabilities - are set to play a key role in this evolution. At the same time, AI has already demonstrated its potential in communications, making its adoption in space not a question of if, but when. This dissertation was written with the goal of building a bridge between the highly dynamic research side and the practical challenges of real-world implementations. We hope to have taken a step toward turning AI-powered satellite communications into a reality.

Bibliography

[1] F. Rosenblatt, "The Perceptron—a perceiving and recognizing automaton," Cornell Aeronautical Laboratory, Tech. Rep. 85-460-1, 1957. [Online]. Available: https://techreport.ch/_downloads/rosenblatt1957perceptron.pdf

[2] T. Haigh, "Between the Booms: AI in Winter," *Commun. ACM*, vol. 67, no. 11, p. 18–23, Oct. 2024. [Online]. Available: https://doi.org/10.1145/3688379

[3] S. V. Mahadevkar, B. Khemani, S. Patil, K. Kotecha, D. R. Vora, A. Abraham, and L. A. Gabralla, "A Review on Machine Learning Styles in Computer Vision—Techniques and Future Directions," *IEEE Access*, vol. 10, pp. 107 293–107 329, 2022.

[4] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds. Online: Association for Computational Linguistics, Jul. 2020, pp. 7871–7880. [Online]. Available: https://aclanthology.org/2020.acl-main.703/

[5] H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes, and A. Mian, "A Comprehensive Overview of Large Language Models," 2024. [Online]. Available: https://arxiv.org/abs/2307.06435

[6] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang, W. Ye, Y. Zhang, Y. Chang, P. S. Yu, Q. Yang, and X. Xie, "A Survey on Evaluation of Large Language Models," *ACM Trans. Intell. Syst. Technol.*, vol. 15, no. 3, Mar. 2024. [Online]. Available: https://doi.org/10.1145/3641289

[7] A. Esteva, K. Chou, S. Yeung, N. Naik, A. Madani, A. Mottaghi, Y. Liu, E. Topol, J. Dean, and R. Socher, "Deep learning-enabled medical computer vision," *NPJ digital medicine*, vol. 4, no. 1, p. 5, 2021.

[8] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis, "Highly accurate protein structure prediction with AlphaFold," *Nature*, vol. 596, no. 7873, pp. 583–589, 2021. [Online]. Available: https://doi.org/10.1038/s41586-021-03819-2

[9] B. Schmauch, A. Romagnoni, E. Pronier, C. Saillard, P. Maillé, J. Calderaro, A. Kamoun, M. Sefta, S. Toldo, M. Zaslavskiy, T. Clozel, M. Moarii, P. Courtiol, and G. Wainrib, "A deep learning model to predict RNA-Seq expression of tumours from whole slide images," *Nature Communications*, vol. 11, no. 1, p. 3877, 2020. [Online]. Available: https://doi.org/10.1038/s41467-020-17678-4

[10] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[11] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nature Reviews Physics*, vol. 3, no. 6, pp. 422–440, 2021. [Online]. Available: https://doi.org/10.1038/s42254-021-00314-5

[12] D. Hong, B. Zhang, H. Li, Y. Li, J. Yao, C. Li, M. Werner, J. Chanussot, A. Zipf, and X. X. Zhu, "Cross-city matters: A multimodal remote sensing benchmark dataset for cross-city semantic segmentation using high-resolution domain adaptation networks," *Remote Sensing of Environment*, vol. 299, p. 113856, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0034425723004078

[13] H. Li, Z. Yuan, G. Dax, G. Kong, H. Fan, A. Zipf, and M. Werner, "Semi-Supervised Learning from Street-View Images and OpenStreetMap for Automatic Building Height Estimation," in *12th International Conference on Geographic Information Science (GIScience 2023)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), R. Beecham, J. A. Long, D. Smith, Q. Zhao, and S. Wise, Eds., vol. 277. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, pp. 7:1–7:15. [Online]. Available: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.GIScience.2023.7

[14] F. Meng, T. Ren, Z. Liu, and Z. Zhong, "Toward earthquake early warning: A convolutional neural network for rapid earthquake magnitude estimation," *Artificial Intelligence in Geosciences*, vol. 4, pp. 39–46, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666544123000187

[15] G. Furano, A. Tavoularis, and M. Rovatti, "AI in space: applications examples and challenges," in *2020 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2020, pp. 1–6.

[16] L. Ostrogovich, R. Del Prete, G. Tomasicchio, N. Longépé, and A. Renga, "A Dual-Mode Approach for Vision-Based Navigation in a Lunar Landing Scenario," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2024, pp. 6799–6808.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6000–6010.

[18] C. O. Dumitru, G. Schwarz, F. Castel, J. Lorenzo, and M. Datcu, "Artificial intelligence data science methodology for Earth Observation," in *Advanced Analytics and Artificial Intelligence Applications*. InTech Publishing London, UK, 2019, pp. 1–20.

[19] B. Zhang, Y. Wu, B. Zhao, J. Chanussot, D. Hong, J. Yao, and L. Gao, "Progress and Challenges in Intelligent Remote Sensing Satellite Systems," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 15, pp. 1814–1822, 2022.

[20] J. Murphy, J. E. Ward, and B. Mac Namee, "An Overview of Machine Learning Techniques for Onboard Anomaly Detection in Satellite Telemetry*," in *2023 European Data Handling & Data Processing Conference (EDHPC)*, 2023, pp. 1–6.

[21] B. Al Homssi, K. Dakic, K. Wang, T. Alpcan, B. Allen, R. Boyce, S. Kandeepan, A. Al-Hourani, and W. Saad, "Artificial Intelligence Techniques for Next-Generation Massive Satellite Networks," *Comm. Mag.*, vol. 62, no. 4, p. 66–72, Apr. 2024. [Online]. Available: https://doi.org/10.1109/MCOM.004.2300277

[22] F. Fourati and M.-S. Alouini, "Artificial intelligence for satellite communication: A review," *Intelligent and Converged Networks*, vol. 2, no. 3, pp. 213–243, 2021.

[23] S. Mahboob and L. Liu, "Revolutionizing Future Connectivity: A Contemporary Survey on AI-Empowered Satellite-Based Non-Terrestrial Networks in 6G," *IEEE Communications Surveys & Tutorials*, vol. 26, no. 2, pp. 1279–1321, 2024.

[24] T. O'Shea and J. Hoydis, "An Introduction to Deep Learning for the Physical Layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563–575, 2017.

[25] H. Hellström, J. M. B. da Silva Jr., M. M. Amiri, M. Chen, V. Fodor, H. V. Poor, and C. Fischione, "Wireless for Machine Learning: A Survey," *Foundations and Trends® in Signal Processing*, vol. 15, no. 4, pp. 290–399, 2022. [Online]. Available: http://dx.doi.org/10.1561/2000000114

[26] A. Khandelwal, T. Yun, N. V. Nayak, J. Merullo, S. H. Bach, C. Sun, and E. Pavlick, "$100K or 100 Days: Trade-offs when Pre-Training with Academic Resources," 2024. [Online]. Available: https://arxiv.org/pdf/2410.23261

[27] N. Ahmed, M. Wahed, and N. C. Thompson, "The growing influence of industry in AI research," *Science*, vol. 379, no. 6635, pp. 884–886, 2023. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.ade2420

[28] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P.-l. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. H. Yoon, "In-Datacenter Performance Analysis of a Tensor Processing Unit," *SIGARCH Comput. Archit. News*, vol. 45, no. 2, p. 1–12, Jun. 2017. [Online]. Available: https://doi.org/10.1145/3140659.3080246

[29] G. Lentaris, K. Maragos, I. Stratakos, L. Papadopoulos, O. Papanikolaou, D. Soudris, M. Lourakis, X. Zabulis, D. Gonzalez-Arjona, and G. Furano, "High-Performance Embedded Computing in Space: Evaluation of Platforms for Vision-Based Navigation," *Journal of Aerospace Information Systems*, vol. 15, no. 4, pp. 178–192, 2018.

[30] G. Giuffrida, L. Fanucci, G. Meoni, M. Batič, L. Buckley, A. Dunne, C. van Dijk, M. Esposito, J. Hefele, N. Vercruyssen, G. Furano, M. Pastena, and J. Aschbacher, "The PHI-Sat-1 Mission: The First On-Board Deep Neural Network Demonstrator for Satellite Earth Observation," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–14, 2022.

[31] European Space Agency, "ESA'S ANNUAL SPACE ENVIRONMENT REPORT," pp. 1–121, 2024. [Online]. Available: https://www.sdo.esoc.esa.int/environment_report/Space_Environment_Report_latest.pdf

[32] Space.com, Tereua Pultarova, "SpaceX Starlink satellites made 50,000 collision-avoidance maneuvers in the past 6 months. What does that mean for space safety?" https://www.space.com/spacex-starlink-50000-collision-avoidance-maneuvers-space-safety, [Online; accessed 22-November-2024].

[33] DeepSig Inc., "White Paper: A/I Wireless Signal Identification and Analysis Technical White Paper," https://www.deepsig.ai/download-wireless-signal-identification-and-analysis-white-paper/, [Online; accessed 27-January-2025].

[34] J.-s. Seo, J. Saikia, J. Meng, W. He, H.-s. Suh, Anupreetham, Y. Liao, A. Hasssan, and I. Yeo, "Digital Versus Analog Artificial Intelligence Accelerators: Advances, trends, and emerging designs," *IEEE Solid-State Circuits Magazine*, vol. 14, no. 3, pp. 65–79, 2022.

[35] Z. S. Gerard Maral, Michel Bousquet, *Satellite Communications Systems*. John Wiley & Sons, Ltd, 2020.

[36] M. Gardill, W. Kinsner, J. Budroweit, A. Al-Hourani, E. Dunkel, J. Swope, D. Evans, and M. Staebler, "Towards Space Edge Computing and Onboard AI for Real-Time Teleoperations," in *IEEE International Conference on Cognitive Informatics and Cognitive Computing*, June 2023. [Online]. Available: https://ai.jpl.nasa.gov/public/documents/papers/ieee-leo-sats-report.pdf

[37] F. Harris, "How to Synchronize Carrier Frequency, Carrier Phase, and Symbol Timing Loops in Modern Digital Modems; Slide deck," 2021, IEEE Military Communications Conference (MILCOM) 2021. [Online]. Available: https://milcom2021.milcom.org/program/tutorials.html#tut-09

[38] M. Petry, A. Koch, and M. Werner, "Envisioning Physical Layer Flexibility Through the Power of Machine-Learning," in *2023 IEEE Globecom Workshops (GC Wkshps)*, 2023, pp. 50–55.

[39] M. Petry, G. Wuwer, A. Koch, P. Gest, M. Ghiglione, and M. Werner, "Accelerated Deep-Learning Inference on the Versal adaptive SoC in the Space Domain," in *2023 European Data Handling & Data Processing Conference (EDHPC)*, 2023, pp. 1–8.

[40] M. Petry, B. Parlier, A. Koch, and M. Werner, "Auto-Regressive RF Synchronization Using Deep-Learning," in *2024 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN)*, 2024, pp. 145–150.

[41] M. Petry, A. Koch, and M. Werner, "Zero-Copy AI-Augmented Signal Processing Pipeline on Heterogeneous Satellite Processors," in *2024 58th Asilomar Conference on Signals, Systems, and Computers*, 2024, pp. 354–361.

[42] M. Petry and R. Mayr, "AI-enabled 5G base-station with Hardware Acceleration for Non-Terrestrial Networks on a Space-Grade System-on-Chip," in *Proceedings of the first joint European Space Agency SPAICE Conference / IAA Conference on AI in and for Space*. IAA, September 2024, pp. 308–312.

[43] H. Nyquist, "Certain factors affecting telegraph speed," *The Bell System Technical Journal*, vol. 3, no. 2, pp. 324–346, 1924.

[44] C. E. Shannon, "A Mathematical Theory of Communication," *The Bell System Technical Journal*, vol. 27, no. 2, pp. 379–423, 1948.

[45] M. S. De Alencar, *Appendix A Fourier and Hilbert Transforms*. River Publishers, 2018, pp. 205–232.

[46] J. Hoydis, S. Cammerer, F. A. Aoudia, A. Vem, N. Binder, G. Marcus, and A. Keller, "Sionna: An Open-Source Library for Next-Generation Physical Layer Research," 2023. [Online]. Available: https://arxiv.org/abs/2203.11854

[47] S. Wang and G. Y. Li, "Machine Learning in Communications: A Road to Intelligent Transmission and Processing," *Communications of Huawei Research*, vol. 7, pp. 2–20, 2024.

[48] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge, UK: Cambridge University Press, 2005. [Online]. Available: https://doi.org/10.1017/CBO9780511807213

[49] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Hoboken, NJ, USA: Wiley-Interscience, 2006. [Online]. Available: https://onlinelibrary.wiley.com/doi/book/10.1002/047174882X

[50] G. L. Stüber, *Principles of Mobile Communication*, 4th ed. Cham, Switzerland: Springer, 2017. [Online]. Available: https://link.springer.com/book/10.1007/978-3-319-55615-4

[51] ETSI, *Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Boradcasting, Interactive Services, News Gathering and other broadband satellite applicaitons; Part 2: DVB-S2 Extensions (DVB-S2X)*, ETSI Std. EN 302 307-2, 2014.

[52] G. Ungerboeck, "Channel coding with multilevel/phase signals," *IEEE Transactions on Information Theory*, vol. 28, no. 1, pp. 55–67, 1982.

[53] F. J. Harris, *Multirate Signal Processing for Communication Systems*, 2nd ed. River Publishers, 2021. [Online]. Available: https://doi.org/10.1201/9781003338888

[54] H. Meyr, M. Moeneclaey, and S. A. Fechtel, *Digital Communication Receivers: Synchronization, Channel Estimation, and Signal Processing*, 1st ed., ser. Wiley Series in Telecommunications and Signal Processing. New York, NY: John Wiley & Sons Ltd, 1997, vol. 2.

[55] F. Harris, *Let's Assume the System Is Synchronized*. Dordrecht: Springer Netherlands, 2011, pp. 311–325. [Online]. Available: https://doi.org/10.1007/978-94-007-0107-6_20

[56] ——, "Software-defined radio for the masses," 2022, Lecture slides of the University of California San Diego.

[57] F. Harris, C. Dick, and M. Rice, "Digital receivers and transmitters using polyphase filter banks for wireless communications," *IEEE Transactions on Microwave Theory and Techniques*, vol. 51, no. 4, pp. 1395–1412, 2003.

[58] J. G. Proakis, *Digital Communications*, 3rd ed. New York, NY: McGraw-Hill, 1995.

[59] F. Harris, E. Venosa, X. Chen, and C. Dick, "Band edge filters perform non data-aided carrier and timing synchronization of software defined radio QAM receivers," in *The 15th International Symposium on Wireless Personal Multimedia Communications*, 2012, pp. 271–275.

[60] P. Šimek, J. Škramlík, and V. Valouch, "A frequency locked loop strategy for synchronization of inverters used in distributed energy sources," *International Journal of Electrical Power & Energy Systems*, vol. 107, pp. 120–130, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0142061518306586

[61] F. M. Gardner, "Interpolation in Digital Modems – Part I: Fundamentals," *IEEE Transactions on Communications*, vol. 41, no. 3, pp. 501–507, 1993.

[62] K. Mueller and M. Muller, "Timing Recovery in Digital Synchronous Data Receivers," *IEEE Transactions on Communications*, vol. 24, no. 5, pp. 516–531, 1976.

[63] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[64] F. Fleuret, *The Little Book of Deep Learning*. self-published, 2024, [Online at https://fleuret.org/public/lbdl.pdf; accessed 19-December-2024].

[65] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 210–229, 1959.

[66] IBM, "What is Machine Learning?" https://web.archive.org/web/20231227153910/https://www.ibm.com/topics/machine-learning, [Online (archived); accessed 19-December-2024].

[67] S. Cammerer, *Dissertation: Trainable Communication Systems*. Universität Stuttgart, 2021.

[68] R. M. Schmidt, "Recurrent Neural Networks (RNNs): A gentle Introduction and Overview," 2019. [Online]. Available: https://arxiv.org/abs/1912.05911

[69] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 11 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[70] M. Beck, K. Pöppel, M. Spanring, A. Auer, O. Prudnikova, M. Kopp, G. Klambauer, J. Brandstetter, and S. Hochreiter, "xLSTM: Extended Long Short-Term Memory," 2024. [Online]. Available: https://arxiv.org/abs/2405.04517

[71] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes,

N. Lawrence, and K. Weinberger, Eds., vol. 27. Curran Associates, Inc., 2014. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf

[72] A. Baydin, B. Pearlmutter, A. Radul, and J. Siskind, "Automatic Differentiation in Machine Learning: A Survey," *Journal of Machine Learning Research*, vol. 18, no. 153, 2018.

[73] T. Sousa, "Dataflow Programming: Concept, Languages and Applications," Doctoral Symposium on Informatics Engineering, 01 2012.

[74] W. R. Sutherland, "PhD Thesis: The on-line graphical specification of computer procedures," http://hdl.handle.net/1721.1/13474, 1966.

[75] B. van Merrienboer, O. Breuleux, A. Bergeron, and P. Lamblin, "Automatic differentiation in ML: Where we are and where we should be going," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/file/770f8e448d07586afbf77bb59f698587-Paper.pdf

[76] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: a system for large-scale machine learning," in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'16. USA: USENIX Association, 2016, p. 265–283.

[77] F. Andrews, "The heritage of telegraphy," *IEEE Communications Magazine*, vol. 27, no. 8, pp. 12–18, 1989.

[78] S. Redl, M. W. Oliphant, M. K. Weber, and M. K. Weber, *An Introduction to GSM*, 1st ed. USA: Artech House, Inc., 1995.

[79] E. Zainea, A. Marţian, I. Marcu, and O. Fratu, "Transition from analog to digital broadcasting A spectral efficiency review," in *2012 10th International Symposium on Electronics and Telecommunications*, 2012, pp. 171–174.

[80] S. O'Leary, *Understanding Digital Terrestrial Broadcasting*. Artech House, 2000.

[81] H. Chen, R. Abbas, P. Cheng, M. Shirvanimoghaddam, W. Hardjawana, W. Bao, Y. Li, and B. Vucetic, "Ultra-Reliable Low Latency Cellular Networks: Use Cases, Challenges and Approaches," *IEEE Communications Magazine*, vol. 56, no. 12, pp. 119–125, 2018.

[82] P. Popovski, J. J. Nielsen, C. Stefanovic, E. d. Carvalho, E. Strom, K. F. Trillingsgaard, A.-S. Bana, D. M. Kim, R. Kotaba, J. Park, and R. B. Sorensen, "Wireless Access for Ultra-Reliable Low-Latency Communication: Principles and Building Blocks," *IEEE Network*, vol. 32, no. 2, pp. 16–23, 2018.

[83] N. Keshtiarast and M. Petrova, "ML Framework for Wireless MAC Protocol Design," in *2024 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN)*, 2024, pp. 560–565.

[84] N. Keshtiarast, O. Renaldi, and M. Petrova, "Wireless MAC Protocol Synthesis and Optimization With Multi-Agent Distributed Reinforcement Learning," *IEEE Networking Letters*, pp. 1–1, 2024.

[85] 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*, vol. 25, no. 1, pp. 154–160, 2018.

[86] T. Liu, M. Zhang, J. Zhu, R. Zheng, R. Liu, and Q. Wu, "ACCP: Adaptive Congestion Control Protocol in Named Data Networking Based on Deep Learning," *Neural Computing and Applications*, vol. 31, no. 9, pp. 4675–4683, sep 2019. [Online]. Available: https://doi.org/10.1007/s00521-018-3408-2

[87] H. Xie, Z. Qin, G. Y. Li, and B.-H. Juang, "Deep Learning Enabled Semantic Communication Systems," *IEEE Transactions on Signal Processing*, vol. 69, pp. 2663–2675, 2021.

[88] H. Ye, L. Liang, G. Y. Li, and B.-H. Juang, "Deep Learning-Based End-to-End Wireless Communication Systems With Conditional GANs as Unknown Channels," *IEEE Transactions on Wireless Communications*, vol. 19, no. 5, pp. 3133–3143, 2020.

[89] S. Dörner, M. Henninger, S. Cammerer, and S. ten Brink, "WGAN-based Autoencoder Training Over-the-air," in *2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2020, pp. 1–5.

[90] F. Euchner, J. Sanzi, M. Henninger, and S. Ten Brink, "GAN-based Massive MIMO Channel Model Trained on Measured Data," in *2024 27th International Workshop on Smart Antennas (WSA)*, 2024, pp. 109–116.

[91] M. Ebada, S. Cammerer, A. Elkelesh, and S. ten Brink, "Deep Learning-Based Polar Code Design," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2019, pp. 177–183.

[92] S. Cammerer, J. Hoydis, F. A. Aoudia, and A. Keller, "Graph Neural Networks for Channel Decoding," in *2022 IEEE Globecom Workshops (GC Wkshps)*, 2022, pp. 486–491.

[93] H. Huang, Y. Peng, J. Yang, W. Xia, and G. Gui, "Fast Beamforming Design via Deep Learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 1, pp. 1065–1069, 2020.

[94] F. Ait Aoudia and J. Hoydis, "End-to-End Learning for OFDM: From Neural Receivers to Pilotless Communication," *IEEE Transactions on Wireless Communications*, vol. 21, no. 2, pp. 1049–1063, 2022.

[95] M. B. Fischer, S. Dörner, F. Krieg, S. Cammerer, and S. t. Brink, "Adaptive NN-based OFDM Receivers: Computational Complexity vs. Achievable Performance," in *2022 56th Asilomar Conference on Signals, Systems, and Computers*, 2022, pp. 194–199.

[96] S. Dörner, S. Cammerer, J. Hoydis, and S. t. Brink, "Deep Learning Based Communication Over the Air," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 132–143, 2018.

[97] F. A. Aoudia, J. Hoydis, S. Cammerer, M. Van Keirsbilck, and A. Keller, "Deep Learning-Based Synchronization for Uplink NB-IoT," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, 2022, pp. 1478–1483.

[98] M. H. Jespersen, M. Pajovic, T. Koike-Akino, Y. Wang, P. Popovski, and P. V. Orlik, "Deep Learning for Synchronization and Channel Estimation in NB-IoT Random Access Channel," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–7.

[99] S. Dörner, J. Clausius, S. Cammerer, and S. t. Brink, "Learning Joint Detection, Equalization and Decoding for Short-Packet Communications," *IEEE Transactions on Communications*, vol. 71, no. 2, pp. 837–850, 2023.

[100] Y. Sun, H. Shen, B. Li, W. Xu, P. Zhu, N. Hu, and C. Zhao, "Trainable Joint Channel Estimation, Detection, and Decoding for MIMO URLLC Systems," *IEEE Transactions on Wireless Communications*, vol. 23, no. 9, pp. 12 172–12 188, 2024.

[101] F. Ait Aoudia and J. Hoydis, "Waveform Learning for Next-Generation Wireless Communication Systems," *IEEE Transactions on Communications*, vol. 70, no. 6, pp. 3804–3817, 2022.

[102] C. Studer, S. Medjkouh, E. Gonultaş, T. Goldstein, and O. Tirkkonen, "Channel Charting: Locating Users Within the Radio Environment Using Channel State Information," *IEEE Access*, vol. 6, pp. 47 682–47 698, 2018.

[103] F. Euchner and S. T. Brink, "ESPARGOS: Phase-Coherent WiFi CSI Datasets for Wireless Sensing Research," in *2024 Kleinheubach Conference*, 2024, pp. 1–4.

[104] P. Stephan, F. Euchner, and S. ten Brink, "Channel Charting-Based Channel Prediction on Real-World Distributed Massive MIMO CSI," 2024. [Online]. Available: https://arxiv.org/abs/2410.11486

[105] J. Hoydis, F. A. Aoudia, S. Cammerer, F. Euchner, M. Nimier-David, S. T. Brink, and A. Keller, "Learning Radio Environments by Differentiable Ray Tracing," *IEEE Transactions on Machine Learning in Communications and Networking*, vol. 2, pp. 1527–1539, 2024.

[106] C. Ruah, O. Simeone, J. Hoydis, and B. Al-Hashimi, "Calibrating Wireless Ray Tracing for Digital Twinning Using Local Phase Error Estimates," *IEEE Transactions on Machine Learning in Communications and Networking*, vol. 2, pp. 1193–1215, 2024.

[107] Y. C. Eldar, A. Goldsmith, D. Gündüz, and H. V. Poor, Eds., *Machine Learning and Wireless Communications*. Cambridge, United Kingdom: Cambridge University Press, 2022. [Online]. Available: https://doi.org/10.1017/9781108966559

[108] D. Gündüz, P. de Kerret, N. D. Sidiropoulos, D. Gesbert, C. R. Murthy, and M. van der Schaar, "Machine Learning in the Air," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2184–2199, 2019.

[109] Z. Qin, H. Ye, G. Y. Li, and B.-H. F. Juang, "Deep Learning in Physical Layer Communications," *IEEE Wireless Communications*, vol. 26, no. 2, pp. 93–99, 2019.

[110] Y. Sun, M. Peng, Y. Zhou, Y. Huang, and S. Mao, "Application of Machine Learning in Wireless Networks: Key Techniques and Open Issues," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3072–3108, 2019.

[111] S. Cammerer, F. A. Aoudia, J. Hoydis, A. Oeldemann, A. Roessler, T. Mayer, and A. Keller, "A Neural Receiver for 5G NR Multi-User MIMO," in *2023 IEEE Globecom Workshops (GC Wkshps)*, 2023, pp. 329–334.

[112] X. Lin, "Artificial intelligence in 3GPP 5G-Advanced: A survey," https://www.comsoc.org/publications/ctn/artificial-intelligence-3gpp-5g-advanced-survey, Aug 2023, [Online (archived); accessed 25-December-2024].

[113] X. Lin, L. Kundu, C. Dick, and S. Velayutham, "Embracing AI in 5G-Advanced Toward 6G: A Joint 3GPP and O-RAN Perspective," *IEEE Communications Standards Magazine*, vol. 7, no. 4, pp. 76–83, 2023.

[114] Technical Specification Group Radio Access Network, "Study on enhancement for Data Collection for NR and EN-DC," 3rd Generation Partnership Project, Tech. Rep. TR 37.817 V17.0.0 (2022-04), 2020.

[115] J. Hoydis, F. A. Aoudia, A. Valcarce, and H. Viswanathan, "Toward a 6G AI-Native Air Interface," *IEEE Communications Magazine*, vol. 59, no. 5, pp. 76–81, 2021.

[116] X. Lin, M. Chen, T. Yoo, Y. Wang, L. Bariah, N. H. Tran, and K. Huang, "The Interplay Between Generative AI and 5G-Advanced Toward 6G," *Netwrk. Mag. of Global Internetwkg.*, vol. 38, no. 5, p. 7–9, Sep. 2024. [Online]. Available: https://doi.org/10.1109/MNET.2024.3429692

[117] A. Celik and A. M. Eltawil, "At the Dawn of Generative AI Era: A Tutorial-cum-Survey on New Frontiers in 6G Wireless Intelligence," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 2433–2489, 2024.

[118] H. Zou, Q. Zhao, Y. Tian, L. Bariah, F. Bader, T. Lestable, and M. Debbah, "TelecomGPT: A Framework to Build Telecom-Specific Large Language Models," *arXiv preprint arXiv:2407.09424*, 2024, arXiv:2407.09424v1 [eess.SP]. [Online]. Available: https://arxiv.org/abs/2407.09424

[119] A. Maatouk, N. Piovesan, F. Ayed, A. D. Domenico, and M. Debbah, "Large Language Models for Telecom: Forthcoming Impact on the Industry," *IEEE Communications Magazine*, pp. 1–7, 2024.

[120] H. He, S. Jin, C.-K. Wen, F. Gao, G. Y. Li, and Z. Xu, "Model-Driven Deep Learning for Physical Layer Communications," *IEEE Wireless Communications*, vol. 26, no. 5, pp. 77–83, 2019.

[121] B. Thuraisingham, "Trustworthy Machine Learning," *IEEE Intelligent Systems*, vol. 37, no. 1, pp. 21–24, 2022.

[122] K. R. Varshney, *Trustworthy Machine Learning*. Chappaqua, NY, USA: Independently Published, 2022. [Online]. Available: https://www.trustworthymachinelearning.com/

[123] N. Islam and S. Shin, "Deep Learning in Physical Layer: Review on Data Driven End-to-End Communication Systems and Their Enabling Semantic Applications," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 4207–4240, 2024.

[124] K. Berahmand, F. Daneshfar, E. S. Salehi, Y. Li, and Y. Xu, "Autoencoders and their applications in machine learning: a survey," *Artificial Intelligence Review*, vol. 57, no. 2, p. 28, 2024. [Online]. Available: https://doi.org/10.1007/s10462-023-10662-6

[125] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," https://stanford.edu/~jlmcc/papers/PDP/Volume%201/Chap8_PDP86.pdf, Institute for Cognitive Science, University of California, San Diego, Technical Report, 1985, [Online (archived); accessed 30-December-2024].

[126] D. Bank, N. Koenigstein, and R. Giryes, *Autoencoders*. Springer International Publishing, Jan. 2023, pp. 353–374, publisher Copyright: © Springer Nature Switzerland AG 2023.

[127] Y. Yang, S. Mandt, and L. Theis, "An Introduction to Neural Data Compression," *Foundations and Trends® in Computer Graphics and Vision*, vol. 15, no. 2, pp. 113–200, 2023. [Online]. Available: http://dx.doi.org/10.1561/0600000107

[128] T. Rappaport, *Wireless communications: Principles and practice*, 2nd ed., ser. Prentice Hall communications engineering and emerging technologies series. Prentice Hall, 2002, includes bibliographical references and index.

[129] S. Cammerer, F. A. Aoudia, S. Dörner, M. Stark, J. Hoydis, and S. ten Brink, "Trainable Communication Systems: Concepts and Prototype," *IEEE Transactions on Communications*, vol. 68, no. 9, pp. 5489–5503, 2020.

[130] T. J. O'Shea, K. Karra, and T. C. Clancy, "Learning to communicate: Channel auto-encoders, domain specific regularizers, and attention," in *2016 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, 2016, pp. 223–228.

[131] S. Dörner, S. Rottacker, M. Gauger, and S. t. Brink, "Bit-wise Autoencoder for Multiple Antenna Systems," in *2021 17th International Symposium on Wireless Communication Systems (ISWCS)*, 2021, pp. 1–5.

[132] N. Samuel, T. Diskin, and A. Wiesel, "Learning to Detect," *IEEE Transactions on Signal Processing*, vol. 67, no. 10, pp. 2554–2564, 2019.

[133] Y. Zhang, H. Wu, and M. Coates, "On the Design of Channel Coding Autoencoders With Arbitrary Rates for ISI Channels," *IEEE Wireless Communications Letters*, vol. 11, no. 2, pp. 426–430, 2022.

[134] NEC, "Satellite Bus Subsystems," https://web.archive.org/web/20120905094500/http://www.nec.com/en/global/solutions/space/satellite_bus/index.html, [Online; archived 2012-09-05 at the Wayback Machine, NEC, accessed 03 January 2025.].

[135] "Boeing 702 Satellite Bus Image," https://www.boeing.com/space/boeing-satellites, [Online; accessed 03-January-2025].

[136] "Inmarsat-4 / EADS Eurostar 3000 Image," https://space.skyrocket.de/doc_sat/astrium_eurostar-3000.htm, [Online; accessed 03-January-2025].

[137] "ARROW Satellite Bus Image," https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRi7eJgEuULACujR9fm_ioG9acxC1zhoOExNg&s, [Online; accessed 03-January-2025].

[138] E. Klein-Lebbink, "Advanced satellite technology needs," in *2011 IEEE MTT-S International Microwave Symposium*, 2011, pp. 1–4.

[139] P. Inigo, O. Vidal, B. Roy, E. Alberty, N. Metzger, D. Galinier, J. Anzalchi, G. Huggins, and S. Stirland, "Review of terabit/s satellite, the next generation of HTS systems," in *2014 7th Advanced Satellite Multimedia Systems Conference and the 13th Signal Processing for Space Communications Workshop (ASMS/SPSC)*, 2014, pp. 318–322.

[140] H. Fenech, *High-Throughput Satellites*, ser. Satellite Communication Series. Norwood, MA: Artech House Publishers, 2021.

[141] "Eutelsat OneWeb," https://oneweb.net/, [Online; accessed 05-January-2025].

[142] A. Koch, A. Krstova, F. Hegwein, M. C. De Lera, F. Ales, M. Petry, R. Ali, M. Mallah, L. Hili, M. Ghiglione, and M. Werner, "On-Board Anomaly Detection on a Flight-Ready System," in *2023 European Data Handling & Data Processing Conference (EDHPC)*, 2023, pp. 1–4.

[143] A. Koch, M. Petry, and M. Werner, "AI-augmented Anomaly Detection Pipeline for Real-Time Onboard Processing of Spacecraft Telemetry," *Proceedings of the first joint European Space Agency SPAICE Conference / IAA Conference on AI in and for Space*, September 2024.

[144] M. Pignol, F. Malou, and C. Aicardi, *COTS in Space: Constraints, Limitations and Disruptive Capability*. Cham: Springer International Publishing, 2019, pp. 301–327. [Online]. Available: https://doi.org/10.1007/978-3-030-04660-6_12

[145] J. Budroweit and H. Patscheider, "Risk Assessment for the Use of COTS Devices in Space Systems under Consideration of Radiation Effects," *Electronics*, vol. 10, no. 9, 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/9/1008

[146] K. Technologies, "Klepsydra AI - Performance Benchmark," 2020, [Online; accessed 06-January-2025]. [Online]. Available: https://klepsydra.com/klepsydra-embedded-ai-dnn-inference-performance-benchmark-technical-report/

[147] ——, "Klepsydra AI and Frontgrade Gaisler Collaborate to Expand Microprocessing Versatility in Space Missions Through AI," 2024, [Online; accessed 06-January-2025]. [Online]. Available: https://klepsydra.com/klepsydra-ai-and-frontgrade-gaisler-collaborate-to-expand-microprocessing\-versatility-in-space-missions-through-ai/

[148] N. Corporation, "Nvidia Jetson AGS Orin Datasheet," https://www.nvidia.com/content/dam/en-zz/Solutions/gtcf21/jetson-orin/nvidia-jetson-agx-orin-technical-brief.pdf, 2022, [Online; accessed 06-January-2025].

[149] I. Rodriguez-Ferrandez, L. Kosmidis, M. M. Trompouki, D. Steenari, and F. J. Cazorla, "Evaluating the Computational Capabilities of Embedded Multicore and GPU Platforms for On-Board Image Processing," in *2023 European Data Handling & Data Processing Conference (EDHPC)*, 2023, pp. 1–7.

[150] I. Rodriguez-Ferrandez, M. Tali, L. Kosmidis, A. Costantino, and D. Steenari, "Exploring Total Ionizing Dose Radiation Effects Across Generations of NVIDIA Jetson Devices: A Comparative Analysis," in *2024 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2024, pp. 1–6.

[151] I. Troxel, U. Cindemir, D. Steenari, D. Sabogal, M. Tali, O. Flordal, M. Gruber, A. Costantino, and E. Brune, "Radiation Test Results Demonstrating Operate-through Capability of an ESA Application Using Software Mitigation on a COTS

GPU," in *2023 European Data Handling & Data Processing Conference (EDHPC)*, 2023, pp. 1–6.

[152] M. A. Felix, W. S. Slater, D. C. Landauer, R. E. Pinson, and B. B. Rutherford, "Total Ionizing Dose Radiation Testing of NVIDIA Jetson Orin NX System on Module," in *2024 IEEE Space Computing Conference (SCC)*, 2024, pp. 116–121.

[153] G. Benelli, G. Giuffrida, R. Ciardi, D. Davalle, G. Todaro, and L. Fanucci, "GPU at SAT, the AI enabling ecosystem for on-board satellite applications," in *2023 European Data Handling & Data Processing Conference (EDHPC)*, 2023, pp. 1–4.

[154] FastML Team, "fastmachinelearning/hls4ml," 2023. [Online]. Available: https://github.com/fastmachinelearning/hls4ml

[155] M. Blott, T. B. Preußer, N. J. Fraser, G. Gambardella, K. O'brien, Y. Umuroglu, M. Leeser, and K. Vissers, "FINN-R: An end-to-end deep-learning framework for fast exploration of quantized neural networks," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 11, no. 3, pp. 1–23, 2018.

[156] The MathWorks, Inc., "Custom IP Core Generation," https://www.mathworks.com/help/hdlcoder/ug/custom-ip-core-generation.html, 2022, [Online; accessed 28-June-2023].

[157] G. Todaro, M. Monopoli, G. Benelli, L. Zulberti, and T. Pacini, "Enhanced Soft GPU Architecture for FPGAs," in *2023 18th Conference on Ph.D Research in Microelectronics and Electronics (PRIME)*, 2023, pp. 177–180.

[158] L. M. V. Pereira, D. R. Melo, C. A. Zeferino, and E. A. Bezerra, "Analysis of LEON3 systems integration for a Network-on-Chip," in *2018 IEEE 19th Latin-American Test Symposium (LATS)*, 2018, pp. 1–3.

[159] Y. Sun and A. M. Kist, "Deep Learning on Edge TPUs," 2022. [Online]. Available: https://arxiv.org/abs/2108.13732

[160] H. T. LTD, "Hailo-8™ AI Processor Data Brief," https://hailo.ai/wp-content/uploads/2023/12/hailo8_product_brief_3.23.pdf, 2023, [Online; accessed 06-January-2025].

[161] J. Goodwill, G. Crum, J. MacKinnon, C. Brewer, M. Monaghan, T. Wise, and C. Wilson, "NASA SpaceCube Edge TPU SmallSat Card for Autonomous Operations and Onboard Science-Data Analysis," in *35th Annual AIAA/USU Conference on Small Satellites*. Logan, UT: NASA Goddard Space Flight Center, 2021.

[162] B. Rajendran, A. Sebastian, M. Schmuker, N. Srinivasa, and E. Eleftheriou, "Low-Power Neuromorphic Hardware for Signal Processing Applications: A review of architectural and system-level design approaches," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 97–110, Nov. 2019, publisher Copyright: © 1991-2012 IEEE.

[163] M. L. Alles, R. D. Schrimpf, R. A. Reed, L. W. Massengill, R. A. Weller, M. H. Mendenhall, D. R. Ball, K. M. Warren, T. D. Loveless, J. S. Kauppila, and B. D. Sierawski, "Radiation hardness of FDSOI and FinFET technologies," in *IEEE 2011 International SOI Conference*, 2011, pp. 1–2.

[164] M. Adhiwiyogo, R. D'Souza, S. Leibson, and R. Shak, "White Paper: Pushing AI Boundaries with Scalable Compute-Focused FPGAs," pp. 1–6, 2020.

[165] V. J. R. Batthula, R. S. Segall, D. Berleant, H. Aboudja, and P. Tsai, "Future Satellite Lifetime Prediction From the Historical Trend in Satellite Half-Lives," in *Proceedings of the 26th World Multi-Conference on Systemics, Cybernetics and Informatics: WMSCI 2022*, N. Callaos, E. Gaile-Sarkane, S. Hashimoto, and B. Sánchez, Eds., vol. II. International Institute of Informatics and Cybernetics, 2022, pp. 6–11. [Online]. Available: https://doi.org/10.54808/WMSCI2022.02.6

[166] R. Bekkerman, M. Bilenko, and J. Langford, *Scaling up Machine Learning: Parallel and Distributed Approaches*. USA: Cambridge University Press, 2011.

[167] N. Strom, "Scalable Distributed DNN Training Using Commodity GPU Cloud Computing," in *Proc. Interspeech 2015*, 2015, pp. 1488–1492.

[168] E. C. for Space Standardization, "ECSS-Q-HB-80-03A Rev.1 – Software dependability and safety," https://ecss.nl/hbstms/20347/, 2017, [Online; accessed 07-January-2025].

[169] NASA Technical Standards System, "NASA-STD-87398 - Software Assurance and Software Safety Standard," https://standards.nasa.gov/standard/NASA/NASA-STD-87398, 2022, [Online; accessed 07-January-2025].

[170] C. Qian, M. Zhang, Y. Nie, S. Lu, and H. Cao, "A Survey of Bit-Flip Attacks on Deep Neural Network and Corresponding Defense Methods," *Electronics*, vol. 12, no. 4, 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/4/853

[171] S. Hong, P. Frigo, Y. Kaya, C. Giuffrida, and T. Dumitras, "Terminal Brain Damage: Exposing the Graceless Degradation in Deep Neural Networks Under Hardware Fault Attacks," in *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 497–514. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/hong

[172] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y.-J. A. Zhang, "The Roadmap to 6G: AI Empowered Wireless Networks," *IEEE Communications Magazine*, vol. 57, no. 8, pp. 84–90, 2019.

[173] M. A. Jamshed, A. Kaushik, M. Dajer, A. Guidotti, F. Parzysz, E. Lagunas, M. D. Renzo, S. Chatzinotas, and O. A. Dobre, "Non-Terrestrial Networks for 6G: Integrated, Intelligent and Ubiquitous Connectivity," 2024. [Online]. Available: https://arxiv.org/abs/2407.02184

[174] J. Hoydis, F. A. Aoudia, A. Valcarce, and H. Viswanathan, "Toward a 6G AI-Native Air Interface," *IEEE Communications Magazine*, vol. 59, no. 5, pp. 76–81, 2021.

[175] E. Nachmani, Y. Beery, and D. Burshtein, "Learning to Decode Linear Codes Using Deep Learning," 2016. [Online]. Available: https://arxiv.org/abs/1607.04793

[176] J. Andrews, "10x the capacity with half the energy: Site-specific deep learning for 6G - Keynote Speach Slidedeck," https://www.asilomarsscconf.org/AndrewsPlenary_Asilomar2024.pdf, 2024, [Online; accessed 10-January-2025].

[177] T. J. O'Shea, L. Pemula, D. Batra, and T. C. Clancy, "Radio transformer networks: Attention models for learning to synchronize in wireless systems," in *2016 50th Asilomar Conference on Signals, Systems and Computers*, 2016, pp. 662–666.

[178] B. Parlier, "Autoregressive Deep Learning-Based Signal Synchronization," Master's thesis, RWTH Aachen University, Aachen, Germany, October 2023.

[179] T. Erpek, T. J. O'Shea, Y. E. Sagduyu, Y. Shi, and T. C. Clancy, *Deep Learning for Wireless Communications*. Cham: Springer International Publishing, 2020, pp. 223–266. [Online]. Available: https://doi.org/10.1007/978-3-030-31764-5_9

[180] J. Liu, K. Mei, X. Zhang, D. Ma, and J. Wei, "Online Extreme Learning Machine-Based Channel Estimation and Equalization for OFDM Systems," *IEEE Communications Letters*, vol. 23, no. 7, pp. 1276–1279, 2019.

[181] M. Goutay, F. A. Aoudia, J. Hoydis, and J.-M. Gorce, "Machine Learning for MU-MIMO Receive Processing in OFDM Systems," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 8, pp. 2318–2332, 2021.

[182] J. Ney, S. Dörner, M. Herrmann, M. Hassani Sadi, J. Clausius, S. ten Brink, and N. Wehn, "FPGA-based Trainable Autoencoder for Communication Systems," in *Proceedings of the 2022 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 154. [Online]. Available: https://doi.org/10.1145/3490422.3502337

[183] SmartSat, "Machine Learning Onboard Satellites," https://smartsatcrc.com/app/uploads/2021/11/Machine-Learning-Onboard-Satellites.pdf, SmartSat, Technical Report 010, 2021.

[184] M. J. Veyette, K. Aylor, D. Stafford, M. Herrera, S. Jumani, C. Lineberry, C. Macklen, E. Maxwell, R. Stiles, and M. Jenkins, "AI/ML for Mission Processing Onboard Satellites." [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2022-1472

[185] G. Mateo-Garcia, J. Veitch-Michaelis, C. Purcell, N. Longepe, S. Reid, A. Anlind, F. Bruhn, J. Parr, and P. P. Mathieu, "In-orbit demonstration

of a re-trainable machine learning payload for processing optical imagery," *Scientific Reports*, vol. 13, no. 1, p. 10391, 2023. [Online]. Available: https://doi.org/10.1038/s41598-023-34436-w

[186] K. Lange, F. Fontana, F. Rossi, M. Varile, and G. Apruzzese, " Machine Learning in Space: Surveying the Robustness of On-Board ML Models to Radiation ," in *2024 IEEE Space Computing Conference (SCC)*. Los Alamitos, CA, USA: IEEE Computer Society, Jul. 2024, pp. 51–64. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SCC61854.2024.00012

[187] C. Krittanawong, Ed., *Chapter 16: Current AI Technology in Space; Precision Medicine for Long and Safe Permanence of Humans in Space*, 1st ed. Elsevier Science, 2024.

[188] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep Learning with Limited Numerical Precision," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 1737–1746. [Online]. Available: https://proceedings.mlr.press/v37/gupta15.html

[189] K. Holstein, J. Wortman Vaughan, H. Daumé, M. Dudik, and H. Wallach, "Improving Fairness in Machine Learning Systems: What Do Industry Practitioners Need?" in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, ser. CHI '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1–16. [Online]. Available: https://doi.org/10.1145/3290605.3300830

[190] J. Downey, B. Hilburn, T. O'Shea, and N. West, "Machine learning remakes radio," *IEEE Spectrum*, vol. 57, no. 5, pp. 35–39, 2020.

[191] C. for the Fourth Industrial Revolution, "Top 10 Emerging Technologies of 2024," World Economic Forum, Tech. Rep., 6 2024.

[192] Technical Specification Group Radio Access Network, "Study on New Radio (NR) to support non-terrestrial networks (Release 15)," 3rd Generation Partnership Project, Tech. Rep. TR 38.811 V15.4.0 (2020-10), 2017.

[193] ——, "Study on scenarios and requirements for next generation access technologies (Release 14)," 3rd Generation Partnership Project, Tech. Rep. TR 38.913 V18.0.0 (2024-04), 2015.

[194] Gatehouse Satcom, "The State of 5G NR NTN: Key insights and future directions for 2025 and beyond," https://gatehousesatcom.com/watch-webinar-the-state-of-5g-new-radio-ntn-2025/, [Online; accessed 01-February-2025].

[195] Technical Specification Group Radio Access Network, "Solutions for NR to support non-terrestrial networks (NTN) (Release 16)," 3rd Generation Partnership Project, Tech. Rep. TR 38.821 V16.2.0 (2023-03), 2018.

[196] Technical Specification Group Services and System Aspects, "Study on using Satellite Access in 5G; Stage 1 (Release 16)," 3rd Generation Partnership Project, Tech. Rep. TR 22.822 V16.0.0 (2018-06), 2017.

[197] M. Petry, R. Mayr, and M. Hennerich, "5G Base-Station with Hardware Acceleration for Non-Terrestrial Networks on a Space-Grade System-on-Chip," *Proceedings of the GNU Radio Conference*, vol. 9, no. 1, 2024. [Online]. Available: https://pubs.gnuradio.org/index.php/grcon/article/view/152

[198] Technical Specification Group Radio Access Network, "Study on Central Unit (CU) - Distributed Unit (DU) lower layer split for NR (Release 15)," 3rd Generation Partnership Project, Tech. Rep. TR 38.816 V15.0.0 (2017-12), 2017.

[199] ——, "NG-RAN; F1 application protocol (F1AP) (Release 15)," 3rd Generation Partnership Project, Tech. Rep. TS 38.473 V18.4.0 (2024-12), 2017.

[200] ——, "Study on new radio access technology: Radio access architecture and interfaces (Release 14)," 3rd Generation Partnership Project, Tech. Rep. TR 38.801 V14.0.0 (2017-03), 2016.

[201] L. M. P. Larsen, A. Checko, and H. L. Christiansen, "A Survey of the Functional Splits Proposed for 5G Mobile Crosshaul Networks," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 146–172, 2019.

[202] X. Lin, "An Overview of the 3GPP Study on Artificial Intelligence for 5G New Radio," 2023. [Online]. Available: https://arxiv.org/abs/2308.05315

[203] ——, "The Bridge Toward 6G: 5G-Advanced Evolution in 3GPP Release 19," 2023. [Online]. Available: https://arxiv.org/abs/2312.15174

[204] Technical Specification Group Radio Access Network, "Study on Artificial Intelligence (AI)/Machine Learning (ML) for NR air interface (Release 18)," 3rd Generation Partnership Project, Tech. Rep. TR 38.843 V18.0.0 (2023-12), 2022.

[205] ——, "Study on Artificial Intelligence (AI)/Machine Learning (ML) for mobility in NR (Release 19)," 3rd Generation Partnership Project, Tech. Rep. TR 38.744 V0.0.4 (2024-10), 2024.

[206] ——, "Study on enhancements for Artificial Intelligence (AI)/Machine Learning (ML) for NG-RAN (Release 19)," 3rd Generation Partnership Project, Tech. Rep. TR 38.743 V19.0.0 (2024-09), 2024.

[207] Technical Specification Group Services and System Aspects, "Study on Artificial Intelligence/Machine Learning (AI/ ML) management (Release 18)," 3rd Generation Partnership Project, Tech. Rep. TR 28.908 V18.1.0 (2024-09), 2024.

[208] X. Lin, "An Overview of AI in 3GPP's RAN Release 18: Enhancing Next-Generation Connectivity," https://www.comsoc.org/publications/ctn/overview-ai-3gpps-ran-release-18-enhancing-next-generation-connectivity, 2024, [Online; accessed 21-January-2025].