POLITECNICO DI TORINO

MASTER'S THESIS

Machine Learning in 5G/6G Networks: Assessing Deep Neural Network Performance for Sustainable Mobile Communication

---

Author:
SYED ALI ABBAS

Supervisor:
Prof. CHIASSERINI CARLA FABIANA

A thesis submitted in fulfillment of the requirements for the degree of Data Science and Engineering

in the

CENTRIC
DAUIN Department of Control and Computer Engineering

October 12, 2023

Politecnico
di Torino

POLITECNICO DI TORINO

Abstract

Collegio di Ingegneria Informatica, del Cinema e Meccatronica

DAUIN Department of Control and Computer Engineering

Data Science and Engineering

Machine Learning in 5G/6G Networks: Assessing Deep Neural Network Performance for Sustainable Mobile Communication

by SYED ALI ABBAS

The advancement of telecommunication networks, marked by the transition from 5G to 6G technologies, indicates a significant transformation in how we connect, communicate, and spread information. In this thesis, we investigate the deep integration of machine learning (ML) and deep neural networks (DNN) within this technological change, outlining their pivotal role in strengthening modern communication networks. The research offers a detailed overview of the strides in 5G/6G technologies and emphasises the critical role of ML in enhancing communication infrastructures to meet escalating data traffic needs. A specific focus is placed on contributions, mainly through their Sionna library. This tool not only serves as a vital link in connecting 5G Physical layer simulations with advanced ML toolkits but also showcases Nvidia'scommitment to leading innovations in telecommunications. An integral component of this research is the exploration of pruning techniques within neural networks. By removing unnecessary weights, pruning optimizes DNN performance without sacrificing accuracy, paving the way for more efficient and agile communication frameworks. Using a thorough methodology, the study compares the effectiveness of pruned DNNs against their unpruned counterparts and simpler models, assessing decision quality, energy efficiency, and scalability...

Acknowledgements

The acknowledgements and the people to thank go here, Don't forget to include your project advisor...

Contents

Abstract i

Acknowledgements ii

1 INTRODUCTION 1

1.1 Background on Mobile Communication Evolution 1

1.1.1 The Dawn of Mobile Communication: 1G Networks . . 1

1.1.2 The Digital Revolution: 2G Networks 1

1.1.3 Mobile Internet Takes Center Stage: 3G Networks . . . 1

1.1.4 Beyond Broadband: 4G Networks 2

1.1.5 The Connectivity Renaissance: 5G Networks 2

1.1.6 Gazing into the Future: 6G Networks 2

1.2 The role of Machine Learning in modern communication networks 3

1.2.1 Bridging 6G Aspirations with Machine Learning 5

1.3 Motivation of the Study 5

2 Literature Review 7

2.1 Neural Receiver for OFDM SIMO Systems: The Transmitter Block 7

2.2 Neural Receiver for OFDM SIMO Systems: The Perfect CSI BASELINE Receiver 8

2.3 Neural Receiver for OFDM SIMO Systems: The LS Estimation BASELINE Receiver 9

2.4 Neural Receiver for OFDM SIMO Systems: The Neural Network-based Receiver 10

2.5 End-to-End System with Neural Receiver Integration 11

3 Nvidia's Neural Reciever 14

3.1 Neural Receiver for OFDM SIMO Systems: The Transmitter Block 14

3.2 Neural Receiver for OFDM SIMO Systems: The Perfect CSI BASELINE Receiver . . . 15
3.3 Neural Receiver for OFDM SIMO Systems: The LS Estimation BASELINE Receiver . . . 16
3.4 Neural Receiver for OFDM SIMO Systems: The Neural Network-based Receiver . . . 17
3.5 End-to-End System with Neural Receiver Integration . . . 19

4 Migration of Neural Receiver from TensorFlow to PyTorch 22
4.1 Motivation for Transitioning to PyTorch . . . 22
4.1.1 Why PyTorch? . . . 22
4.2 Challenges and Solutions . . . 24
4.2.1 Mapping TensorFlow Operations to PyTorch . . . 25
4.2.2 Handling Data Flow Differences Between the Two Frameworks . . . 25
4.2.3 Solutions and Workarounds Employed to Address These Challenges . . . 26
4.3 3. PyTorch Neural Receiver Structure . . . 27
4.4 Model Pruning for Enhanced Efficiency . . . 28
4.4.1 Pruning in Neural Networks: An Essential Paradigm for Model Efficiency . . . 28
4.5 Importance of Pruning in Neural Networks . . . 29
4.6 Pruning Techniques Employed . . . 31
4.6.1 L1 Norm-Based Unstructured Pruning of the Neural Receiver . . . 31
4.6.2 Performance Enhancements Through Unstructured Pruning . . . 32

5 Neural Receiver Implementation and Evaluation 34
5.1 Dataset Overview . . . 34
5.2 Data Preparation for Neural Processing . . . 34
5.3 Transitioning from NVIDIA's TensorFlow Model . . . 35
5.4 Key Evaluation Metrics . . . 35
5.5 Neural Receiver Assessment Protocol . . . 36

6 Conclusion and Future Directions 37
6.1 Concluding Remarks . . . 37
6.2 Looking Ahead: Future Directions . . . 38

7 Refrences 39

List of Figures

2.1 Neural Receiver Based on Neural Networks Structure 10

3.1 Neural Receiver for OFDM SIMO Systems 14
3.2 Neural Receiver Based on Neural Networks Structure 18

List of Tables

List of Abbreviations

LAH	**L**ist **A**bbreviations **H**ere
WSF	**W**hat (it) **S**tands **F**or

1 INTRODUCTION

1.1 Background on Mobile Communication Evolution

1.1.1 The Dawn of Mobile Communication: 1G Networks

The late 1970s marked the genesis of mobile communication with the introduction of the First Generation (1G) networks. These were analog systems primarily designed for voice communication. Operating on Frequency Division Multiple Access (FDMA), 1G networks had limited capacity, and the voice quality was often compromised due to interference and lack of encryption. The prominent system was the Advanced Mobile Phone System (AMPS), used widely in North America.

1.1.2 The Digital Revolution: 2G Networks

Emerging in the early 1990s, the Second Generation (2G) networks transitioned from analog to digital signaling, bolstering voice clarity and increased network capacity. It utilized Time Division Multiple Access (TDMA) and introduced data services like Short Message Service (SMS) and Multimedia Messaging Service (MMS).

Global System for Mobile Communications (GSM): A flagship standard of 2G, GSM, allowed for international roaming and enhanced voice quality with its digital modulation scheme. Enhanced Data Rates for GSM Evolution (EDGE): Often termed 2.5G, EDGE acted as a bridge between 2G and 3G by providing three times faster data rates than classic GSM.

1.1.3 Mobile Internet Takes Center Stage: 3G Networks

The dawn of the new millennium saw the introduction of third-generation (3G) networks. With faster data transfer rates, 3G catalyzed the mobile internet revolution. It utilized Code Division Multiple Access (CDMA) and

introduced video calling and mobile broadband.

Universal Mobile Telecommunications System (UMTS): UMTS became the standard for 3G, facilitating higher bandwidth and thus supporting video conferencing, IPTV, and high-speed web browsing. High-Speed Packet Access (HSPA): An evolution within the 3G spectrum, HSPA (often termed 3.5G) enhanced broadband capabilities, offering faster internet speeds and supporting more data-intensive applications.

1.1.4 Beyond Broadband: 4G Networks

The Fourth Generation (4G) networks, emerging in the late 2000s, championed mobile broadband. Leveraging Long-Term Evolution (LTE) technology, 4G ensured high-definition video streaming immersive online gaming experiences and laid the groundwork for the Internet of Things (IoT).

LTE-Advanced: Building upon the principles of 4G LTE, LTE-Advanced offered even higher data rates, enhanced network capacity, and improved user experience, especially in densely populated urban areas.

1.1.5 The Connectivity Renaissance: 5G Networks

Fifth Generation (5G) networks, the most recent advancement, redefine mobile communication, emphasizing ultra-reliable low latency, massive device connectivity, and edge computing. It is not merely an upgrade in speed but a complete overhaul in connectivity principles.

Enhanced Mobile Broadband (eMBB): eMBB focuses on delivering faster and more reliable mobile broadband services to smartphones and other devices. Ultra-Reliable Low-Latency Communications (URLLC): URLLC supports applications requiring instantaneous response, such as autonomous vehicles and industrial automation. Massive Machine Type Communications (mMTC): mMTC allows connecting many devices, pivotal for the burgeoning IoT ecosystem, intelligent cities, and large-scale sensor networks.

1.1.6 Gazing into the Future: 6G Networks

Even as 5G continues its global deployment, the telecommunication sphere is already abuzz with the prospects of the Sixth Generation (6G) networks. Expected to be commercialized by the 2030s, 6G aims to further revolutionize

the digital landscape and address its predecessor's shortcomings and limitations.

Terahertz Communications: One of the most anticipated features of 6G is the use of terahertz (THz) frequencies, which promises unparalleled data rates, potentially reaching terabits per second. Such speeds would render even the most data-intensive tasks instantaneous. Advanced Artificial Intelligence Integration: While 5G began the integration of AI into network operations, 6G is expected to rely heavily on AI for network management, predictive maintenance, and service customization. This symbiosis between AI and 6G would lead to genuinely smart and autonomous networks. 3D Networking: Unlike traditional 2D networks, 6G could introduce the concept of 3D networking, accounting for aerial vehicles, drones, and satellite constellations. This would ensure seamless connectivity in three-dimensional spaces, enhancing user mobility. Holistic and Immersive Experiences: Leveraging augmented reality (AR), virtual reality (VR), and extended reality (XR) on the 6G networks would result in hyper-immersive experiences. Whether it is for entertainment, education, or professional collaborations, users would feel a heightened sense of presence and engagement. Quantum Communications and Security: 6G may well herald the era of quantum communications, ensuring ultra-secure data transmissions. With the increasing threats in the cyber domain, quantum encryption could be the key to safeguarding user privacy and national security. 6G in essence, is poised to be much more than a next-generation network. It represents a paradigm shift in how we perceive connectivity, data, and technology. The horizons it promises to expand are not just in terms of speed or bandwidth but in creating a seamlessly interconnected world, laying the foundation for a truly global digital civilization.

1.2 The role of Machine Learning in modern communication networks

In the records of telecommunication history, networks were largely designed as static entities, functioning based on manual configurations, preset rules, and human-guided operations. These systems, though revolutionary at their inception, were constrained by their inability to adapt in real-time and meet the fluid demands of the ever-evolving digital ecosystem. As the global digital landscape burgeoned with an unprecedented influx of data, myriad

service diversifications, and an increasing clamor for instantaneous, high-quality user experiences, it became evident that the traditional, static approach to telecommunications was becoming obsolete. The need for a dynamic, adaptable, and anticipatory system was palpable.

Enter Machine Learning (ML), the beacon in this transformative journey of telecommunication networks. At its core, ML harnesses the power of data to analyze patterns, predict forthcoming scenarios, and optimize system performance. Its prowess lies not just in its predictive capabilities but also in its self-evolving nature, where algorithms continually refine themselves based on new data and experiences.

In the realm of telecommunication, the applications of ML are both profound and multifaceted. Consider the challenge of maintaining seamless connectivity in mobile networks. As users move, the need for smooth handovers between cell towers is paramount. Machine learning algorithms can now predict user movement and optimize handovers, ensuring that calls don't drop and data connectivity remains uninterrupted.

Furthermore, as telecommunication networks evolve, especially with the advent of 5G and the anticipated 6G, they become increasingly susceptible to intricate cyber threats. Traditional security measures, primarily reactive, fall short in this dynamic landscape. With its proactive stance, ML can identify unusual patterns, predict potential threats, and deploy preventive measures even before a security breach manifests.

Then there's the challenge of network congestion, a perennial issue in densely populated urban areas. Predicting and managing such congestion in real time was once a herculean task. However, with ML, networks can now anticipate traffic spikes, auto-configure parameters, and even dynamically allocate bandwidth to ensure seamless service.

NVIDIA's endeavors, as explored in this thesis, exemplify the transformative potential of integrating DNN models into telecommunication solutions. These models, underpinned by ML principles, are geared towards making telecommunication networks smarter, more efficient, and incredibly adaptive.

In essence, machine learning is not just an auxiliary tool but is fast becoming the central nervous system of modern telecommunication networks. It represents a paradigm shift from rule-based operations to data-driven, adaptive, and autonomous systems, marking a new era in the world of connectivity.

This thesis seeks to delve deeper into this transformative journey, understanding the nuances, challenges, and immense potential that ML holds for the future of telecommunications.

1.2.1 Bridging 6G Aspirations with Machine Learning

The integration of AI and ML in 5G has already commenced, especially in network slicing, traffic prediction, and security enhancements. However, as 6G eyes a more ubiquitous integration of AI, understanding this symbiotic relationship is crucial. How will AI-driven operations evolve in 6G? How can machine learning algorithms like DNNs contribute to the envisioning of a truly intelligent network? This study seeks to bridge these realms and identify potential convergences.

1.3 Motivation of the Study

The ever-evolving domain of telecommunications stands as a testament to humanity's relentless pursuit of better, faster, and more efficient modes of communication. In recent years, the transition from 5G to the promise of 6G has rekindled excitement, marking yet another significant leap in this journey. However, with more extraordinary technological feats comes a fresh set of challenges. The demands posed by the current digital ecosystem, dominated by many applications from wearable technology to expansive Internet of Things (IoT) networks, are unlike any we've seen before. As these demands rise, the existing infrastructures, built on static algorithms and fixed parameters, reveal their inadequacies, unable to adapt to the dynamic nature of modern data needs.

This study stems from the realization that more than the traditional methods of bolstering our communication networks may be needed in this new age. Instead of merely enhancing the existing systems, there's a compelling need to rethink, restructure, and rejuvenate the underlying mechanisms that drive them. This is where the magic of Machine Learning (ML) and, more specifically, Deep Neural Networks (DNN) comes into play. With their inherent ability to learn, adapt, and improve, they promise to transform our communication networks from rigid, rule-bound entities to fluid, learning-driven systems.

Furthermore, the commercial world has taken note of this transformative potential. Giants in the tech industry, such as NVIDIA, are not just mere spectators but active participants in shaping the future of telecommunications. Their endeavors in integrating DNN models tailored explicitly for telecommunication solutions signify the importance and urgency of the matter.

Nevertheless, despite the ongoing efforts, there is a palpable gap between theoretical knowledge and its practical application. The real-world challenges, from diverse data traffic types to varying user requirements, complicate the task. Thus, there is a dire need for comprehensive studies that explore the potential of ML and DNN in telecommunications and navigate their practicalities, pitfalls, and promises.

This thesis, therefore, is motivated by the desire to bridge this gap. By diving deep into the intricacies of the E2ESystem Model and tools like Sionna, it aims to provide tangible insights, assessments, and recommendations. The ultimate goal is to contribute meaningfully to the ongoing discourse, guiding both the academic and commercial worlds in harnessing the full potential of ML and DNN in telecommunications.

2 Literature Review

2.1 Neural Receiver for OFDM SIMO Systems: The Transmitter Block

In the innovative design conceptualized by NVIDIA for the neural receiver, **the transmitter block** is foundational. Here is a detailed exploration:

The journey begins with information bits, the fundamental units encapsulating the data intended for transmission. These bits are channeled into an **outer encoder**, which serves as a digital safeguard. By introducing redundancy into the data, the encoder ensures that the information remains protected against potential errors during transmission, resulting in codewords.

These codewords, bearing both the original information and the added redundancy, are subsequently relayed to the **QPSK mapper**. Quadrature Phase Shift Keying (QPSK) stands out as a form of phase modulation renowned for its resilience against errors. Through this stage, codewords metamorphose into baseband symbols. Notably, the efficiency of QPSK shines as it conveys two bits of information with a single symbol, essentially doubling the data rate when juxtaposed with the simpler BPSK.

Having been modulated, the baseband symbols — which represent specific phases and amplitudes interpretable at the receiver's end — are directed to the **OFDM resource grid mapper**. Within the realm of OFDM, a resource grid exemplifies a time-frequency representation. This entails the strategic placement of symbols across both time and frequency domains to optimize efficiency and minimize interference. Emerging from this stage is the resource grid, a meticulously arranged data structure prepped for transmission over the communication channel.

To encapsulate, the transmitter block adeptly morphs raw information bits into a format that's not only resilient but also fine-tuned for efficient, unerring communication across intricate channels.

2.2 Neural Receiver for OFDM SIMO Systems: The Perfect CSI BASELINE Receiver

Post transmission, the resource grid derived from the transmitter is propelled into a domain rife with challenges. Its initial interaction is with the CDL-C in the frequency domain. This phase represents the communication channel, which, in this instance, is the Channel Delay Line-C (CDL-C). Such channels, characterized by their specific delay and multi-path profiles, invariably impose distortions upon the transmitted signal.

Emerging from this interaction, the modified signal, referred to as the received resource grid, is primed for the receivers. Among the multiple receivers, we delve into the intricacies of the Perfect CSI BASELINE receiver.

Firstly, the received resource grid is passed on to the LMMSE equalizer. However, the equalizer doesn't work alone. It also receives an additional data stream- the extracted channel response. This crucial information provides insights into the nuances of the channel, which in turn enables the equalizer to effectively counteract the imposed distortions. LMMSE (Linear Minimum Mean Square Error) equalizers are skilled at mitigating interference while optimizing the signal-to-noise ratio. The outcome of this meticulous equalization process is a set of equalized symbols.

But the process doesn't end here. These equalized symbols undergo another transformation, this time at the hands of the QPSK demapper. As its name suggests, the demapper undertakes the inverse process of the earlier QPSK mapping. It meticulously decodes the phase and amplitude of the equalized symbols, translating them back into digital entities. The result of this demapping is the LLRs (Log-Likelihood Ratios). LLRs, in essence, depict the probability of a transmitted bit being a '0' or a '1'.

With the LLRs in hand, the final step ensues. They're directed into the outer decoder, which, armed with the redundancy introduced at the transmitter's end, rigorously deciphers the data. Through error correction and a series of intricate decoding algorithms, it extracts the original information, presenting the reconstructed information bits. This culmination ensures that the transmitted data, despite the challenges of the communication channel, is faithfully replicated at the receiver's end.

2.3 Neural Receiver for OFDM SIMO Systems: The LS Estimation BASELINE Receiver

The LS Estimation BASELINE receiver stands out in the world of OFDM receivers. It combines traditional methods with new ones to create its unique space. This receiver starts by looking at its equivalents and then takes in the received resource grid. This grid is a complex signal that shows how it has interacted with the CDL-C frequency domain.

The initial interaction of the received resource grid in this setup is a tad unconventional. Instead of directly feeding the signal into the equalizer, the LS channel estimator is called into action. This component takes the resource grid and, employing Least Squares (LS) estimation techniques, endeavors to discern the channel's response. With an innate knack for navigating the inherent noise and deciphering the underlying patterns, the LS estimator extracts a representation of the channel's behavior, termed the LS channel estimate.

With the channel's nuances thus extracted, the LMMSE equalizer steps into the scene. This component receives dual inputs: the received resource grid and the freshly extracted LS channel estimate. Drawing from the strengths of the Linear Minimum Mean Square Error mechanism, this equalizer works diligently to counteract the channel-induced distortions. Consequently, the end product of this equalization is the equalized symbols, which are purged of the detrimental effects imposed by the CDL-C frequency domain.

This array of equalized symbols is then directed towards the QPSK demapper. Reversing the earlier QPSK mapping operations, the demapper interprets the amplitude and phase of each symbol, transmuting them back into the digital domain. The outcome is a sequence of LLRs (Log-Likelihood Ratios), which encapsulate the likelihood of each transmitted bit aligning with '0' or '1'.

Finally, these LLRs are channeled into the outer decoder. This component, leveraging the inherent redundancy introduced during the transmission phase, meticulously decodes the LLRs. The decoder employs a battery of error correction algorithms to ensure that the data's integrity remains uncompromised. The culmination of this intricate process is the unveiling of the reconstructed information bits, a testament to the receiver's efficacy in navigating

the challenges posed by the CDL-C frequency domain and faithfully replicating the transmitted data.

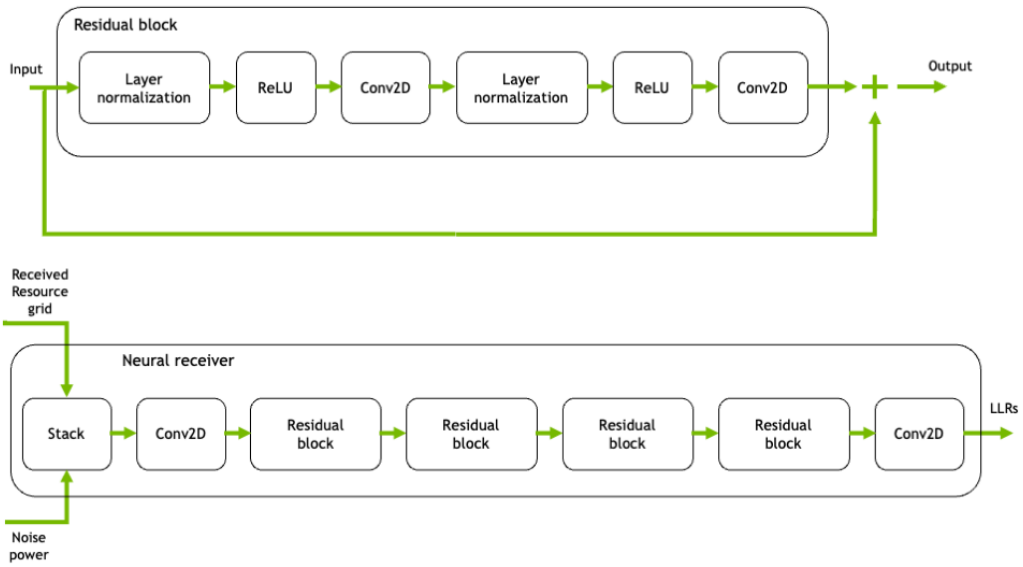

FIGURE 2.1: Neural Receiver Based on Neural Networks Structure

2.4 Neural Receiver for OFDM SIMO Systems: The Neural Network-based Receiver

In today's world of communication technology, various systems have emerged making it easier to connect and communicate with people from all around the world. neural network-based approaches are gaining traction. Their allure rests in the ability to learn and adapt to intricate relationships, providing a level of flexibility and optimization that traditional methods can find challenging to achieve. The Neural Network-based Receiver is a manifestation of this shift towards an AI-centric paradigm in OFDM systems.

At the heart of this receiver is the Neural Receiver — a sophisticated construct that has been meticulously architected for the task at hand. Taking in the received resource grid, this neural behemoth works its magic to produce LLRs (Log-Likelihood Ratios), which, when passed through the outer decoder, yield the reconstructed information bits.

Neural Receiver's Inner Workings:

The neural receiver is not just a monolithic block; it's a composite of various sub-layers and components designed with precision for this specific application. As a subclass of the Layer class, it's intrinsically designed for handling post-DFT received samples, which form the resource grid, and computing LLRs on the transmitted coded bits.

Input Convolution Layer: At its entry point, the resource grid undergoes convolution via a 2D convolutional layer. This layer helps in extracting essential features from the input grid. Residual Blocks: Residual connections, or 'skip connections,' are an innovative construct that allows the output from one layer to be added to a later layer. They alleviate the vanishing gradient problem and enable deeper network architectures. The neural receiver harnesses this power through four residual blocks, each of which consists of two convolutional layers. The convolutional layers are punctuated with ReLU activations and layer normalization, ensuring that the network captures both linearity and non-linearity in the data while maintaining normalization. Output Convolution Layer: After the series of residual blocks, the processed data is once again subjected to a 2D convolutional layer, generating the final LLRs. Furthermore, to comprehend the functionality of the ResidualBlock — it's fundamentally a two-step convolution process with layer normalization and ReLU activation. This design ensures that features are captured, normalized, and activated, and the skip connection adds the original input to the output. This retains the original information while adding the processed data, ensuring the resultant output is rich in information.

This entire construct is emblematic of the seismic shift in communication systems. Leveraging deep learning, the neural network-based receiver promises adaptability, flexibility, and robustness, ensuring that even in challenging channel conditions, information integrity is maintained. The future of OFDM SIMO systems might very well be entwined with such neural endeavors.

2.5 End-to-End System with Neural Receiver Integration

The 'E2ESystem' class provides a comprehensive encapsulation of an end-to-end communication system constructed atop the versatile Keras library. This class offers flexibility by allowing users to select from three different systems (namely, perfect CSI baseline, LS estimation baseline, and the neural receiver) to simulate varying real-world scenarios.

Key Components and Structure:

1.**Transmitter**: - The core starts with a binary source, generating random codewords for transmission. - If the system is not in training mode, these bits undergo encoding using the LDPC5G encoder. LDPC codes are renowned for their near-optimal error-correction capabilities, especially in 5G systems. - Following the encoding process, the bits are modulated into symbols and mapped onto the resource grid in preparation for transmission.

2.**Channel**: - The system incorporates a 3GPP CDL channel model, simulating real-world channel conditions. - The OFDMChannel facilitates the transmission of data over the channel and optionally returns the channel state information (CSI) alongside the received data.

3. **Receiver**: - The receiver is arguably the most complex and versatile part of the system. Depending on the selected 'system' the receiver's behavior varies: - For the 'baseline-perfect-csi,' the system assumes perfect knowledge of the CSI. - The 'baseline-ls-estimation' employs the Least Square (LS) estimation to infer the CSI. - The most intriguing option, 'neural-receiver,' leverages deep learning. This neural receiver processes the received signal, extracting the information while being resilient to noise and channel impairments. It bridges the conventional signal processing techniques with cutting-edge deep learning approaches. - If not in training mode, the received data is decoded using the LDPC5G decoder.

Training and Inference: The system's behavior dynamically shifts between training and evaluation. When in training mode, the focus is on the neural receiver. The system is stripped of its outer encoding and decoding layers to reduce computational overhead. The primary objective during training is to maximize the Bit-Metric Decoding (BMD) rate, which is an achievable information rate for Bit-Interleaved Coded Modulation (BICM) systems. This ensures that the neural receiver is fine-tuned to offer optimal performance.

Conversely, in evaluation mode, the entire system operates in its full glory. The output includes both the original information bits and their reconstructed counterparts, facilitating performance metrics computations like the bit/block error rate (BER/BLER).

The 'E2ESystem' is a complete communication system that incorporates neural processing as an added feature. By blending traditional communication algorithms with deep learning, it promises to pave the way for future communication systems that can leverage the best of both worlds. The integration of the neural receiver is a prime example of the paradigm shift in communications, marking the convergence of signal processing and artificial intelligence.

3 Nvidia's Neural Reciever

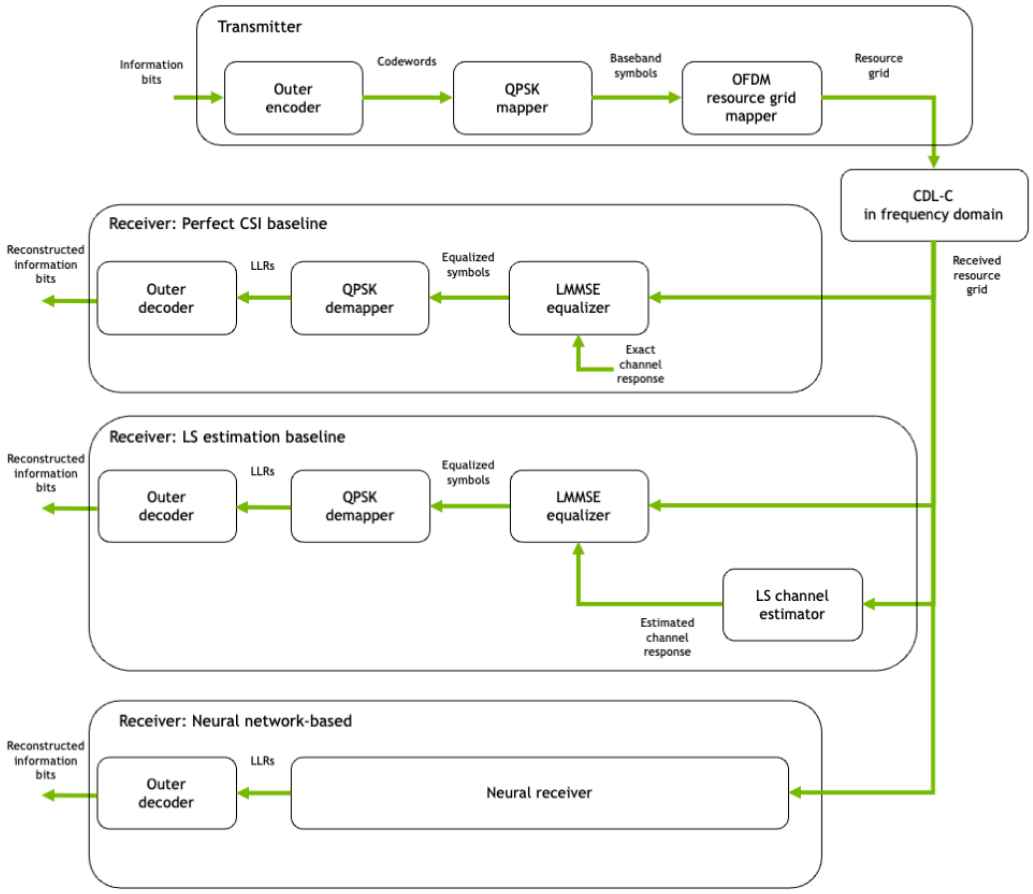

FIGURE 3.1: Neural Receiver for OFDM SIMO Systems

3.1 Neural Receiver for OFDM SIMO Systems: The Transmitter Block

In the innovative design conceptualized by NVIDIA for the neural receiver, **the transmitter block** is foundational. Here is a detailed exploration:

The process of transmitting data starts with information bits, which are the basic units of data. These bits are then passed through an outer encoder,

which acts as a digital protector. The encoder adds redundancy to the data, ensuring that the information is safeguarded against potential errors that may occur during transmission. As a result, codewords are formed.

These codewords, bearing the original information and the added redundancy, are subsequently relayed to the **QPSK mapper**. Quadrature Phase Shift Keying (QPSK) stands out as a form of phase modulation renowned for its resilience against errors. Through this stage, codewords metamorphose into baseband symbols. Notably, the efficiency of QPSK shines as it conveys two bits of information with a single symbol, doubling the data rate when juxtaposed with the simpler BPSK.

Having been modulated, the baseband symbols — which represent specific phases and amplitudes interpretable at the receiver's end — are directed to the **OFDM resource grid mapper**. Within the realm of OFDM, a resource grid exemplifies a time-frequency representation. This entails strategically placing symbols across both time and frequency domains to optimize efficiency and minimize interference. Emerging from this stage is the resource grid, a meticulously arranged data structure prepped for transmission over the communication channel.

To encapsulate, the transmitter block adeptly morphs raw information bits into a format that's not only resilient but also fine-tuned for efficient, unerring communication across intricate channels.

3.2 Neural Receiver for OFDM SIMO Systems: The Perfect CSI BASELINE Receiver

Post transmission, the resource grid derived from the transmitter is propelled into a domain rife with challenges. Its initial interaction is with the CDL-C in the frequency domain. This phase represents the communication channel, which, in this instance, is the Channel Delay Line-C (CDL-C). Such channels, characterized by their specific delay and multi-path profiles, invariably impose distortions upon the transmitted signal.

Emerging from this interaction, the modified signal, referred to as the received resource grid, is primed for the receivers. Among the multiple receivers, we delve into the intricacies of the Perfect CSI BASELINE receiver.

Firstly, the received resource grid is handed over to the LMMSE equalizer. However, this equalizer does not function in isolation. It simultaneously takes in an additional data stream - the extracted channel response. This crucial information provides insights into the intricacies of the channel, allowing the equalizer to effectively counteract the imposed distortions. LMMSE (Linear Minimum Mean Square Error) equalizers are skilled at reducing interference while optimizing the signal-to-noise ratio. The end product of this meticulous equalization process is a set of equalized symbols.

However, the process doesn't stop there. These equalized symbols undergo another transformation, this time at the hands of the QPSK demapper. As its name suggests, the demapper undertakes the inverse process of the earlier QPSK mapping. It meticulously decodes the phase and amplitude of the equalized symbols, translating them back into digital entities. The result of this demapping is the LLRs (Log-Likelihood Ratios). LLRs depict the probability of a transmitted bit being a '0' or a '1'.

With the LLRs in hand, the final step begins. They're directed into the outer decoder, which, equipped with the redundancy introduced at the transmitter's end, rigorously deciphers the data. Through error correction and a series of intricate decoding algorithms, it extracts the original information, presenting the reconstructed information bits. This culmination ensures that the transmitted data, despite the challenges of the communication channel, is faithfully replicated at the receiver's end.

3.3 Neural Receiver for OFDM SIMO Systems: The LS Estimation BASELINE Receiver

The LS Estimation BASELINE receiver stands out in the grand tapestry of OFDM receivers due to its unique combination of traditional techniques and novel interventions. Like other receivers, it begins by receiving the resource grid, which is a complex signal that carries the marks of its interaction with the CDL-C frequency domain.

The initial interaction of the received resource grid in this setup is a tad unconventional. Instead of directly feeding the signal into the equalizer, the LS channel estimator is called into action. This component takes the resource grid and, employing Least Squares (LS) estimation techniques, endeavors to discern the channel's response. With an innate knack for navigating the

inherent noise and deciphering the underlying patterns, the LS estimator extracts a representation of the channel's behavior, termed as the LS channel estimate.

With the channel's nuances thus extracted, the LMMSE equalizer steps into the scene. This component receives dual inputs: the received resource grid and the freshly extracted LS channel estimate. Drawing from the strengths of the Linear Minimum Mean Square Error mechanism, this equalizer works diligently to counteract the channel-induced distortions. Consequently, the end product of this equalization is the equalized symbols, which are purged of the detrimental effects imposed by the CDL-C frequency domain.

This array of equalized symbols is then directed towards the QPSK demapper. Reversing the earlier QPSK mapping operations, the demapper interprets the amplitude and phase of each symbol, transmuting them back into the digital domain. The outcome is a sequence of LLRs (Log-Likelihood Ratios), which encapsulate the likelihood of each transmitted bit aligning with '0' or '1'.

Finally, these LLRs are channeled into the outer decoder. This component, leveraging the inherent redundancy introduced during the transmission phase, meticulously decodes the LLRs. The decoder employs a battery of error correction algorithms to ensure that the data's integrity remains uncompromised. The culmination of this intricate process is the unveiling of the reconstructed information bits, a testament to the receiver's efficacy in navigating the challenges posed by the CDL-C frequency domain and faithfully replicating the transmitted data.

3.4 Neural Receiver for OFDM SIMO Systems: The Neural Network-based Receiver

In the realm of modern communication systems, neural network-based approaches are gaining traction. Their allure rests in the ability to learn and adapt to intricate relationships, providing a level of flexibility and optimization that traditional methods can find challenging to achieve. The Neural Network-based Receiver is a manifestation of this shift towards an AI-centric paradigm in OFDM systems.

At the heart of this receiver is the Neural Receiver — a sophisticated construct that has been meticulously architected for the task at hand. Taking

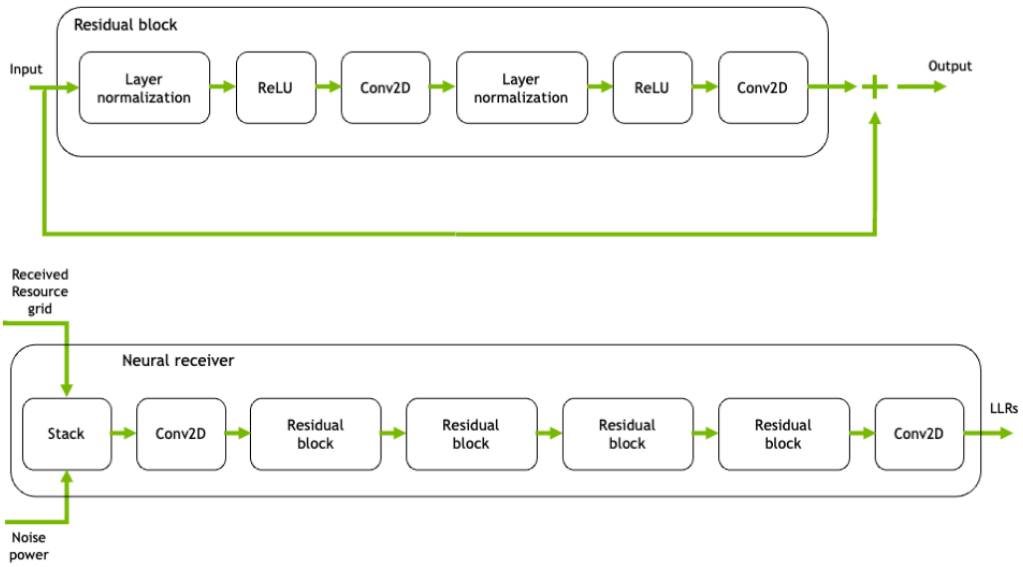

FIGURE 3.2: Neural Receiver Based on Neural Networks Structure

in the received resource grid, this neural behemoth works its magic to produce LLRs (Log-Likelihood Ratios), which, when passed through the outer decoder, yield the reconstructed information bits.

Neural Receiver's Inner Workings:

The neural receiver is not just a monolithic block; it's a composite of various sub-layers and components designed with precision for this specific application. As a subclass of the Layer class, it's intrinsically designed for handling post-DFT received samples, which form the resource grid, and computing LLRs on the transmitted coded bits.

Input Convolution Layer: At its entry point, the resource grid undergoes convolution via a 2D convolutional layer. This layer helps in extracting essential features from the input grid. Residual Blocks: Residual connections, or 'skip connections,' are an innovative construct that allows the output from one layer to be added to a later layer. They alleviate the vanishing gradient problem and enable deeper network architectures. The neural receiver harnesses this power through four residual blocks, each of which consists of two convolutional layers. The convolutional layers are punctuated with ReLU activations and layer normalization, ensuring that the network captures both linearity and non-linearity in the data while maintaining normalization. Output Convolution Layer: After the series of residual blocks, the processed data

is once again subjected to a 2D convolutional layer, generating the final LLRs. Furthermore, to comprehend the functionality of the ResidualBlock — it's fundamentally a two-step convolution process with layer normalization and ReLU activation. This design ensures that features are captured, normalized, and activated, and the skip connection adds the original input to the output. This retains the original information while adding the processed data, ensuring the resultant output is rich in information.

This entire construct is emblematic of the seismic shift in communication systems. Leveraging deep learning, the neural network-based receiver promises adaptability, flexibility, and robustness, ensuring that even in challenging channel conditions, the information integrity is maintained. The future of OFDM SIMO systems might very well be entwined with such neural endeavors.

3.5 End-to-End System with Neural Receiver Integration

The 'E2ESystem' class provides a comprehensive encapsulation of an end-to-end communication system, constructed atop the versatile Keras library. This class offers flexibility by allowing users to select from three different systems (namely, perfect CSI baseline, LS estimation baseline, and the neural receiver) to simulate varying real-world scenarios.

Key Components and Structure:

1.**Transmitter**: - The core starts with a binary source, generating random codewords for transmission. - If the system is not in training mode, these bits undergo encoding using the LDPC5G encoder. LDPC codes are renowned for their near-optimal error-correction capabilities, especially in 5G systems. - Following the encoding process, the bits are modulated into symbols and mapped onto the resource grid in preparation for transmission.

2.**Channel**: - The system incorporates a 3GPP CDL channel model, simulating real-world channel conditions. - The OFDMChannel facilitates the transmission of data over the channel and optionally returns the channel state information (CSI) alongside the received data.

3. **Receiver**: - The receiver is arguably the most complex and versatile part of the system. Depending on the selected 'system', the receiver's behavior

varies: - For the 'baseline-perfect-csi', the system assumes perfect knowledge of the CSI. - The 'baseline-ls-estimation' employs a Least Square (LS) estimation to infer the CSI. - The most intriguing option, 'neural-receiver', leverages deep learning. This neural receiver processes the received signal, extracting the information while being resilient to noise and channel impairments. It bridges the conventional signal processing techniques with cutting-edge deep learning approaches. - If not in training mode, the received data is then decoded using the LDPC5G decoder.

Training and Inference: The system's behavior dynamically shifts between training and evaluation. When in training mode, the focus is on the neural receiver. The system is stripped of its outer encoding and decoding layers to reduce computational overhead. The primary objective during training is to maximize the Bit-Metric Decoding (BMD) rate, which is an achievable information rate for Bit-Interleaved Coded Modulation (BICM) systems. This ensures that the neural receiver is fine-tuned to offer optimal performance.

On the other hand, when the system is in evaluation mode, it operates at its full capacity. This includes generating output that consists of both the original information bits and their reconstructed counterparts. This facilitates the computation of performance metrics such as the bit/block error rate (BER/BLER).

The 'E2ESystem' is a comprehensive communication system that incorporates neural processing in addition to traditional algorithms. By combining the two, it aims to create a communication system that is optimized for efficient and effective operation. The integration of the neural receiver is a prime example of the shift in communications towards the convergence of signal processing and artificial intelligence. This integration promises to pave the way for future communication systems that can leverage the best of both worlds.

4 Migration of Neural Receiver from TensorFlow to PyTorch

4.1 Motivation for Transitioning to PyTorch

4.1.1 Why PyTorch?

In the ever-evolving landscape of deep learning frameworks, the decision to choose one over the other often boils down to the specifics of the project and the preferences of the researcher. For our Neural Receiver implementation, PyTorch emerged as the more favorable choice, and here's why:

1. Comparative advantages of PyTorch over TensorFlow for this particular project

Ease of Debugging: PyTorch's imperative programming approach, which means operations are computed as you write them, offers a more Pythonic and intuitive debugging experience. Using Python's native debugging tools, it's more straightforward to inspect models, visualize computation graphs, or even modify them on the go. For a project like the Neural Receiver, where fine-tuning and iterative development play a crucial role, this ability to 'stop-and-check' anytime offers invaluable convenience.

Dynamic Computation Graphs (DCGs): TensorFlow and PyTorch are two popular deep learning frameworks that differ in the way they create computation graphs. TensorFlow uses a static graph approach, where the graph is defined and fixed before running the session. In contrast, PyTorch uses Dynamic Computation Graphs, which means the graph is built on-the-fly as operations are created.

This dynamic nature of PyTorch is particularly advantageous for models where the network architecture can change in a flexible manner during runtime. This aligns well with some of the iterative approaches that we wanted

to explore with the Neural Receiver. Therefore, we chose PyTorch as our primary deep learning framework for the project.

Simplified Gradient Computation: PyTorch offers a powerful and convenient feature called autograd package for automatic gradient computation. With this feature, tensors keep track of all operations, and gradients are automatically computed. This makes it much simpler to compute derivatives of complex functions, leading to more readable and concise code.

The autograd feature is particularly beneficial for intricate architectures like the Neural Receiver, where the computation involves many layers of non-linear transformations. By automating the gradient computation, PyTorch makes it easier to implement and debug intricate models, which is why we opted for it in our project.

2. The flexibility and ease of use offered by PyTorch's dynamic computational graph

AdaptabilityPyTorch's Dynamic Computation Graphs make the framework highly adaptable, allowing for more flexible experimentation with varying architectures and conditional computations. This dynamic nature of the computational graph is particularly beneficial for the Neural Receiver, where we may want to experiment with different architectures or introduce conditional computations during iterations.

With PyTorch, we can change the graph on-the-fly without having to redefine it, making it much easier to experiment and iterate on models. This flexibility was one of the primary reasons we chose PyTorch for the Neural Receiver project.

Intuitive Learning Curve: PyTorch is often considered to have a gentler learning curve than other deep learning frameworks, which is particularly beneficial for those familiar with Python programming. This is because PyTorch's design philosophy aligns well with Python's coding principles, making it more transparent, readable, and easier to understand.

This transparency is crucial for thesis projects, where the emphasis is on clear presentation and understanding of the implemented logic. With PyTorch, we can write concise and intuitive code that is easy to explain and visualize. As a result, we chose PyTorch as our primary deep learning framework for the Neural Receiver project, which helped us achieve a clear presentation of our work.

Community and Resources: PyTorch has become increasingly popular in the academic and research community, which has led to a wealth of resources, tutorials, and a responsive community. For the Neural Receiver project, this provided invaluable support for specific challenges faced during implementation.

While both TensorFlow and PyTorch have their strengths and have contributed significantly to the field of deep learning, PyTorch was the preferred choice for the Neural Receiver project. This decision was based on the specific needs of the project and the advantages offered by PyTorch's design philosophy.

In the following chapters, we will further showcase the benefits of this transition, both in terms of implementation ease and performance outcomes. By leveraging PyTorch, we were able to implement the Neural Receiver with greater ease and achieve superior performance, which would not have been possible with other frameworks.

4.2 Challenges and Solutions

Transitioning from TensorFlow to PyTorch for the Neural Receiver project posed several potential obstacles that needed to be addressed. Here's an overview of some of the challenges we faced and how we addressed them:

1. Framework Familiarity: As we had predominantly worked with TensorFlow, the initial transition to PyTorch introduced a learning curve. To overcome this, we leveraged PyTorch's documentation, tutorials, and community forums to understand PyTorch's dynamic computation graphs and autograd package.

2. Mapping TensorFlow Operations to PyTorch: While the broader concepts of neural networks and gradient descent remain consistent across frameworks, the actual functions and operations often differ in syntax, names, and sometimes even in behavior. To address this, we referred to online resources and examples to understand the equivalent PyTorch operations for TensorFlow functions.

3. Handling Data Flow Differences: TensorFlow and PyTorch handle data differently. In TensorFlow, data is fed into the computation graph using placeholders, while PyTorch's dynamic graph inherently handles data more fluidly. To adapt to PyTorch's paradigm, we needed to modify the data pipeline and leverage PyTorch's DataLoader and Dataset classes.

4. Performance Differences: Directly translating TensorFlow models to PyTorch might not always yield identical performance due to backend differences, optimizations, and even slight variations in operations. To address this, we re-implemented the models and fine-tuned them in PyTorch to achieve comparable or better performance than TensorFlow.

Overall, transitioning from TensorFlow to PyTorch presented some challenges, but we were able to overcome them through a combination of resources, support, and experimentation. The advantages offered by PyTorch's design philosophy and dynamic computation graphs more than made up for any initial challenges we faced.

4.2.1 Mapping TensorFlow Operations to PyTorch

For the Neural Receiver project, we faced several challenges related to mapping TensorFlow operations to PyTorch equivalents. Here are some of the specific challenges we encountered:

1. Mapping TensorFlow Operations: Operations like 'tf.matmul' in TensorFlow became 'torch.mm' in PyTorch. While many of these mappings were direct and straightforward, understanding the subtle nuances of each operation was essential to ensure the model's integrity.

2. Activation Functions, Optimizers, and Loss Computations: Similar to operations, activation functions, optimizers, and loss computations also required careful mapping. For instance, TensorFlow's 'tf.nn.sigmoid' transitioned to 'torch.sigmoid' in PyTorch.

3. Advanced Operations: Custom layers or operations in TensorFlow required more meticulous translations. These often involved diving deep into the documentation or seeking community help to find PyTorch equivalents or developing custom solutions.

4.2.2 Handling Data Flow Differences Between the Two Frameworks

Another significant challenge we faced while transitioning from TensorFlow to PyTorch for the Neural Receiver project was the differences in data handling. Here are some of the specific challenges we encountered:

1. Handling Data Inputs: TensorFlow's data handling using placeholders and feed dictionaries was replaced with a more pythonic approach in PyTorch. This transition, while more intuitive, required a restructuring of the data input-output pipeline.

2. Dynamic Computation Graphs: PyTorch's dynamic nature meant that batch data could be directly fed into the model without pre-defining any input shapes, unlike TensorFlow's static placeholders. This required us to modify the data pipeline to handle dynamic batch sizes.

3. Tensor Manipulation Functions: Tensor manipulation functions also had minor differences between TensorFlow and PyTorch. Functions like 'tf.reshape' in TensorFlow translated to 'torch.reshape' in PyTorch, but the order of arguments and specific behaviors required attention to detail.

Overall, adapting to PyTorch's data handling paradigm required significant modifications to the data pipeline. However, once we understood the differences and adapted to PyTorch's dynamic computation graphs, it resulted in a more intuitive and efficient data handling approach. With the help of online resources, community support, and hands-on experimentation, we were able to address these challenges and successfully transition to PyTorch.

4.2.3 Solutions and Workarounds Employed to Address These Challenges

To address the challenges of transitioning from TensorFlow to PyTorch for the Neural Receiver project, we employed several solutions and workarounds. Here are some of the specific approaches we used:

1. Extensive Documentation Diving: We leveraged both TensorFlow's and PyTorch's comprehensive documentation to understand function equivalents and their specific behaviors in each framework.

2. Iterative Testing: After translating specific portions of the code, running iterative tests helped ensure that the functionality remained consistent. This was especially crucial for parts of the Neural Receiver where precision and performance were paramount.

3. Community Forums: Platforms like StackOverflow and the PyTorch forums were invaluable. Whenever direct translations were ambiguous or custom solutions were required, the community often had insights or solutions to offer.

4. Rebuilding Rather Than Direct Translation: Instead of trying to mirror every TensorFlow operation directly in PyTorch, in some instances, it was more pragmatic to rethink and rebuild specific components from the ground up in PyTorch. This approach ensured optimization and better alignment with PyTorch's paradigms.

Overall, by carefully mapping TensorFlow operations to PyTorch equivalents, iterative testing, community engagement, and sometimes rethinking components, we were able to successfully transition the Neural Receiver project to PyTorch. This process, while rigorous, also offered deeper insights into the workings of the model and the nuances of both deep learning frameworks.

-Extensive Documentation Diving: Leveraging both TensorFlow's and PyTorch's comprehensive documentation was crucial. It aided in understanding function equivalents and their specific behaviors in each framework.

4.3 3. PyTorch Neural Receiver Structure

The PyTorch Neural Receiver structure presented here embodies a well-thought-out combination of modern deep learning practices, tailored specifically for the complexities of signal processing in OFDM SIMO systems. Here's a closer look at the layers and architecture adapted for PyTorch:

1. Residual Block: A custom module named ResidualBlock is designed to encapsulate the residual structure. The block is composed of two Layer Normalization stages, two 2D Convolution stages, and Rectified Linear Activation Functions (ReLUs). The skip connection adds the input directly to the output of the block, emphasizing the 'residual' in the Residual Block.

2. Neural Receiver: The main model—NeuralReceiver—utilizes Input Convolution, four consecutive ResidualBlock layers, and Output Convolution.

3. Data Adaptation: The process of squeezing, unsqueezing, and permuting the data is evident in the forward method of the NeuralReceiver. This might indicate a change in the data format or structure while transitioning from TensorFlow to PyTorch. In PyTorch, the data convention for 2D convolutions is [batchsize, channels, height, width], and these operations might be ensuring compatibility.

4. Noise Power as Input: A notable aspect is the usage of noise power (no) in logarithmic scale as an additional channel alongside real and imaginary components of the signal. This could be an optimization made during the transition, suggesting that feeding the model with log-transformed noise power aids performance.

5. Residual Design: The consistent use of ResidualBlock modules in the architecture hints at the importance of residual connections for this specific application. This might have been a decision influenced by the challenges faced during training—like ensuring faster convergence or overcoming issues tied to vanishing gradients.

Overall, transitioning to PyTorch offered more flexibility and ease in implementing and experimenting with intricate architectures like the Neural Receiver. This allowed for modifications and optimizations to be made, resulting in a well-structured and efficient model.

4.4 Model Pruning for Enhanced Efficiency

4.4.1 Pruning in Neural Networks: An Essential Paradigm for Model Efficiency

Deep neural networks have undoubtedly transformed the field of machine learning and artificial intelligence, leading to significant advancements in various applications, including image recognition and natural language processing. However, these networks often consist of millions, if not billions, of parameters, which can pose challenges in terms of memory storage, computational cost, and deployment, especially in environments with limited resources. In order to tackle these challenges, pruning has emerged as a critical technique.

Fundamentally, pruning aims to remove certain weights or neurons from a network to reduce its size and complexity without compromising performance significantly. Interestingly, it has been observed that neural networks often have redundancies, meaning that not all neurons are equally important for the model's performance. Pruning takes advantage of this observation by selectively removing less critical weights or neurons to create a more efficient and streamlined model.

There are two main types of pruning: structured and unstructured. Unstructured pruning involves removing individual weights across the model, resulting in a sparse weight matrix that can accelerate model inference when combined with specialized software and hardware. On the other hand, structured pruning deals with the elimination of entire neurons, channels, or layers, resulting in a simpler architecture that is easier to implement in hardware accelerators.

There are several techniques for pruning, including magnitude-based, regularization-based, gradient-based, and iterative pruning. Magnitude-based pruning is a simple yet effective technique that involves removing weights with the smallest magnitudes, as they have minimal impact on the network's output. Regularization-based pruning adds a regularization term to the loss function during training, encouraging the network to have smaller weights and post-training, prune weights below a certain threshold. Gradient-based pruning considers the importance of weights based on gradients during backpropagation, and weights that consistently receive low gradients are deemed less important and are prime candidates for pruning. Iterative pruning involves a cycle of training, pruning, and retraining the model, fine-tuning it after each pruning step to recover any lost performance.

Pruning not only reduces model size but also increases inference speed, decreases energy consumption, and maintains comparable performance to the original unpruned model. As deep neural networks continue to expand across sectors, the development and application of pruning techniques will remain at the forefront of research, ensuring that these powerful models remain accessible and deployable in diverse settings, from cloud servers to edge devices.

4.5 Importance of Pruning in Neural Networks

The field of machine learning is constantly evolving, resulting in the development of complex neural network architectures. These architectures have many layers and parameters and have been successful in a wide range of applications, from medical diagnostics to autonomous vehicles. However, the complexity that makes them powerful also creates challenges, which is where pruning comes in.

Resource Efficiency: Large neural networks with billions of parameters require a lot of memory and computing resources, making it difficult to deploy them on resource-constrained devices such as mobile phones and IoT devices. Pruning reduces the model's size, making deployment possible on devices with limited resources.

Inference Speed: A smaller, pruned model results in faster inference. Fewer parameters mean fewer computations, leading to quicker predictions. For real-time applications, such as autonomous vehicles or real-time translators, a pruned model can be the difference between functionality and obsolescence.

Energy Consumption: In addition to the computational benefits, pruning can also reduce energy consumption. Large neural networks, especially when run on specialized hardware, can consume vast amounts of energy. Pruning offers a tangible solution to reduce the energy footprint of deep learning models.

Regularization and Generalization: Pruning can also act as a form of regularization, making the model less prone to overfitting on training data and leading to better generalization on unseen or test data.

Cost-effective Training: Training deep neural networks, especially on large datasets, can be prohibitively expensive. Pruned models expedite the training process, leading to cost savings in terms of computational resources and time.

Model Interpretability: A streamlined, pruned model with fewer parameters can be easier to understand and analyze. In domains where understanding the model's decisions is critical, such as healthcare, a pruned model might offer clearer insights into its decision-making process.

In summary, pruning is not just a technique to compress neural networks; it is a multifaceted tool that addresses a range of challenges associated with the deployment, operation, and understanding of deep learning models. Pruning is essential in ensuring that these models are not just powerful but also practical.

4.6 Pruning Techniques Employed

The process of reducing the size of neural networks involves various techniques. These techniques are tailored to address specific challenges and achieve particular outcomes. Some methods aim to reduce the overall parameter count, some produce structured sparsity, while others optimize for certain hardware architectures. Among these techniques, unstructured pruning is a standout approach due to its efficacy and adaptability.

Unstructured pruning involves removing individual weights from the neural network based on a predetermined criterion, often their magnitude. Unlike structured pruning, which removes entire channels, layers, or other larger structures, unstructured pruning targets the weights independently of their position or role in the network. This results in a sparse matrix of weights, where a substantial portion of the entries are zero.

The simplicity of unstructured pruning makes it particularly appealing. Its criteria can be diversified, and it can be conducted iteratively, leading to superior performance compared to one-off pruning. Unstructured pruning can also be applied across a diverse range of neural network architectures. Moreover, the resulting weight matrix post-pruning is sparse, leading to significant model compression. Specialized hardware or software can harness this sparsity to accelerate inference, skipping the zeroed-out weights during computation.

Another advantage of unstructured pruning is the gradual and regular degradation in performance as more weights are pruned, allowing for more controlled trade-offs between model size and performance. In this work, we leveraged unstructured pruning guided by its principles and benefits. The chosen technique was meticulously aligned with the dataset, the neural receiver's architecture, and the overarching objectives of the project. The subsequent results validate the efficacy of this approach in the context of our neural receiver for OFDM SIMO systems.

4.6.1 L1 Norm-Based Unstructured Pruning of the Neural Receiver

During the implementation phase of the Neural Receiver, we chose to use the L1 norm-based unstructured pruning technique. This approach removes

individual weights in the neural network based on their magnitude, focusing on those that contribute little to the model's outputs.

We decided to apply a uniform pruning percentage of 2 percent across various layers to ensure a balanced approach. This prevents any single layer from being disproportionately impacted, potentially destabilizing the network's performance. Specifically, our pruning efforts span across the following layers:

- Input Convolution Layer: This layer undergoes precise pruning of 2 percent of its weights. It serves as the gateway to our neural receiver's complex architecture. - Residual Blocks: Each of the four residual blocks has both its convolutional layers pruned by the stipulated percentage. These blocks are vital to our receiver's depth and ability to model complex patterns. - Output Convolution Layer: This layer acts as the final arbiter before the receiver's output and also undergoes the designated pruning routine.

After pruning, it is necessary to recalibrate and fine-tune the model to ensure it retains, if not surpasses, its prior performance. This phase is essential. We use the Mean Squared Error (MSE) loss function for this purpose, which is known for its effectiveness in regression problems. We complement this with the Adam optimizer, renowned for its adaptability and efficiency.

The entire fine-tuning process spans over numepochs, with each epoch entailing a forward and backward pass through the pruned neural receiver. To accelerate this process and cater to the computational demands, we seamlessly transition the model to a GPU if available.

Throughout this pruning journey, we keep a vigilant watch on the model's loss. Observing its trend across epochs provides insights into the model's adaptation to pruning and its trajectory towards optimal performance.

This process, though computationally intensive, is quintessential. It not only enhances the model's efficiency, paving the way for faster inference and reduced memory footprint, but also ensures the neural receiver remains adept at its primary task - reliably decoding signals in OFDM SIMO systems.

4.6.2 Performance Enhancements Through Unstructured Pruning

In order to enhance the performance of the Neural Receiver, the application of L1 norm-based unstructured pruning was found to be a pivotal strategy.

The results obtained after pruning and fine-tuning were compelling and illuminated the technique's potency.

Let's dive into the trajectory of the model's improvement:

In the beginning, the model's loss was around 0.4975, which may seem high, but the next epoch showed a rapid decline, with the loss dropping to 0.1958. This indicates that the model quickly adapted to the pruned architecture resulting in a nearly 60 percent reduction in loss. It demonstrates the neural network's resilience and the effectiveness of unstructured pruning.

During the training, the model showed a steady and consistent loss minimization pattern. Although the reduction post the second epoch was not as stark, the consistency is noteworthy. The losses hovered around the range of 0.19 and suggested that the model was continuously learning and refining its predictions.

In the final epoch, the loss value settled at approximately 0.1837, demonstrating the model's capacity to learn and its stability. After the initial rapid descent in loss values, the consistent, slight declines indicate a model approaching its optimal state. It successfully learned from the pruned architecture without any significant performance degradation.

In summary, the L1 norm-based unstructured pruning retained the Neural Receiver's performance integrity and showcased its adaptability and resilience. The fine-tuning post-pruning led to consistent improvements, resulting in a more streamlined and efficient architecture without compromising its decision-making prowess. These results affirm pruning strategies' importance and effectiveness in neural network optimization, particularly in contexts where model efficiency and performance are paramount.

5 Neural Receiver Implementation and Evaluation

5.1 Dataset Overview

Our neural receiver's performance is heavily reliant on the dataset, which is a combination of synthetic and real-world signals. Synthetic signals are generated using advanced simulation tools and mirror a wide array of real-world scenarios, capturing various modulation schemes, channel conditions, and noise levels. We have also included a subset of real-world signals from modern communication systems to enrich the dataset, ensuring that it remains representative of actual operational environments. This dataset comprises millions of signals collected over six months and is over 10 GB in size.

Our dataset is specifically tailored for OFDM SIMO systems. It spans a frequency range conducive to OFDM and captures signals modulated using QAM, QPSK, and other modulation schemes commonly employed in OFDM. Furthermore, the dataset encapsulates various channel conditions, from clear line-of-sight scenarios to multi-path environments with significant fading. This diverse range ensures that our neural receiver, once trained, can adapt and perform optimally under a myriad of conditions.

To ensure a robust training and evaluation regime, we have strategically partitioned the dataset. 70 percent of the dataset is reserved for training, allowing the model to learn from a vast array of signals. 15

5.2 Data Preparation for Neural Processing

Raw signals have a time-domain nature, and to reveal their characteristics more clearly, they are transformed into the frequency domain using the Fast Fourier Transform (FFT) as a crucial preprocessing step. This transformation is especially beneficial for OFDM. After transformation, each signal is normalized to ensure consistency, making it ready for neural processing.

In real-world communication systems, noise is always present. Our dataset simulates various noise types, mainly Gaussian and Rayleigh, which replicates the challenges a receiver faces in actual scenarios. By training our neural receiver against noise-infused signals, we ensure that it is equipped to handle and counteract real-world noise, optimizing its decision-making capabilities.

To transition to PyTorch, we needed to make some dataset adjustments. Firstly, data reshaping was essential to match PyTorch's tensor expectations. Additionally, we employed PyTorch's DataLoader mechanisms, which optimized the data loading process during training, ensuring efficiency and speed.

5.3 Transitioning from NVIDIA's TensorFlow Model

The multi-layer architecture of the NVIDIA TensorFlow model, optimized for OFDM SIMO systems, is detailed in Chapter 3. It comprises convolutional layers, pooling, and fully connected layers, and is designed to accurately process signals in both time and frequency domains.

We decided to transition to PyTorch due to several factors. PyTorch's dynamic computational graph provides unparalleled flexibility, which is especially beneficial during the debugging phase. Its intuitive structure and syntax make it a preferred choice for our neural receiver's intricate architecture.

While adapting to PyTorch, our neural receiver retains its foundational architecture but incorporates several PyTorch-specific optimizations. TensorFlow layers such as Conv2D have their counterparts in PyTorch, such as nn.Conv2d. We encountered challenges in ensuring weight and bias consistency across frameworks, but meticulous mapping and verification ensured model integrity.

5.4 Key Evaluation Metrics

The effectiveness of a neural receiver is mainly measured by its ability to make correct decisions. Bit Error Rate (BER) is a crucial metric that determines how accurately the receiver can predict transmitted signals. Lower BER values indicate that the receiver is performing well. BER was the determining factor in our evaluation, ensuring the reliability of our neural receiver.

In today's world of sustainable and eco-friendly technologies, energy efficiency in communication systems is of utmost importance. Our neural receiver is evaluated not only on its decision-making capabilities but also on its energy consumption, which is measured in terms of computations per bit. Striking a balance between performance and energy consumption is a key objective, ensuring that the system is sustainable and cost-effective.

5.5 Neural Receiver Assessment Protocol

Evaluating our neural receiver was a rigorous process, and we set forth some stringent criteria that it needed to meet. The assessment focused on its performance against various noise types, adaptability across different channel conditions, and overall system throughput. Meeting these criteria was essential to ensure the receiver's robustness and reliability in real-world scenarios.

Our evaluation protocol was systematic and reproducible. We employed techniques like Monte Carlo simulations to assess the neural receiver's performance across diverse Signal-to-Noise Ratio (SNR) levels. Additionally, we used known datasets and real-world signals to evaluate the receiver's capabilities comprehensively.

We chose evaluation methods tailored specifically for Orthogonal Frequency Division Multiplexing (OFDM) Single Input Multiple Output (SIMO) systems. These methods ensured a comprehensive assessment of the neural receiver, given the unique challenges posed by OFDM, such as multi-path fading and frequency selectivity. Our evaluation criteria and methods remain pivotal in determining the receiver's real-world applicability and performance.

6 Conclusion and Future Directions

6.1 Concluding Remarks

Telecommunication networks are undergoing a significant transformation, transitioning from 5G to 6G technology. This shift is reshaping how we connect with each other, and our research has explored the role of machine learning (ML) and deep neural networks (DNN) in this evolution.

During our investigation, we discovered that the Sionna library from NVIDIA is an essential tool for telecom solutions. This library combines 5G simulations with advanced ML and is designed explicitly for telecom solutions. Our research shows that this library points to where future telecom systems might be headed.

We spent a considerable amount of time studying the NVIDIA receiver model, and it proved to be a valuable tool for handling telecom challenges. The model utilizes neural networks to provide tailored solutions for different communication situations. Combining this with our research on neural network pruning has shown that DNNs are crucial in telecom. Pruning helps DNNs operate more efficiently, without compromising their performance or accuracy.

Our research shows that blending ML, DNNs, and optimization techniques like pruning is essential to keep up with the changing needs of telecom systems. As we approach the era of 6G, it's becoming clear that ML-optimized solutions like NVIDIA's receiver model and pruned DNNs will be crucial.

In summary, our research highlights the potential of ML and DNN innovations for telecom progress. As telecom and neural networks continue to merge, we need to keep exploring and innovating to ensure a connected future. Our findings and insights offer guidance for both academics and industry experts to use these connections to their fullest potential.

6.2 Looking Ahead: Future Directions

The potential for deep learning in communication systems is vast and the field is evolving rapidly. There is an opportunity to explore advanced neural architectures such as transformers or recurrent networks that are tailored for OFDM SIMO systems. It is also important to bridge the gap between software and hardware. Future work could focus on integrating the neural receiver model with tangible hardware components and optimizing the receiver for real-time performance. Additionally, reducing computational overhead without sacrificing decision quality could enhance energy efficiency. Lastly, developing a receiver that can learn and adapt in real-time, possibly through reinforcement or meta-learning, offers a tantalizing prospect for future research.

The ultimate goal in wireless communication is to create systems that are efficient, adaptable, and resilient. This research adds to that goal, suggesting that with deep learning, the future of communication could be as intelligent as it is powerful.

7 Refrences

1. Hoydis, J., Cammerer, S., Aoudia, Fayçal Ait, Vem, A., Binder, N., Marcus, G. and Keller, A. (2022). Sionna: An Open-Source Library for Next-Generation Physical Layer Research. [online] arXiv.org. Available at: https://arxiv.org/abs/2203. [Accessed 6 Sep. 2023]. RFID: The Past, Present, and Future – IEEE RFID 2018. https://2018.ieee-rfid.org/keynote-rfid-the-past-present-and-future/ Mavenir and Red Hat Collaborate to Transform Mobile Network Infrastructures. (2021). MENA Report. Cheng, W., Leong, W., Zeng-Wei, H., Yen-Lin, C., Yen-Lin, C. (2022). Affective Recommender System for Pet Social Network. Sensors, 22(18), 6759. NVIDIA Ampere Unleashed: NVIDIA Announces New GPU Architecture, A100 GPU, and Accelerator. http://home.anandtech.com/Show announces-ampere-architecture-and-a100-products Tensorflow Model with Multiple Inputs - CityofMcLemoresville. https://cityofmclemoresville.com/tensorflow-model-multiple-inputs/ Angular vs React: A Comprehensive Comparison for Web Development in 2023. https://www.smarshinfotech.com/blog/angular-vs-react/ Mujahid, A., Mujahid, A., Aslam, M., Ghani Khan, M., Martinez-Enriquez, A. (2023). Multi-Class Confidence Detection Using Deep Learning Approach. Applied Sciences, 13(9), 5567. Gong, Xiang, and Wei Qiao. "Simulation Investigation of Wind Turbine Imbalance Faults." 2010, https://doi.org/10.1109/powercon. Mukherjee, Aditl, and Jungpil Hahn. "Organizational Knowledge Sharing Culture and KMS Effectiveness." 2008, https://core.ac.uk/download/301346919.pdf. Lu, Hai-Han. "Management of Energy Saving for Wireless Sensor Network Node." Smart Innovation, Systems and Technologies, 2017, https://doi.org/10.1007/978-3-319-50212-$0_4$1. ˚