CSI Feedback Enhancement using Machine Learning

Muhammad Karam Shehzad

université PARIS-SACLAY

CSI Feedback Enhancement using Machine Learning

Amélioration du retour d'information des CSI à l'aide de l'apprentissage automatique

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 580, Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat: Sciences des réseaux, de l'information et de la communication
Graduate School : Sciences de l'Ingénierie et des Systèmes
Référent: CentraleSupélec

Thèse préparée au **Laboratoire des signaux et systèmes (Université Paris-Saclay, CNRS, CentraleSupélec)**, sous la direction de **Mohamad ASSAAD**, Professor, CentraleSupélec, France, et la co-direction de **Luca ROSE**, Senior Research and Standardization Specialist, Nokia, France

Thèse soutenue à Paris-Saclay, le 12 Décembre 2023, par

Muhammad Karam SHEHZAD

Composition du jury

Membres du jury avec voix délibérative

Mérouane DEBBAH Professor, Khalifa University, Abu Dhabi, United Arab Emirates	Président
Carles Navarro MANCHON Associate Professor, Aalborg University, Aalborg, Denmark	Examinateur
E. Veronica BELMEGA Professor, Université Gustave Eiffel (ESIEE Paris) and LIGM research lab, Marne-la-Valée, France	Rapporteure & Examinatrice
Alessio ZAPPONE Professor, University of Cassino and Southern Lazio, Cassino, Italy	Rapporteur & Examinateur

THESE DE DOCTORAT
NNT : 2023UPAST213

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre:Amélioration du retour d'information des CSI à l'aide de l'apprentissage automatique

Mots clés: Intelligence Artificielle, Prédiction de Canal, Rétroaction CSI, Apprentissage Automatique, Massive MIMO, RIS, Planification des Utilisateurs, 3GPP, 6G

Résumé: Acquérir les information d'état du canal est indispensable dans un réseau cellulaire. Dans les protocoles de communication actuels, le CSI en liaison descendante est estimé par l'User Equipment (l'équipement utilisateur - UE) via des pilotes dédiés puis renvoyés à la station de base. L'information retour du CSI nécessite la transmission d'une importante quantité de données supplémentaire Over-The-Air (OTA) afin d'améliorer la qualité du CSI acquis au niveau de la station de base. La compression est l'une des solution utilisées dans les standards, mais elle dégrade les performances d'acquisition du CSI. Des solutions alternatives telles que l'intelligence artificielle (AI) et l'apprentissage automatique (ML) sont apparues récemment. Ces techniques sont prometteuses pour palier aux problèmes dus aux déficits d'algorithmes et de modèles.

Cette thèse de doctorat propose d'utiliser les techniques AI/ML pour améliorer les performances du retour d'information CSI. La première partie de la thèse développe un nouveau cadre ML, baptisé CSIFB-PNet, qui exploite des prédicteurs à deux canaux pour améliorer la rétroaction CSI. En utilisant le mécanisme de retour CSI conventionnel, les modèles ML sont entrainés aux deux extrémités (BS et UE) de la chaine de communication. De façon plus générale, le UE évalue le retour CSI généré par le modèle ML. La solution proposée permet de réduire la quantité d'information nécessaire à la rétroaction du CSI tout en améliorant sa précision. Cette solution est ensuite étendue à l'entraînement unilatéral du modèle ML pour améliorer davantage les performances. Le modèle est généré au niveau du UE puis est partagé avec la station de base. Les résultats observés confirment la validité de l'entraînement unilatéral du modèle ML. Nous aborderons également divers algorithmes ML pour la prédiction des canaux, car elle joue un rôle central dans le travail proposé. Les résultats sont évalués à l'aide des données réelles enregistrées sur le campus de Nokia Bell-Labs à Stuttgart, en Allemagne.

La mise en œuvre de modèle ML au niveau du UE peut être impossible pour diverses raisons, en particulier la consommation d'énergie. Motivés par ce problème, nous proposons un mécanisme d'apprentissage CSI au niveau de la station de base, appelé CSILaBS, pour pallier à l'utilisation de modèles ML au niveau du UE. La station de base calcule une fonction prédictive légère (PF) en exploitant les modèles ML, pour aider l'UE à évaluer la rétroaction. Ce processus permet de réduire la quantité de données supplémentaires OTA tout en minimisant la complexité au niveau du UE. Pour aller plus loin, dans un environnement multi-utilisateurs, nous avons proposé plusieurs solutions afin de planifier le retour CSI en utilisant des fonctions prédictives légères tout en visant à améliorer sa précision. Les résultats de simulation obtenus montrent l'efficacité de la planification du retour et que le seuil d'erreur de prédiction pour la boucle retour de planification est un paramètre de conception important qui doit être affiné pour maximiser la précision du CSI.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre:Amélioration du retour d'information des CSI à l'aide de l'apprentissage automatique

Mots clés: Intelligence Artificielle, Prédiction de Canal, Rétroaction CSI, Apprentissage Automatique, Massive MIMO, RIS, Planification des Utilisateurs, 3GPP, 6G

Résumé:

Dans la deuxième partie de la thèse, nous exploitons les surfaces intelligentes reconfigurables (RIS) pour améliorer les performances d'un prédicteur de canal, qui est la composante principale des schémas d'amélioration de retour du CSI. La prédiction d'un canal très dynamique est une tâche difficile. Pour résoudre ce problème, nous avons utilisé l'amélioration des performances de prédiction de canaux multi-utilisateurs en introduisant une corrélation temporelle dans le canal composite. L'objectif a été atteint en exploitant le quasi-RIS. Plus précisément, la motivation de notre travail est de prédire l'ensemble du canal composite multi-utilisateurs à des moments temporels bien définis en utilisant quasi-RIS. Le principal avantage est la prédiction de l'ensemble du canal composite provenant de l'ensemble du RIS. De ce fait, la complexité et le nombre de données d'information supplémentaires nécessaire pour l'estimation des canaux peuvent être réduits, ce qui es en principe le problème majeur lié au RIS. Grâce à de nombreuses simulations approfondies, nous montrons que le RIS intelligemment configuré améliore les performances de prédiction des canaux tout en économisant suffisamment d'éléments RIS pour d'autres applications.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Title: CSI Feedback Enhancement using Machine Learning

Keywords: Artificial Intelligence, Channel Prediction, CSI Feedback, Machine Learning, Massive MIMO, RIS, User Scheduling, 3GPP, 6G

Abstract: Acquisition of channel state information (CSI) is indispensable in a cellular network. In the current communication architecture, the downlink CSI is estimated by the user equipment (UE) via dedicated pilots and then fed back to the base station (BS). CSI feedback requires immense over-the-air (OTA) overhead to improve the quality of CSI acquired at the BS. Compression is one of the solutions used in the standards; however, it degrades the performance of CSI acquisition. Alternative solutions to compression may be found in Artificial Intelligence (AI) and Machine Learning (ML), which have emerged as promising techniques to bring benefits to algorithm-deficit and model-deficit problems.

This thesis utilizes AI/ML to enhance the performance of CSI feedback. The first part of the thesis develops a novel ML framework, coined as CSIFB-PNet, which exploits twin channel predictors to enhance CSI feedback. By using conventional CSI feedback mechanism, the ML models are trained at both ends (BS and UE). Broadly, UE evaluates the feedback with respect to ML model. The proposed framework can reduce OTA feedback overhead and improve the precision of acquired CSI. This framework is then extended towards one-sided training of the ML model to further enhance the performance, using UE to train the model and then reporting it to BS. The observed results corroborate the validity of one-sided ML training. We also address various ML-based algorithms for channel prediction, as it plays a pivotal role in the proposed framework. This framework is evaluated using the real-world data recorded at the Nokia Bell-Labs campus in Stuttgart, Germany.

ML implementation at the UE can be infeasible for various reasons, in particular UE power consumption. Motivated by this issue, we propose a CSI learning mechanism at BS, called CSILaBS, to avoid ML at UE. BS computes a light-weight predictor function (PF) by exploiting ML, to assist the UE to evaluate feedback. This process can reduce OTA feedback overhead, and minimize UE computation cost. Besides, in a multiuser environment, we propose various mechanisms to select the feedback by exploiting PF while aiming to enhance CSI accuracy. Simulation results show the effectiveness of CSI feedback schemes and that the prediction error threshold for CSI feedback selection is an important design parameter that can be fine-tuned to maximize CSI accuracy.

In the second part of the thesis, we exploit reconfigurable intelligent surface (RIS) to enhance the performance of a channel predictor, which is the core component of proposed CSI feedback enhancement schemes. Prediction of a highly dynamic channel is cumbersome. To mitigate this issue, we focus on enhancing multiuser channel prediction performance by introducing time correlation in the composite channel. The objective is achieved by exploiting quasi-RIS. More specifically, the motivation of our work is to predict the entire multiuser composite channel for desired time slots by using quasi-RIS. The primary benefit is the prediction of the whole composite channel arriving from the entire RIS; hence, the complexity and overhead for channel estimation can be reduced, which is the major bottleneck in RIS. Through extensive simulations, we show that intelligently configured RIS enhances channel prediction performance while saving sufficient RIS elements for other applications.

Acknowledgments

I would like to express my heartfelt gratitude to all who stood by my side through thick and thin during my thesis. I want to express my profound appreciation to my thesis supervisor **Luca Rose**, senior research and standardization specialist at Nokia, Paris. He helped me by all means to make this thesis a success. He has been very supportive at every step; his timely feedback and encouraging comments enabled me to complete this thesis on stroke of time. He has shown his full determination to advise on selecting the thesis topic and contributing to the patents and research publications. This journey had had ups and downs, but his encouragement and motivation kept me afloat amidst all trials and tribulations. I am also utmost grateful for his support to help me settle in Paris city as it was not easy to arrive and settle in the peak time of Covid pandemic. By working with him, I started showing a keen interest in standardization activities, and he guided me on how to hone my skills in my chosen field. Above all, he has helped me shape my career, leading me to pursue my career at the world's leading telecommunication organization.

This endeavor would not have been possible without my thesis director **Mohamad Assaad**, who is a full professor at CentraleSupélec, Paris. I am deeply indebted to his unstinted support and encouragement. I still remember the day of 12^{th} October, 2020, when I met him the first time; I was a bit nervous but his positivity helped me to establish a cordial relation with him. His motivating ideas on the thesis topic gave me extra energy to consolidate thesis. The technical discussions with him helped me to polish my research work and follow the latest trends in my field. His door was always open for me to discuss the challenges I had faced in the different stages of my PhD. His unwavering support kept me motivated to contribute further to my PhD. Also, he had been very supportive of management-related issues that I faced at the beginning of my PhD work.

I would also like to sincerely thank the thesis evaluation committee, composed of Professors **Mérouane DEBBAH**, **Carles Navarro MANCHON**, **E. Veronica BELMEGA**, and **Alessio ZAPPONE**, for accepting the invitation to serve as jury. I am grateful for their time and effort spent reviewing the thesis work.

I would like to extend my sincere thanks to my colleagues at Nokia Bell-Labs, with whom I spent three productive years of my PhD. I got the opportunity to exchange views on different topics and get feedback on my thesis, which helped me to improve my technical skills. Especially, I am grateful to **Philippe Sehier**, who reviewed my papers and gave his constructive criticism to enhance the quality. I still remember his warm welcome in a face-to-face meeting when the Covid pandemic was at its peak. Throughout my stay at Nokia Bell-Labs, his continuous support acted as a catalyst in my progress. His friendly behaviour will always be remembered. I am also grateful to **Amitava Ghosh**, who took my final interview and considered my hiring as a PhD student and where the journey began. Being such a big name in wireless communications, I was very nervous about being interviewed by him, but his gentle behaviour and constructive remarks on my pre-PhD work helped me to excel during PhD. Among others, I want to thank **M. Majid Butt** for his continuous support and guidance towards research-related activities in particular and for shaping my future in general. I am always in debt to him for his gentle behaviour and teaching me all those things which are of pivotal importance for a researcher. I will not forget the days when sometimes I used to go to the office in your car and all the interesting discussions, which catalysed my PhD journey. Taking the opportunity, I would like to thank all my team members at Nokia Bell-Labs, with whom I had fruitful discussions over coffee and lunch breaks.

I would be remiss in not mentioning my colleagues, who have been on the same journey to earn a PhD, for keeping me motivated throughout this adventure. I will never forget the cherishing moments that I had with you. Among others, I would sincerely thank **Shashwat**, **Abdelmounaim**, and **Soumeya**, with whom I had spent the days of my PhD at Nokia. Besides,

my fellows at CentraleSupélec, **Elisa**, **Amina**, **Wasif**, **Safaa**, **Zalfa**, **Ansar**, **Abelhak**, **Asmaa**, **Mahdi**, and many others, I am grateful to all of you for your support and friendship. Taking the opportunity, I would like to thank all the people from Nokia and CentraleSupélec who helped me by any and every means in the journey of achieving this milestone. Also, I would like to sincerely thank my colleague **Syed Muhammad Fahad**, who has been very supportive throughout my PhD. His unwavering support in settling in Paris will never be forgotten. He was always there whenever I faced any issue. Your gentle behaviour has set an example for me to excel personally and professionally.

There are many people who are always there for you, even being miles away. Yes, I am in debt to all my friends, especially my family members. Because of them, I got the opportunity to make my dreams come true. They have supported me through the whole turbulence I underwent. I would like to thank all my friends with whom I studied at Namal, NUST, and other institutes. Besides, I pay gratitude to all the people who helped me in the early days of my educational career and during the PhD, especially **Atiq** and **Rameez** because of whom I pursued higher education. While discussing academic life, I will never forget **Wajid Ashraf**; you have been an inspiration for all the students, particularly me. Words cannot express my gratitude. I will always remain in debt to you for your efforts to see me excel in my career and life in particular. There are many unforgettable days, but I will always remember when you told me about Namal and helped me pursue my life at Namal, which was a turning point. While mentioning Namal, I would like to thank **Imran Khan**, who built that institute for a social cause where students like me can shape their future. Had Namal not supported me when I was searching to start my Bachelor's degree, I would not have been writing this thesis, as the rest of the doors seemed closed then. I could not have undertaken this journey without his efforts in making Namal a great success. Taking the opportunity, I would like to thank all my teachers who taught me at Namal, NUST, and all other institutes. Thank you very much for your continuous support. I would like to thank my brother **Khurram**, my sisters **Iram** and **Kiran**, and all friends, **Saad**, **Hamza**, **Cheikh**, **Faizan**, **Waheed**, **Asaad**, **Youssef**, **Abdellah**, **Iram**, **Faiqa**, **Abbirah**, **Paul**, **Issam**, **Laila**, for your unwavering support in the journey of my PhD. Besides, I am grateful to all the teachers and friends from high school, Namal, NUST, University of Málaga, and University of Bradford. There is much more to appreciate, but words cannot express my gratitude. I conclude my acknowledgement with the quote: *"Knowledge is in the end based on Acknowledgement"*.

Contents

List of Figures

List of Tables

Acronyms

3GPP Third generation partnership project. 11

5G fifth-generation. 8

5GC 5G core. 11

6G sixth-generation. 8

ACK acknowledgment. 54

AF application function. 11

AI Artificial Intelligence. 8

AR autoregressive. 29

BiLSTM bidirectional long short-term memory. 23

BS base station. 11

CP channel predictor. 18

CQI channel quality indicator. 13

CS compressive sensing. 14

CSI channel state information. 11

CSI-RS CSI reference symbols. 13

CSIFB-PNet CSI feedback prediction network. 18

CSILaBS CSI learning at BS. 18

CsiNet Autoencoder-based CSI network. 15

DL deep learning. 14

FLOPs floating-point operations per second. 16

GAN generative adversial network. 16

LoS line-of-sight. 11

LSTM long short-term memory. 34

LTE long-term evolution. 8

MAC medium access control. 57

MIMO multiple-input multiple-output. 10

ML Machine Learning. 8

MSE mean-squared error. 39

NACK negative acknowledgement. 54

NF network function. 11

NLoS non-line-of-sight. 14

NMSE normalized mean-squared errorr. 43

NN neural network. 16

NP NeuralProphet. 23

NR new radio. 8

NWDAF network data analytics function. 11

OAM operation, administration, and maintenance. 11

OFDM orthogonal frequency-division multiplexing. 23

OTA over-the-air. 13

PF predictor function. 18

PHY physical. 9

PMI precoding matrix indicator. 13

RAN radio access network. 11

RI rank indicator. 13

RIS reconfigurable intelligent surfaces. 18

RNN recurrent neural network. 23

SINR signal-to-interference-plus-noise ratio. 13

SISO single-input single-output. 12, IV

THz TeraHertz. 8

UE user equipment. 12

VAE variational autoencoder. 16

Résumé long en français

Globalement , le terme CSI (Channel State Information) est un terme utilisé pour représenter la connaissance d'un canal de propagation . Qu'est-ce qu'un canal ? D'une manière simplifié , tout system de communication sans fil est composé de trois éléments : le transmetteur , le canal et le récepteur . Le transmetteur est une entité du réseau , également appelé émetteur , qui génère le message et le transmet au récepteur . Le message transmis se propage à travers le support , appelé canal de propagation . Le canal est sensible à de nombreux obstacles qui peuvent entraver la communication . Par conséquent , la connaissance du canal avant de transmettre les données est indispensable .

Le CSI fournit des informations sur la façon dont le message transmis se propage entre l'émetteur et le récepteur. La connaissance du CSI est d'une importance capitale pour plusieurs raisons . Nous allons développer ce point à l'aide d'un exemple . Considérons un récepteur UE (User Equipment) qui se déplace dans un environnement stationnaire , voir Fig. 1 . La puissance du signal reçu par l'UE fluctue suivant sa position dans l'environnement . Si l'on imagine qu'un émetteur , tel qu'une station de base BS , communique en permanence avec l'UE qui se trouve dans un évanouissement profond du canal , la probabilité de perte de données est élevée car le gain du canal est très faible . Par contre , si la station de base reçoit des informations au moment où ces évanouissements du canal se produisent , les performances de la station de base peuvent être nettement améliorées . Par exemple , dans le cas d'un évanouissement profond au niveau d'un UE , la station de base peut programmer la communication avec un autre UE dont le gain du canal est élevé . Par conséquent , l'efficacité spectrale globale du système peut être maximisée .

Pour améliorer les performances d'un système de communication , de nombreuses techniques de transmission , telles que la taille de la charge en bits , le codage , les méthodes de précodage , la modulation adaptative , la planification en fonction du canal , etc. , dépendent de la connaissance précise du CSI au niveau de l'émetteur . Un CSI précis améliore nettement les performances de nombreuses technologies sans fil , telles que le MIMO (Multiple Input Multiple Output) , les transmissions ultra-fiables , le relais et la sécurité de la couche physique . Par conséquent , la connaissance du CSI au niveau

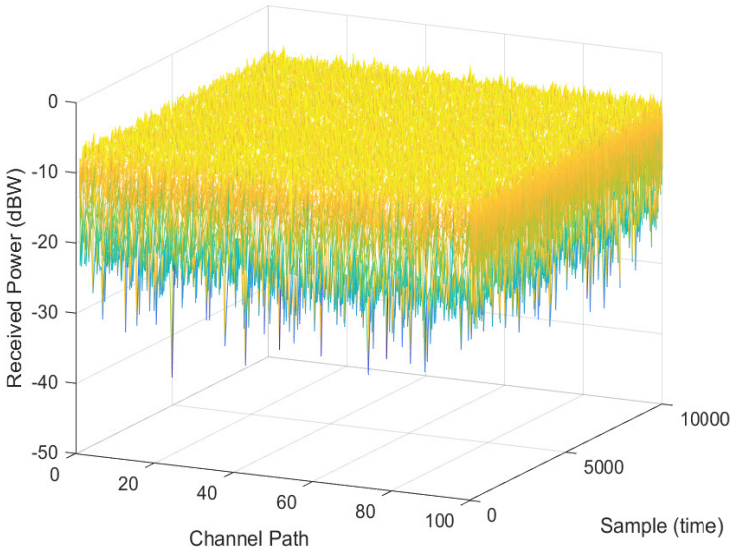


Figure 1: Illustration d'un modèle de canal stationaire à ligne à retard enregistrée (TDL:Tapped Delay Line) pour une entrée et sortie unique (SISO) canal .

de l'émetteur est d'une importance capitale . La question clé réside dans la manière dont l'émetteur peut obtenir l'information sur le canal , c'est-à-dire le CSI , avant de transmettre les données . Nous aborderons cette question dans la sous-section suivante .

Afin d'acquérir le CSI sur la liaison descendante au niveau de la station de base , deux étapes sont nécessaires . Dans la première étape , l'UE estime le CSI à l'aide des pilotes , également appelés symboles de référence ou CSI-RS (CSI-Reference Symbols) , transmis par la station de base . Dans la deuxième étape , le CSI estimé est renvoyé à la station de base via le canal de contrôle de la liaison montante . L'estimation du CSI n'est pas au cœur de cette thèse . Par conséquent , en supposant que l'estimation du CSI est parfaite , nous nous concentrons sur le renvoi du CSI à la station de base .

Qu'est-ce que le retour d'information CSI ?

Le concept de renvoi du CSI de la liaison descendante à la station de base est largement connu sous le nom de feedback CSI . Le retour CSI standard se compose de trois parties : l'indicateur de matrice de précodage PMI (Precoding Matrix Indicator) , l'indicateur de qualité du canal CQI (Channel Quality Indicator) et l'indicateur de rang RI (Rank Indicator) . L'indicateur de matrice de précodage est le plus crucial , car il aide la station de base à sélectionner un faisceau approprié . En résumé , le CSI-RS offrant le meilleur rapport signal-sur-interférence-plus-bruit (SINR) est choisi , puis le PMI correspondant au livre de codes est transmis à la station de base simultanément au CQI et au RI . Pour réduire la charge associée aux communications par voie hertzienne

(over-the-air (OTA) overhead) , le CSI estimé est compressé, c'est-à-dire que seul le PMI est renvoyé . Par conséquent , le CSI acquis par la station de base est sujet à des erreurs de compression . Une telle compression peut altérer les performances du précodeur MIMO massif .

Objectif de la Thèse et Énoncé du Problèm

Actuellement, le ML (Machine Learning) est adopté dans les couches supérieures , telles que la couche réseau de la norme 5G , tandis qu'au niveau de la couche PHY , il est considéré comme un outil d'optimisation , donc utilisé pour la mise en œuvre , mais n'est pas intégré dans la norme . Cette thèse vise à identifier les domaines spécifiques du standard qui pourraient bénéficier de l'utilisation du ML , ainsi que la manière dont le ML peut être appliqué . L'objectif de cette thèse est d'élargir l'application du ML en exploitant les outils utilisables au niveau de la couche PHY des réseaux 5G pour améliorer les performances des systèmes de communication . Ces algorithmes devraient fournir des méthodes normalisées pour réduire la charge de retour du CSI et améliorer la qualité du CSI .

Dans les réseaux sans fil modernes , les degrés de liberté du système de communication (à savoir le temps, la fréquence et l'espace) sont soutenus par *l'intelligence* , laquelle est considérée comme une ressource supplémentaire disponible et exploitée pour améliorer l'efficacité des communications . Dans le cadre du 3GPP , les efforts de normalisation ont principalement associé le ML aux couches supérieures , comme le réseau , laissant au ML un rôle de détail pour la mise en œuvre dans la couche PHY . Ainsi , dans les réseaux 5G , de nombreux aspects fonctionnels sont régis par des règles prédéterminées , qui ne sont efficaces que si les hypothèses du modèle correspondent précisément à l'environnement naturel . Cependant , une grande quantité de données spécifiques , telles que le CSI , peuvent être exploitées par des mécanismes intelligents pour s'adapter plus efficacement à l'environnement .

En mettant l'accent sur le retour d'information CSI , la problématique de la réduction de la surcharge du retour d'information OTA reste à ce jour non résolue . Bien que des méthodes standard basées sur un livre de codes aient été adoptées en tant que solution , la précision du CSI est fortement impactée par la compression . De plus , les livres de codes sont conçus en supposant une distribution spécifique des canaux . Or , cette hypothèse sous-jacente peut différer de la réalité pratique , entraînant une diminution des performances . Il est donc impératif d'introduire des algorithmes adaptatifs pour contourner les problèmes rencontrés par les méthodes conventionnelles . Dans cette thèse , notre objectif est de résoudre la problématique suivante : *Comment réduire la surcharge du retour CSI tout en améliorant la précision du CSI acquis au niveau de la station de base ?*

Plan de la Thèse et Contributions

La thèse est structurée en cinq chapitres , chacun apportant des contributions spécifiques :

Chapitre 1 (Introduction) , ce chapitre offre une vue d'ensemble de l'intelligence artificielle et de l'apprentissage machine (IA/ML) ainsi que leur rôle dans les communications sans fil . Il aborde les défis majeurs liés au déploiement de l'IA dans ce domaine . Le rôle crucial du CSI (Channel State Information) dans les communications sans fil est mis en lumière , en discutant des travaux traditionnels et de leurs limitations . Enfin , il présente les objectifs de la thèse et l'organisation du manuscrit .

Chapitre 2 (Amélioration du retour du CSI dans le cas d'un seul utilisateur) , ce chapitre se concentre sur la réduction de la charge du retour du CSI en utilisant un prédicteur de canal double CP (Channel Predictor) basé sur le ML , nommé CSIFB-PNet (CSI feedback prediction network) . L'entraînement du ML est réalisé simultanément au niveau de la station de base et de l'utilisateur . En utilisant ces prédicteurs , le retour du CSI est évalué par rapport au canal prédit à l'UE . Par exemple , une prédiction et une estimation parfaites du CSI peuvent éviter un renvoi inutile du CSI . Ce chapitre explore également l'entraînement ML unilatéral , où le ML est entraîné au niveau de l'UE , et le modèle appris est transmis à la station de base pour des prédictions à double extrémité . Les résultats empiriques mettent en évidence : (i) la réduction des erreurs dans le CSI acquis par la station de base ; (ii) la maximisation du gain de précodage ; et (iii) la maximisation de la similarité cosinusoidale avec une surcharge de rétroaction minimale . Enfin , différentes techniques de prédiction du CSI basées sur le ML sont examinées , confirmant l'utilité du ML pour améliorer le retour du CSI tout en minimisant la surcharge OTA .

Le Chapitre 3 (Amélioration du retour du CSI dans un environnement multiutilisateur) Dans un contexte multiutilisateur, ce chapitre étudie les performances du ML dans le Chapitre 2, envisageant son application à la fois au niveau de la station de base et du terminal utilisateur UE pour améliorer la précision du CSI acquis . Cependant , l'implémentation du ML au niveau de l'UE peut être difficile pour diverses raisons , telles que la consommation d'énergie élevée de l'UE et les contraintes de stockage liées à un réseau de neurones lourd . Pour résoudre ce problème , nous proposons une méthode alternative appelée "l'apprentissage du CSI à la station de base" (CSILaBS) . Cette approche vise à éviter le déploiement du ML au niveau de l'UE . En exploitant le prédicteur de canal à la station de base , nous envisageons une fonction prédicteur (PF) légère pour évaluer le retour d'information destiné à l'UE . Cette approche vise à réduire la charge de retour d'information OTA tout en améliorant la qualité du CSI acquis par la station de base . De plus , différents mécanismes sont proposés pour sélectionner le retour d'information au niveau de l'utilisateur , exploitant la fonction prédictive pour améliorer la précision du CSI sans nécessiter l'utilisation intensive de ressources au niveau de l'UE .

Le Chapitre 4 (Rôle des surface intelligente RIS dans la prédiction du CSI) focalise sur l'intégration des surfaces intelligentes réfléchissantes (RIS) dans le processus

de retour des Channel State Information (CSI) . S'appuyant sur les avantages prometteurs du prédicteur de canal (CP) dans l'amélioration du retour du CSI , ce chapitre explore l'utilisation des RIS pour renforcer la prédiction des CSI . Étant donné que la prédiction d'un canal hautement dynamique peut être complexe , cette section se concentre sur l'amélioration de la performance de prédiction du CSI multiutilisateur en introduisant la corrélation temporelle dans le canal composite . Plus spécifiquement , nous examinons l'utilisation d'un quasi-RIS pour atteindre cet objectif . Nos recherches démontrent que l'exploitation intelligente d'un RIS améliore considérablement les capacités de prédiction du canal tout en conservant des éléments du RIS pour d'autres applications . Un avantage supplémentaire réside dans la prédiction de l'ensemble du canal composite du RIS . Cette approche permet de réduire la complexité et la charge liées à l'estimation du canal , éléments souvent contraignants pour les RIS .

Le Chapitre 5 (Conclusion et perspectives d'avenir) le dernier chapitre de cette thèse conclut l'étude en mettant en avant les perspectives futures prometteuses , offrant des pistes pour une continuation des recherches proposées . De plus , une solution proche des standards est présentée pour aborder la déclaration et la vérification des données d'entraînement relatives au Channel State Information (CSI) . Il convient de souligner que chaque chapitre comporte ses propres notations mathématiques ainsi que son modèle de système spécifique .

Brevets et Publications

Les brevets suivants ont été déposés au cours de la thèse. De plus, les publications de recherche produites dans le cadre des travaux de thèse sont fournies.

Brevets

Vous trouverez ci-dessous la liste des seuls brevets déposés. Le reste n'est pas fourni pour des raisons de confidentialité.

- M. K. Shehzad, F. Jardel, L.Rose, and M. M. Butt "Data augmentation for CSI feedback", NC326631, 2022.
- L. Rose, and M. K. Shehzad, "CSI feedback using channel prediction: Exploiting UE's capability", NC328095, 2023.
- L. Rose, and M. K. Shehzad, "Reduced overhead and estimation error to report CSI in wireless networks", NC321108, 2020.
- L. Rose, and M. K. Shehzad, "Shifting AI/ML to BS for CSI feedback enhancement", NC326027, 2022.

Publications

Les travaux de cette thèse ont donné lieu aux publications suivantes:

Article de journaux publié

- M. K. Shehzad, L. Rose, M. M. Butt, I. Z. Kovács, M. Assaad, and M. Guizani, "Artificial intelligence for 6G networks: Technology advancement and standardization," *IEEE Vehicular Technology Magazine*, vol. 17, no. 3, pp. 16–25, 2022.
- M. K. Shehzad, L. Rose, S. Wesemann, and M. Assaad, "ML-based massive MIMO channel prediction: Does it work on real-world data?" *IEEE Wireless Communications Letters*, vol. 11, no. 4, pp. 811–815, 2022.
- M. K. Shehzad, M. Assaad, and L. Rose, "Reconfigurable intelligent surfaces: How to enhance multiuser channel prediction?" *IEEE Wireless Communications Letters*, pp. 1–1, 2023.

Articles de conférences publiés

- M. K. Shehzad, L. Rose, and M. Assaad, "Dealing with CSI compression to reduce losses and overhead: An artificial intelligence approach," in *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2021, pp. 1–6.
- M. K. Shehzad, L. Rose, and M. Assaad, "A novel algorithm to report CSI in MIMO-based wireless networks," in *ICC 2021 - IEEE International Conference on Communications*, 2021, pp. 1–6.

- M. K. Shehzad, L. Rose, and M. Assaad, "RNN-based twin channel predictors for CSI acquisition in UAV-assisted 5G+ networks," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.

- M. K. Shehzad, L. Rose, M. F. Azam, and M. Assaad, "Real-time massive MIMO channel prediction: A combination of deep learning and NeuralProphet," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, 2022, pp. 1423–1428.

Articles de journaux soumis

- M. K. Shehzad, L. Rose, S. Wesemann, M. Assaad, and S. A. Hassan, "Design of an efficient CSI feedback mechanism in massive MIMO systems: A machine learning approach using empirical data," *in IEEE Transactions on Cognitive Communications and Networking (TCCN)*, 2023.

- M. K. Shehzad, L. Rose, and M. Assaad, "Massive MIMO CSI feedback using channel prediction: How to avoid machine learning at UE?," *in IEEE Transactions on Wireless Communications (TWC)*, 2023.

Chapter 1

Introduction

1.1 Background and Motivation

The wireless industry has revolutionized almost every sector, among which mobile communications has brought tremendous achievements. The unabated growth of mobile users and enormous data rate demands have already introduced the world to five evolutionary generations of mobile networks. Recently, fast-growing deployment of fifth-generation (5G) has opened up many challenges, including massive complexity in network architecture, low latency, high cost, power consumption, and deployment of long-term evolution (LTE) new radio (NR), leading to difficulties in network optimization. In such a complex scenario, network intelligence has become a major focus as it will play a pivotal role in complex problem solving [1], e.g., self-healing, self-optimization, and self-configuration of a network [10]. On the other hand, unprecedented growth in global cellular traffic (as sown in Fig. 1.1) have become a challenge, leading the wireless industry to the next generation, called sixth-generation (6G).

With the deployment of 5G networks, standards organizations have started working on the design phase for 6G networks. 6G-era will bring digital, physical and biological worlds together with the goal to improve human experience and well-being. 6G will be operating in TeraHertz (THz) frequencies (0.1-$10\,\text{THz}$), hence beneficial for multiple use cases in industrial applications, providing immense data rates ($\approx 1\,\text{Tb/s}$), accelerating internet-of-things, and wider network coverage. However, 6G networks will be immensely complex, requiring more deployment time, cost and management efforts. On the other side, mobile network operators demand these networks to be intelligent, self-organizing, and cost-effective to reduce operating expenses. Today's networks use model-based methods to optimize various network functions providing characteristics of the process involved. However, these models might be too complex to be implemented in a realistic time frame or they include a great level of abstraction to function in a general environment.

To circumvent the challenges that will be faced by future cellular networks, Artificial Intelligence (AI)/Machine Learning (ML) is the answer providing pragmatic so-

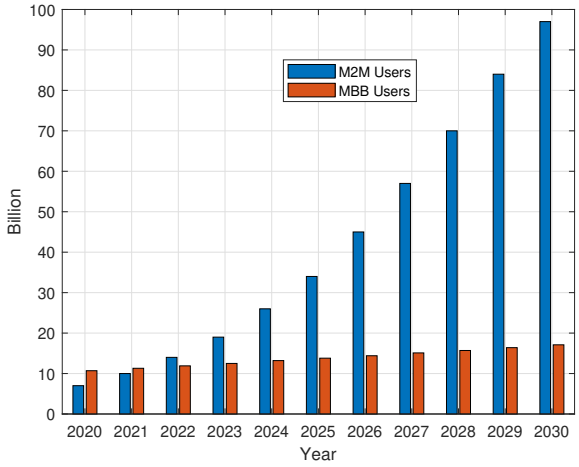


Figure 1.1: Estimation of global mobile subscriptions in machine-to-machine (M2M) and mobile broadband (MBB) from 2020 to 2030. Source: International Telecommunications Union Radiocommunications Sector (ITU-R) Report M.2370-0 [1], [11].

lutions. AI/ML can entirely change the future of wireless network technologies [1]. Future networks will become "cognitive" in a way that many aspects such as spectrum sensing/sharing, slicing, radio resource management, and mobility management, will be ML-based. Further, it is expected that ML will impact 6G air interface fundamentally and it will be designed to support ML natively [12]. ML will pave the way for THz communications at different layers [13], e.g., supporting channel acquisition [6] and modulation classification [14] at the physical (PHY) layer. Similarly, at the link layer, beamforming design and channel allocation can exploit ML [13]. In THz systems, a channel can significantly vary at a micrometer scale, resulting in a tremendous increase in channel estimation frequency and corresponding overhead. ML algorithms can counter this issue by using, e.g., improved channel prediction techniques [2].

1.1.1 Why ML in Communication Systems?

ML is a process of training machines through data without explicit programming. Broadly speaking, ML consists of three paradigms: unsupervised learning, supervised learning, and reinforcement learning. All these paradigms have a *training/exploration phase* to optimize a learning algorithm that later can be used in *prediction/exploitation phase* to infer on unknown inputs.

Motivated by the considerable benefits of ML in various fields, its applications have also been considered in wireless networks almost at all layers of communication. Focusing on PHY, many optimization problems are non-convex, e.g., sum-rate maximization. ML is a powerful tool to find solution(s) for such non-convex optimization problems. Based on advanced learning algorithms, 6G networks provide following major advantages by using ML.

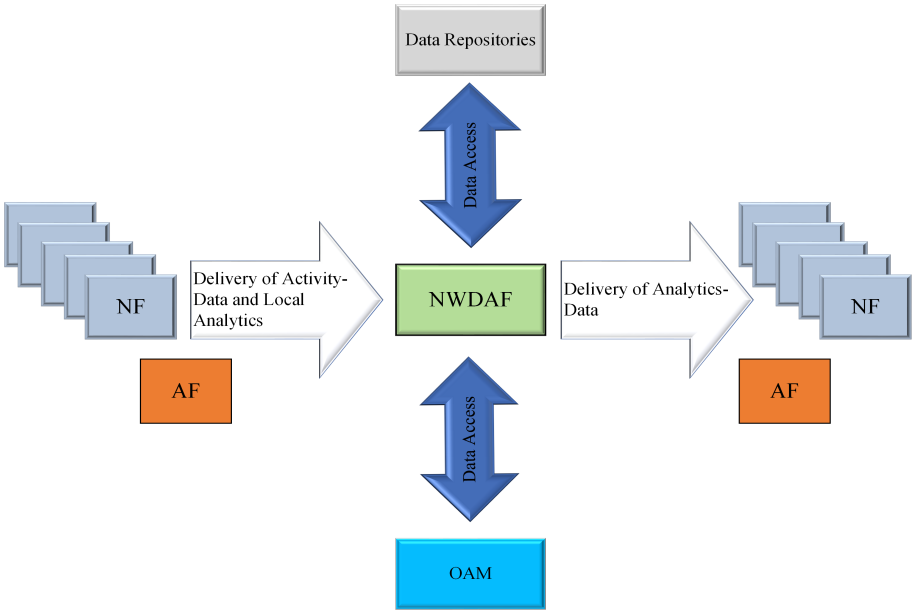


Figure 1.2: A generalized framework for 5G network automation in Release 16, representing that NWDAF should be able to collect data from the operator OAM, AFs and 5GC network functions [10]. Source: [1].

- ML can be effective to deal with network complexity. 6G networks will be more complex due to numerous network topologies, immense growth in the cellular users, staggering data rate demands, complex air interface, vast network coordination methods, etc. Forecasting considerable complexity of 6G networks, the derivation of optimum performance solutions is nearly infeasible without ML.

- ML can play a vital role to deal with model deficit problems. Current cellular networks are amenable for mathematical derivation, for instance, information theory gives closed-form expressions for various problems such as Shannon theorem. However, the inherent complexity of 6G networks hinders the possibility of exploiting closed-form analytical expression(s), which can be due, for instance, to non-linearities either in the channel or network devices. ML offers an efficient way to deal with non-linearities, providing feasible solution(s) in a tractable manner.

- ML can cope with algorithm deficit problems. In current cellular networks, many optimal algorithms, although well-characterized, are impractical to be implemented. Considering the example of multiple-input multiple-output (MIMO) systems where optimal solutions are known (e.g., dirty paper coding), they are overlooked in favour of linear solutions, e.g., linear minimum mean-squared error. It is envisaged that ML can pave the way to implement more efficient yet practical solutions.

1.1.2 Role of ML in Standardization

The potential of ML for 5G has been widely acknowledged in the literature and applications made it even in the standard at higher levels, e.g., networking and security. Third generation partnership project (3GPP) has introduced a specification, named network data analytics function (NWDAF), in Release 15 and 16, as part of the 5G core (5GC) architecture [10]. NWDAF is responsible for providing network analytics when requested by a network function (NF). Data is collected via application function (AF), operation, administration, and maintenance(OAM), NF, and data repositories. The specifications have also addressed the problem of inter-working for automation and data collection, which analytics vendors previously faced. 3GPP NWDAF framework for 5G systems is depicted in Fig. 1.2. This automation gives leverage to network vendors for the deployment and testing of non-real-time ML-related use cases. In Fig. 1.2, inward interfaces aggregate data from different network sources, where communication occurs using existing service-based interfaces. Outward interfaces provide decisions (analytics-based, algorithmic) to AF and NF.

Regarding PHY, ML techniques lag behind, due to a number of issues. First, PHY makes use of abstractions and mathematical models that are inferred from the physical reality and electromagnetic principles. As long as such models describe the real-world precisely, there is no need for ML. Nevertheless, in practice, models and fixed algorithms are inefficient when facing rapidly changing and heterogeneous environments. For example, using the same channel acquisition scheme to acquire channel state information (CSI) from a laptop in line-of-sight (LoS) with a base station (BS), a tablet on a fast train, or a mobile quickly moving in a super densely covered area might not be optimal. Consequently, the standardization efforts of intelligent techniques have gained momentum, and while 3GPP is ready to begin a study item on ML implementations, open-radio access network (RAN) will be ML-native, defining a RAN intelligent controller, which will enhance several RAN functions.

Motivated by the huge applications and promising benefits of ML in communication systems, the thesis focuses on enhancing the performance of acquired CSI in the communication system. In the following, we discuss what CSI is. How is it acquired, and what is the associated problem with CSI acquisition?

1.2 Channel State Information

Broadly speaking, CSI is a term used to represent knowledge of a *channel*. What is a *channel*? In the simplest form, any wireless communication process is composed of three components: *source*, *channel*, and *receiver*. *Source* is a network entity, also referred to as a transmitter, that generates the message and transmits it to the *receiver*. The transmitted message propagates through the medium, called *channel*. The *channel* is susceptible to many impediments that may hinder communication. Hence, knowledge of the *channel* prior to data transmission is indispensable.

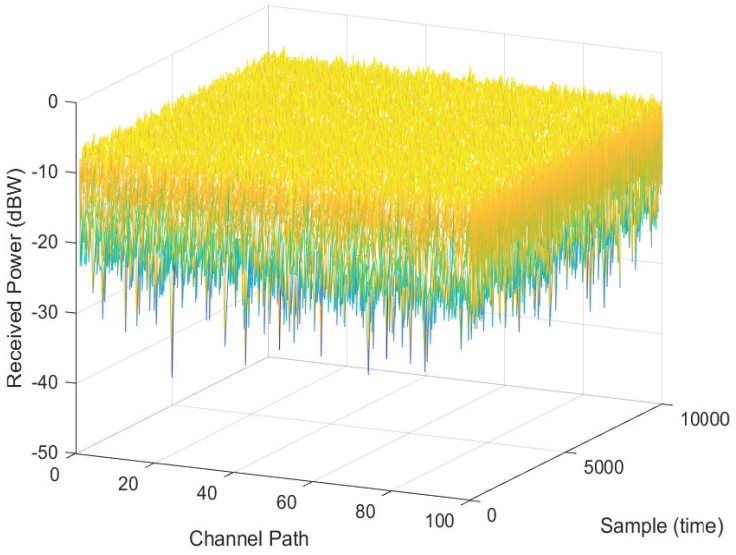


Figure 1.3: An illustration of a standing wave pattern of a tapped delay line (TDL) single-input single-output (SISO) channel.

1.2.1 Quest for CSI

CSI gives information of how the transmitted message propagates between the transmitter and the receiver. CSI is of predominant importance for several reasons. Let us elaborate this point with an example. Consider a receiver, user equipment (UE), is travelling through a standing wave pattern, as illustrated in Fig. 1.3, then the power of the received signal at the UE will fluctuate. Imagine a transmitter, BS, is continuously communicating with the UE. In the case of fading dip, there is a high probability of data loss as channel gain is very low. If the BS gets the information about when these fading dips occur, then the performance of communication system can be greatly improved. For instance, in the case of deep fade at one UE, the BS can schedule the communication with another UE with high channel gain. Consequently, spectral efficiency of the overall system can be maximized.

To improve the performance of a communication system, many transmission techniques, such as bit-loading, coding, precoding methods, adaptive modulation, channel-aware scheduling etc., are bounded to have accurate CSI at the transmitter. Accurate CSI, notably, improves the performance of many wireless techniques, for instance, MIMO, ultra-reliable transmissions, relaying and PHY security. Hence, CSI at the transmitter is of pivotal importance. The question is how the transmitter gets the CSI before data transmission. In the following subsection, we answer this question.

1.2.2 CSI Acquisition at BS

To acquire downlink CSI at the BS, there are two steps. In the first step, UE estimates the CSI through pilots, known as CSI reference symbols (CSI-RS), transmitted by the BS. And in the second step, the estimated CSI is fed back to the BS via the uplink control channel. CSI estimation is beyond the scope of this thesis; therefore, assuming perfect CSI estimation, we focus on CSI feedback.

1.2.3 What is CSI Feedback?

The concept of reporting downlink CSI to BS has been widely acknowledged as CSI feedback. The standard CSI feedback is composed of three parts: precoding matrix indicator (PMI), channel quality indicator (CQI), and rank indicator (RI). PMI is the most important one, as it helps the BS to select an appropriate beam. Briefly speaking, CSI-RS that gives the best signal-to-interference-plus-noise ratio (SINR) is selected, and then the corresponding PMI from the codebook (explained in Section 1.4.1) is reported to the BS along with CQI[1] and RI[2]. To reduce over-the-air (OTA) overhead, estimated CSI is compressed, i.e., reporting PMI; thereby, acquired CSI at BS is prone to compression errors. Such compression can deteriorate the performance of a massive MIMO precoder.

1.3 Thesis Objective and Problem Statement

Currently, ML is adopted at higher layers, e.g., network layer of the 5G standard and at PHY is regarded as an optimization tool, hence used for implementation but non-standard relevant. The thesis will aim to identify the specific standard field that can be enhanced by using ML and the relative standardization opportunities. The thesis aims to expand the ML field, exploiting tool(s) that can be used at PHY of practical 5G networks to improve the performance of communication systems. Such algorithm(s) should provide standard-relevant ways to reduce CSI feedback overhead and improve CSI quality.

In modern wireless networks, the standard degrees of freedom of communication (i.e., time, frequency and space) are being sided by *intelligence*, which is envisioned as a further available resource that can be exploited to enhance communication efficiency. In 3GPP, standardization efforts encapsulate ML at higher layers, such as networking, leaving the role of implementation detail for PHY to ML. As a result, in 5G networks, several functional aspects are driven solely by predetermined rules, which can perform efficiently only if the model assumptions fit the natural environment. However, large amounts of specific data, e.g., CSI, can be exploited by intelligent mechanisms to better cope with the specific environment.

Focusing on CSI feedback, the problem of OTA feedback overhead reduction is yet to be addressed. Though standard codebook-based methods have been opted as a solution;

[1] Used for the indication of channel quality to BS. It has a value between 0 to 15, indicating the modulation and coding level that UE can operate.

[2] It is used to report the number of independent communication channels, which can help to understand how well multiple antennas work, i.e., is the signal transmitted by different antennas correlated or not?

nevertheless, CSI precision is greatly effected due to compression. Further, codebooks are designed by assuming a given channel distribution. The underlying assumption may differ from a practical environment, leading to performance drop. Therefore, there is need to introduce adaptive algorithms to circumvent the issues faced by conventional methods. In this thesis, we address the problem: *How CSI feedback overhead can be reduced while improving the precision of acquired CSI at the BS?*

1.4 State-of-the-Art and Research Trends

CSI feedback overhead in a massive MIMO system occupies substantial uplink bandwidth resources. A plethora of research work has been done on massive MIMO CSI feedback overhead reduction. In this section, we will provide an overview of CSI feedback mechanisms given in the literature. To assist readers, we divide CSI feedback literature into three categories: compressive sensing (CS), codebook, and deep learning (DL). In the following, we give an overview of these techniques and their limitations.

1.4.1 Codebook-Based CSI Feedback

Codebook is a set of precoding matrices, and it was introduced in 2009 with the release of the first LTE standard. In 5G, two types of codebook are defined: type-I and type-II [15]–[17]. Type-I reports only the phase of the selected beam, whereas type-II reports subband and wideband amplitude information. In comparison, type-II is more detailed CSI reporting and is mainly designed for multiuser MIMO, hence supporting multiple beams. In type-II, four beams can be reported at the expense of overhead. Type-II linearly combines a group of beams within a group, while type-I selects one beam from a group of beams [18].

Codebook-based CSI feedback is at the base of various commercial systems, such as LTE/LTE-advanced and 5G-NR, that reduce OTA feedback [19]. In this technique, a codebook is predefined by the standardization bodies and is known to both network entities, i.e., BS and UE. Upon estimating the CSI, the UE maps the estimated CSI to the codebook to search for the closest *codeword*. Then, the UE feedback the *index* of the closest *codeword*. In the standardization terms, the *index* is called PMI, as already explained in Section 1.2.3. The BS acquires the *codeword* by searching the PMI in the codebook, and this *codeword* is assumed as an estimated CSI. This process shows that the acquired CSI at the BS is massively compressed as we assume a *codeword* as estimated CSI. Hence, CSI feedback overhead reduces by compromising on acquired CSI quality.

Some studies have been done on codebook-based CSI feedback in [20]–[25]. For example, by considering LoS and non-line-of-sight (NLoS) components between the BS and UE, codebooks are designed by [21] and [20], respectively. In addition, an antenna grouping-based feedback reduction method is proposed in [22], where multiple correlated antenna elements are mapped to a single-value by exploiting predesigned patterns. A codebook is designed in [23] to compress CSI by utilising CS theory. Similarly, in [24],

an angle-of-departure (AoD)-based codebook design is proposed, compressing CSI more accurately; nevertheless, overhead increases linearly with the AoDs.

There are several challenges associated with codebook-based CSI feedback. Here we highlight some of them. First, the downside of a codebook is its environment-dependence. The existing codebooks are designed on a given channel distribution, which may vary depending on the propagation conditions of each cell. Second, CSI accuracy improves with the size of the codebook. However, increasing the codebook size increases the complexity of searching the *codeword*. Though some adaptive codebooks have been introduced, CSI quality, complexity, and codebook-related overhead still need enhancement [19]. Last but not least, type-II performs better than type-I at the expense of high feedback bits to design the codebook.

1.4.2 Compressive Sensing-Based CSI Feedback

Compressive sensing (CS) is a signal processing technique that can acquire and reconstruct CSI by exploiting underdetermined linear systems. The basic principle behind this is sparsity of CSI can be exploited to acquire CSI using fewer samples than required by Nyquist–Shannon sampling theorem. In a nutshell, CSI at the UE is compressed utilising a sensing matrix [26]. In the literature, many algorithms have been proposed that use CS for CSI feedback [27]–[31]. For example, the authors of [29] and [30] use spatial and temporal correlation of CSI to reduce feedback overhead. This idea has inspired the establishment of protocols for CSI feedback. Multi-dimensional CS theory, which is based on spatial and frequency correlation of the channel and Tucker decomposition [32], is used in [31] to reduce feedback overhead. Such algorithms do not recover accurate CSI, and massive data processing is required, making them infeasible.

CS-based CSI feedback cannot fully recover compressed CSI due to the use of simple sparsity prior while the channel matrix used is not perfectly but is *approximately* sparse [27], [33]. Besides, the assumption of CSI sparsity may not hold in a realistic propagation environment. Last, the computation cost of algorithms used for iterative reconstruction makes them infeasible for deployment in a real-time scenario [33], [34].

1.4.3 Deep Learning-Based CSI Feedback

Deep learning (DL), a class of ML having multiple hidden layers, has also been utilized to reduce CSI feedback overhead. The literature borrows the idea of *autoencoder*, a DL technique used in image compression, for CSI feedback. Particularly, inherent features of *autoencoder*, i.e., *encoder* and *decoder*, are used to compress and recover CSI, respectively. The former, implemented at UE, encodes the estimated CSI into *codeword* and the latter for reverse-engineering, i.e., recovering the estimated CSI, at BS, from transmitted *codeword*. The training of *autoencoder* is done in end-to-end manner.

In the literature [35]–[51], CSI matrix is treated as an image and then the concept of *autoencoder* is used. For example, in [35], the CSI matrix is considered as an image and then a DL-based *encoder* and *decoder* is designed, named Autoencoder-based CSI network (CsiNet), to reduce feedback overhead. A more practical approach, i.e., com-

pression of noisy (estimation noise) CSI, is studied in [40], where noisy features are first removed, and then compression is performed. Due to a lack of structure among the latent vectors in the standard *autoencoder*, it cannot be used in generative modelling. To cope with this issue, [52] introduced the concept of variational autoencoder (VAE). Briefly speaking, the input in VAE is encoded as a distribution over the latent space. The encoded output is the mean and standard deviation, and a point from latent space is sampled using a predefined distribution. Hence, a latent vector is reported as a codeword to BS [36]. [52] proposed CSI feedback mechanism by exploiting VAE. Similarly, in [53] compressed CSI is decoded with a generative adversial network (GAN) [54]. In summary, the entire literature work exploits *autoencoder* for CSI feedback. Further, to train the deep networks, a massive amount of synthetic datasets is considered [37], [42], [49]. Therefore, the robustness of DL-based techniques remains a key challenge, as they might perform well in a given distribution but fail in changing propagation conditions.

There are many disadvantages of *autoencoder*. First, it has high training costs, e.g., huge dataset requirements, massive amounts of parameters for tuning, and model validation. Second, it may lead to imperfect decoding, misunderstanding of influencing variables, and preservation of irrelevant information [55]. Third, the complexity is high, e.g., [35] require floating-point operations per second (FLOPs) 0.56 M and convolutional autoencoder require 58.52 M [36]. Storage and training of massive neural network (NN) architecture at, e.g., UE, can be nearly infeasible [8]. Last but not least, to deploy a DL-based method, a collaboration between the BS and a UE is indispensable, which poses new challenges for the standardization bodies [56]. In the following, we explain ML/DL-related challenges more from a standardization perspective before addressing the thesis outline and contributions of thesis work.

1.5 Challenges for Deploying ML in Cellular Networks

1.5.1 Data Availability and Benchmarking

One of the foremost challenges in wireless networks is *data availability*. Data availability concerns the problem of identifying a standard and accepted set of data (e.g., channel realizations) to test and benchmark ML algorithms. This problem is of pivotal importance for standardization, where typically algorithms and proposals are tested using agreed underlying physical models (e.g., urban macrocells/microcells channel models), evaluation methodologies and calibrated simulators. Contrary to other fields, cellular networks have no standard dataset to train and benchmark an ML algorithm. Therefore, a synthetic or software-generated dataset is of predominant importance to train and benchmark ML algorithm(s) and agree on a common evaluation methodology to rank proposition and standard algorithms. Identifying key performance indicators in wireless networks is another crucial task for ML standardization. It is necessary to design a set of metrics to classify and rank ML algorithms and their performance. Classic approaches such as throughput and SINR might not be sufficient since a slight improvement in these values might come at the cost of huge complexity augmentation and exacerbated energy

consumption.

1.5.2 Selection of ML versus Non-ML Solutions

ML tools are regarded as implementation-oriented rather than a standard relevant aspect. The idea is that each vendor can efficiently implement each standard aspect as long as the external interfaces are respected. A simple example of this is CSI feedback, where a UE needs to select a specific PMI, but the standard does not specify how this selection is performed. However, the idea of having ML-dedicated message exchanges and performance that only an ML-aided algorithm can achieve will pave the way for the standardization of ML algorithms [6], which is the focus of this thesis. This will open the door for several issues, e.g., will the standard impose a specific ML structure, classifying minimum performance and implementation structure, or will it remain far from the implementation? Regarding NNs, there is still an open question, i.e., hyperparameters will be left to vendor-specific implementation or will the standard set them.

1.5.3 Complexity of ML Algorithms

Considering the limited battery life, storage, computational capability, and communication bandwidth in most cellular network entities, an ML model's cost-performance tradeoff becomes a fundamental issue. Another issue is the speed/time-steps at which the training and inference need to be performed. Whereas hard-wired BS has the sufficient computational power to run complex ML algorithms, UEs need to face battery, heating and stringent complexity limits. Possible solutions to such issues include but are not limited to, the implementation of substitute rule-based algorithms at the UE side and migrating the load all on the BS side, which we will explain in Chapter 3.

1.5.4 Stability and Adaptability of ML Techniques

ML algorithms applied to wireless networks must be adaptive, as they must deal with dynamically changing parameters. Particularly, the weights of the NN are evaluated online based on the trained data. However, this approach may not be applicable in wireless, specifically in a standard, where coordination among entities belonging to different operators and provided by different vendors have to coexist, and the need for quick response could prevent one or the other solution. Possible solutions include pre-trained or partially trained NN (i.e., NN in which the starting point is pre-set); cloud-based downloadable dataset for NN training; codebook-based NN, in which a codebook of different NNs is used and agreed upon between the BS and UEs. Another related problem is detecting an outdated ML model with high inference error and replacing it. Replacing an obsolete model with a new model incurs further delay. Thus, there must be a proactive mechanism to adapt the ML model to network conditions such that network functions suffer minimum performance loss.

This thesis mainly addresses data availability challenges by exploiting data augmentation algorithms, selecting ML versus non-ML solutions, and reducing ML complexity by introducing rule-based approaches at the UE.

1.6 Thesis Outline and Contributions

The thesis is composed of five chapters including this introduction chapter. The major contributions of the rest of the chapters are given below.

In Chapter 2 (CSI Feedback Enhancement in Single-User), we consider the problem of CSI feedback overhead reduction by using twin ML-based channel predictor (CP), called CSI feedback prediction network (CSIFB-PNet). ML training is performed at the BS and UE in parallel. With the trained CPs, CSI feedback is evaluated with respect to the predicted channel at the UE. As a toy example, if CSI prediction and estimation are perfect, then there is no need to feedback CSI. Alternatively, if the UE has to feedback something, the overhead will be less than without ML, thanks to predicted CSI. Later, by addressing the issues related to ML training at both ends, we extend our idea to one-sided ML training. Specifically, ML training is performed at the UE, and the trained model is reported to BS for twin predictions. The results, evaluated by exploiting empirical data, hold (i) error reduction in the acquired CSI at the BS; (ii) precoding gain maximization; and (iii) maximizing cosine similarity with minimum feedback overhead. Last, we will focus on various ML-based CSI prediction techniques. The outcomes corroborate the validity of exploiting ML for CSI feedback enhancement at minimum OTA overhead.

Chapter 3 (CSI Feedback Enhancement in Multiuser) investigates the performance in a multiuser environment. Besides, in Chapter 2, ML is considered at the BS and UE to improve the precision of acquired CSI. However, ML implementation at the UE can be infeasible for various reasons, such as UE power consumption and storage of heavy NN. Motivated by this issue, we propose CSI learning at BS (CSILaBS), to avoid ML at UE. To this end, by exploiting CP at BS, light-weight predictor function (PF) is considered for feedback evaluation at the UE, which can reduce OTA feedback overhead and improve acquired CSI quality at the BS. In addition, various mechanisms are proposed to select the feedback at a UE while aiming to improve CSI accuracy.

Chapter 4 (Role of RIS in CSI Prediction) addresses the use of reconfigurable intelligent surfaces (RIS) in the proposed CSI feedback work. Particularly motivated by the promising benefits of CP in CSI feedback enhancement, this chapter exploits RIS for CSI prediction enhancement. As the prediction of a highly dynamic channel is cumbersome, this chapter focuses on enhancing multiuser CSI prediction performance by introducing time correlation in the composite channel. Specifically, we present the use of quasi-RIS to achieve the objective. We show that the intelligently configured RIS enhances channel prediction performance while saving sufficient RIS elements for other applications. Another advantage is the prediction of the whole composite channel arriving from the entire RIS; hence, the complexity and overhead for channel estimation can be reduced, which is the major bottleneck in RIS.

Chapter 5 (Conclusion and Future Outlooks) concludes the thesis work and shed light on the promising future directions, which can be taken as a continuation of proposed studies. Besides, a standard-relevant solution is proposed to address the reporting and verification of training data for the CSI frameworks. It is important to highlight that each chapter has its own mathematical notations and the system model.

1.7 Patents and Publications

The following patents were filed during the thesis. Furthermore, research publications produced in the thesis work are provided.

1.7.1 Patents

Below is the list of only filed patents. The rest are not provided due to confidentiality.

- M. K. Shehzad, F. Jardel, L.Rose, and M. M. Butt "Data augmentation for CSI feedback", NC326631, 2022.
- L. Rose, and M. K. Shehzad, "CSI feedback using channel prediction: Exploiting UE's capability", NC328095, 2023.
- L. Rose, and M. K. Shehzad, "Reduced overhead and estimation error to report CSI in wireless networks", NC321108, 2020.
- L. Rose, and M. K. Shehzad, "Shifting AI/ML to BS for CSI feedback enhancement", NC326027, 2022.

1.7.2 Publications

The work of this thesis has resulted in the following publications:

Published Journal Papers

- M. K. Shehzad, L. Rose, M. M. Butt, I. Z. Kovács, M. Assaad, and M. Guizani, "Artificial intelligence for 6G networks: Technology advancement and standardization," *IEEE Vehicular Technology Magazine*, vol. 17, no. 3, pp. 16–25, 2022.
- M. K. Shehzad, L. Rose, S. Wesemann, and M. Assaad, "ML-based massive MIMO channel prediction: Does it work on real-world data?" *IEEE Wireless Communications Letters*, vol. 11, no. 4, pp. 811–815, 2022.
- M. K. Shehzad, M. Assaad, and L. Rose, "Reconfigurable intelligent surfaces: How to enhance multiuser channel prediction?" *IEEE Wireless Communications Letters*, pp. 1–1, 2023.

Published Conference Papers

- M. K. Shehzad, L. Rose, and M. Assaad, "Dealing with CSI compression to reduce losses and overhead: An artificial intelligence approach," in *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2021, pp. 1–6.
- M. K. Shehzad, L. Rose, and M. Assaad, "A novel algorithm to report CSI in MIMO-based wireless networks," in *ICC 2021 - IEEE International Conference on Communications*, 2021, pp. 1–6.

- M. K. Shehzad, L. Rose, and M. Assaad, "RNN-based twin channel predictors for CSI acquisition in UAV-assisted 5G+ networks," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.

- M. K. Shehzad, L. Rose, M. F. Azam, and M. Assaad, "Real-time massive MIMO channel prediction: A combination of deep learning and NeuralProphet," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, 2022, pp. 1423–1428.

Submitted Journal Papers

- M. K. Shehzad, L. Rose, S. Wesemann, M. Assaad, and S. A. Hassan, "Design of an efficient CSI feedback mechanism in massive MIMO systems: A machine learning approach using empirical data," *in IEEE Transactions on Cognitive Communications and Networking (TCCN)*, 2023.

- M. K. Shehzad, L. Rose, and M. Assaad, "Massive MIMO CSI feedback using channel prediction: How to avoid machine learning at UE?," *in IEEE Transactions on Wireless Communications (TWC)*, 2023.

Chapter 2

CSI Feedback Enhancement in Single-User

2.1 Introduction

Over the past two decades, multi-antenna techniques dubbed massive MIMO have been widely acknowledged as key enablers for future cellular networks [57]. Employing a large number of antennas in a centralized [58] or distributed [59] configuration can help in, e.g., reducing multiuser interference [60], multifold increase in channel capacity [61], cost-effective and reliable coverage. To attain aforementioned gains, massive MIMO technology requires downlink CSI. In the current communication architecture, the downlink CSI is estimated by the UE via dedicated pilots and then fed back to the BS. The feedback information is compressed to reduce OTA overhead. This compression increases the inaccuracy of acquired CSI, thus degrading precoding quality. Different from the existing DL-based strategies, i.e., designing encoders and decoders [35], and accordingly transforming the CSI matrix into an image, we introduce channel prediction for CSI feedback. This chapter introduces the use of ML to develop CSIFB-PNet, a novel CSI feedback mechanism which exploits twin channel predictors to compress and recover the CSI effectively. We call these predictors twins because they use the same weights and test dataset for channel prediction. The twin channel predictors are particularly deployed at the BS and UE for CSI feedback. In summary, the feedback at the UE is evaluated with respect to the predicted channel. Further, we provide a comparison with state-of-the-art [35]. The major contributions of this chapter are:

- By considering a realistic scenario, we propose the idea of using CSIFB-PNet to enhance CSI feedback. More specifically, the proposed approach is hybrid, i.e., a combination of conventional CSI feedback and ML-based twin predictors. CSIFB-PNet can help to eliminate the feedback if the prediction is aligned with the estimation. Alternatively, it reaps the benefits of reducing OTA feedback overhead and CSI compression errors.

- CSIFB-PNet can train the ML model on two-sides as well as one-side, where in the later UE reports the trained model to the BS. In this chapter, we discuss both techniques. Further, CSIFB-PNet can support the possible standardization in 3GPP. Therefore, we also cover the possible standardization points of CSIFB-PNet.

- As CSI prediction is of pivotal importance in CSIFB-PNet, we also address ML-based CPs. Specifically, we exploit time-series models of DL, e.g., bidirectional long short-term memory (BiLSTM). In addition, we use NeuralProphet (NP), a recently introduced time-series model composed of statistical components, e.g., auto-regression (AR), for CSI prediction. Furthermore, inspired by a statistical model, we develop a novel hybrid framework comprising recurrent neural network (RNN) and NP to achieve better prediction accuracy.

- To corroborate the validity of CSIFB-PNet, we use experimental data, which is a huge value-addition over the existing work. The experimental dataset is based on a measurement campaign performed at Nokia Bell-Labs and provides a real-life setting for evaluating and providing interesting insights into our work.

2.1.1 Chapter Organization and Notations

The rest of this chapter is organized as follows. The system model and conventional CSI feedback approach are presented in Section 2.2. In Section 2.3, CSIFB-PNet is explained with two-sided as well as one-sided ML training. The proposed channel prediction models are explained in Section 2.4. Measurement campaign is explained in Section 2.5. Numerical results are presented in Section 2.6. Finally, Section 2.7 concludes the chapter.

Throughout this paper, the matrices and vectors are represented by boldface upper and lower-case, respectively. Also, scalars are denoted by normal lower and upper-case. The estimated, compressed estimated, predicted, and true channel matrix are represented by $\widehat{\mathbf{H}}$, $\widehat{\mathbf{H}}^{Q}$, $\widetilde{\mathbf{H}}$, and $\mathbf{H}$, respectively, and their vectorized forms are respectively denoted by $\widehat{\mathbf{h}}$, $\widehat{\mathbf{h}}^{Q}$, $\widetilde{\mathbf{h}}$, and $\mathbf{h}$. The superscripts $[\cdot]^{\dagger}$ and $[\cdot]^{*}$ denote the transpose and conjugate transpose of a matrix/vector, respectively. In addition, $\mathsf{E}\{\cdot\}$ and $\|\cdot\|^2_{\mathsf{FRO}}$ denote expectation operator and squared Frobenius norm, respectively. Furthermore, the notations $\mathbb{R}$ and $\mathbb{C}$ are representing the real and complex numbers, respectively, and $|\cdot|$ shows absolute value.

2.2 System Model and Conventional Approach

2.2.1 System Model

Consider the downlink scenario of a massive MIMO orthogonal frequency-division multiplexing (OFDM) system, as depicted in Fig. 2.1, where a BS is serving multiple UEs within a cell. For the sake of simplicity, we explain the communication between a single BS-UE link. The BS and UE are equipped with N_t and N_r transmit and receive

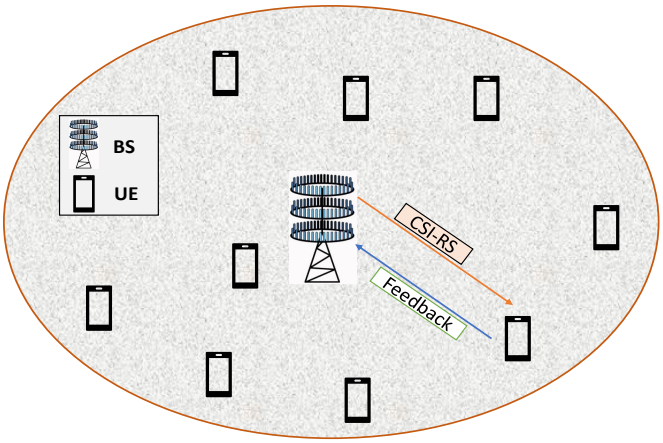


Figure 2.1: Single-cell massive MIMO-OFDM communication environment, where two network entities are depicted, i.e., BS and a UE. To acquire the channel, BS is sending CSI-RS; consequently, compressed estimated channel is fed back via dedicated feedback link.

antennas, respectively. Without loss of generality, the received signal[1] at the UE can be represented as

$$\mathbf{y}(t) = \mathbf{H}(t)\mathbf{p}(t) + \mathbf{n}(t)\,, \tag{2.1}$$

where $\mathbf{y}(t) = \left[y_1(t), y_2(t), \cdots, y_{N_r}(t)\right]^\dagger$ having dimension $N_r \times 1$, and t is the time index. And,

$$\mathbf{H}(t) = \begin{bmatrix} h_{11}(t) & \cdots & h_{1N_t}(t) \\ h_{21}(t) & \cdots & h_{2N_t}(t) \\ \vdots & \ddots & \vdots \\ h_{N_r1}(t) & \cdots & h_{N_rN_t}(t) \end{bmatrix} \tag{2.2}$$

is the channel matrix, which has dimension $N_r \times N_t$. In the above matrix, $h_{n_rn_t} \in \mathbb{C}^{1\times 1}$ is the channel gain between the n_t^{th} transmit and n_r^{th} receive antennas, respectively, where $1 \leq n_t \leq N_t$ and $1 \leq n_r \leq N_r$. Further, $\mathbf{p}(t) = \left[p_1(t), p_2(t), \cdots, p_{N_t}(t)\right]^\dagger$ is an $N_t \times 1$ symbol vector transmitted by the N_t antennas, and $\mathbf{n}(t) = \left[n_1(t), n_2(t), \cdots, n_{N_r}(t)\right]^\dagger$ is the additive noise vector, which has dimension $N_r \times 1$.

In the frequency-division-duplex system, $\mathbf{H}$ should be fed back to the BS by a UE, requiring high OTA overhead. However, $\mathbf{H}$ must be estimated at the UE prior to feedback. Channel estimation is beyond the scope of this study; therefore, we assume perfect

[1] For the reader's understanding, here we are referring to one OFDM symbol and subcarrier.

channel estimation and focus on CSI feedback. In the following, we summarize the CSI feedback process followed in the conventional approach. Then, we will explain CSIFB-PNet, which combines conventional approach and ML-based channel predictors.

2.2.2 Conventional Approach

Conventionally, CSI at the BS is acquired by transmitting CSI-RS. Consequently, the UE estimates the channel, denoted by $\widehat{\mathbf{H}}_{\mathsf{UE}}$, and then feedback a compressed estimated channel, expressed as

$$\widehat{\mathbf{H}}_{c}^{Q}(t) = Q\left(\widehat{\mathbf{H}}_{\mathsf{UE}}(t)\right) , \tag{2.3}$$

where $Q(\cdot)$ is the standard element-wise quantization function using B_Q quantization bits to compress the channel. Here, it is important to highlight that we aim at reducing B_Q as the higher value will result in higher overhead. $\widehat{\mathbf{H}}_{c}^{Q}$ is degraded by two major factors: estimation and *compression* or say *quantization*. Throughout the chapter, we use these two terms interchangeably.

2.3 CSIFB-PNet

CSIFB-PNet considers twin channel predictors at both ends of the communication system, i.e., BS and UE. The UE will evaluate the feedback with respect to the predicted CSI. As a toy example, if there is no difference between the predicted and the estimated CSI at the UE, then feedback is not required. Thus, feedback-related OTA overhead can be eliminated or reduced if the UE decides to feedback something. In the following, we explain two-sided CSIFB-PNet, which is later extended to one-sided CSIFB-PNet. We refer to two-sided CSIFB-PNet as CSIFB-PNet2 and one-sided as CSIFB-PNet1 in the rest of this chapter.

2.3.1 Two-sided CSIFB-PNet

In the two-sided CSIFB-PNet, ML models are trained at both ends by using the same initialization parameters and training data etc. CSIFB-PNet2 is composed of the following phases: assessment, initialization, prediction, estimation, compression, feedback and recovery of compressed CSI. Below, we explain these phases.

2.3.1.1 Assessment Phase

In the beginning, the two network entities exchange a few messages: 1) BS and UE perform a handshake to assess available capabilities, for instance, support of ML algorithm 2) both agree to use ML-based algorithm for CSI prediction 3) both agree to adopt the conventional approach to aggregate past CSI realizations to train ML algorithm 4) the length of the dataset (to be acquired in the initialization phase) is decided for the training of ML model. The assessment phase is pictorially represented in Fig. 2.2. Once such messages are exchanged, the BS and UE perform the initialization phase, summarized below.

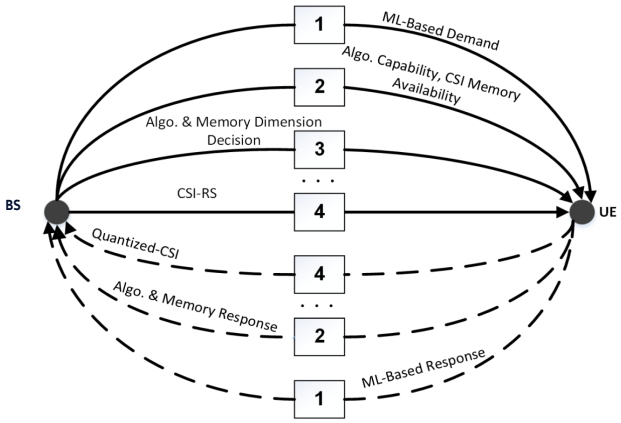


Figure 2.2: Dedicated message exchanges, we call this assessment phase, between the BS and UE before adopting CSIFB-PNet2.

2.3.1.2 Initialization Phase

During the initialization phase, the BS and UE use the conventional approach, detailed in Section 2.2.2. By using the conventional approach, a set of past CSI realizations is established at both ends, where the dataset length is agreed upon in the assessment phase. Let us denote the set of past CSI realizations as $\mathcal{S} \triangleq \{\widehat{\mathbf{H}}^Q(t), \widehat{\mathbf{H}}^Q(t-1), \widehat{\mathbf{H}}^Q(t-2), \cdots, \widehat{\mathbf{H}}^Q(t-d), \cdots, \widehat{\mathbf{H}}^Q(t-S)\}$, where S is the total length of initialization phase.

2.3.1.3 Prediction Phase

Using the same dataset available at the BS and UE, both the network entities train the ML-based channel predictor, e.g., RNN, which we explain later in Section 2.4. It is important to mention that RNNs are twin, i.e., their initial configuration and training dataset is the same. By having the trained RNNs at both sides, let us say at time instant t, the BS and UE predict the CSI, denoted by $\widetilde{\mathbf{H}}(t)$. The prediction is assumed to be the same on both sides. For the sake of notational convenience, we rewrite the predicted CSI at BS and UE as $\widetilde{\mathbf{H}}_{\mathsf{BS}}(t)$ and $\widetilde{\mathbf{H}}_{\mathsf{UE}}(t)$, respectively.

2.3.1.4 Estimation Phase

Suppose that the UE estimates the CSI at time t, which is denoted as $\widehat{\mathbf{H}}_{\mathsf{UE}}(t)$. As alluded to earlier, we assume that there is no error in the CSI estimation; hence, $\widehat{\mathbf{H}}_{\mathsf{UE}} = \mathbf{H}$. In the following, we explain CSI compression at the UE.

2.3.1.5 Compression Phase

Briefly, in the conventional approach, compression at the UE will be done in the same manner as explained in Section 2.2.2, i.e., Equation (2.3). Whereas in CSIFB-PNet, compression at the UE is done in the following manner. The UE computes an

update function by exploiting predicted and estimated CSI as

$$\mathbf{U}(t) = f\left(\widetilde{\mathbf{H}}_{\mathsf{UE}}(t), \widehat{\mathbf{H}}_{\mathsf{UE}}(t)\right) \tag{2.4}$$

where $f(\cdot)$ is an update function which measures the distance between the predicted and estimated CSI at UE. A simple implementation of such a function could be a difference. Hence, Equation (2.4) can be rewritten as

$$\mathbf{D}(t) = \widetilde{\mathbf{H}}_{\mathsf{UE}}(t) - \widehat{\mathbf{H}}_{\mathsf{UE}}(t)\,. \tag{2.5}$$

In succession, the updated function, $\mathbf{D}(t)$, is compressed at the UE using a standard element-wise quantization function. The compressed CSI at the UE can be written as

$$\widehat{\mathbf{H}}_{p}^{Q}(t) = Q\left(\mathbf{D}(t)\right)\,. \tag{2.6}$$

By substituting Equation (2.5) into Equation (2.6), we obtain the compressed CSI as

$$\widehat{\mathbf{H}}_{p}^{Q}(t) = Q\left(\widetilde{\mathbf{H}}_{\mathsf{UE}}(t) - \widehat{\mathbf{H}}_{\mathsf{UE}}(t)\right)\,. \tag{2.7}$$

2.3.1.6 Feedback and Recovery Phase

The feedback and, correspondingly, the recovery of the compressed CSI are categorized into two scenarios. In the following, we explain each scenario.

- In the first scenario, we consider the case of perfect CSI prediction, i.e., when the estimated and predicted CSI at the UE are the same. In other words, $\widetilde{\mathbf{H}}_{\mathsf{UE}}(t) = \widehat{\mathbf{H}}_{\mathsf{UE}}(t)$. In this case, the UE does not need to feedback anything, and hence, feedback-related overhead is eliminated. Notably, the acquired CSI at the BS will be the same, which it had predicted, i.e., $\widetilde{\mathbf{H}}_{\mathsf{BS}}(t)$. Mathematically, the acquired CSI at the BS is given as
$$\bar{\mathbf{H}}_{\mathsf{BS}}(t) = \widetilde{\mathbf{H}}_{\mathsf{BS}}(t)\,. \tag{2.8}$$
- In the second scenario, we consider the case when the UE has to feedback something. In short, when $\widetilde{\mathbf{H}}_{\mathsf{UE}}(t) \neq \widehat{\mathbf{H}}_{\mathsf{UE}}(t)$. Therefore, the UE feedback compressed update, i.e., Equation (2.7). Correspondingly, the acquired CSI (or the recovered CSI) at the BS can be calculated as
$$\bar{\mathbf{H}}_{\mathsf{BS}}(t) = \widetilde{\mathbf{H}}_{\mathsf{BS}}(t) - Q\left(\widetilde{\mathbf{H}}_{\mathsf{UE}}(t) - \widehat{\mathbf{H}}_{\mathsf{UE}}(t)\right)\,, \tag{2.9}$$
which is simply the difference between the predicted CSI at the BS and feedback from the UE. In CSIFB-PNet, before reporting $\widehat{\mathbf{H}}_{p}^{Q}(t)$, the UE computes the error, Λ, and cosine similarity, ρ, between what will be acquired at BS, i.e., $\bar{\mathbf{H}}_{\mathsf{BS}}(t)$, and the estimated CSI at UE, $\widehat{\mathbf{H}}_{\mathsf{UE}}(t)$. Importantly, in the conventional approach, $\bar{\mathbf{H}}_{\mathsf{BS}}(t) = \widehat{\mathbf{H}}_{c}^{Q}(t)$. In the case when the error in the proposed approach, Λ_p, is less than the error in the conventional approach, Λ_c, as well as cosine similarity in the proposed approach, ρ_p, is higher than cosine similarity in the conventional approach, ρ_c, then the UE will feedback $\widehat{\mathbf{H}}_{p}^{Q}(t)$. Otherwise, UE reports $\widehat{\mathbf{H}}_{c}^{Q}(t)$ with a flag indicating the use of conventional approach.

2.3.2 One-Sided CSIFB-PNet

In the two-sided CSIFB-PNet, we considered the training of ML models at both ends, that is, BS and UE. Also, we exploit compressed data on both sides to train the ML model. The reason for considering compressed data for the training is to remain inline with the standards, i.e., if the UE and BS have the compressed data, then they should only rely on that. The downsides of such methodology are synchronization of two ML models, reporting of training data to BS, and exploiting compressed CSI for identical outcomes on both sides. In addition, using the compressed CSI will not achieve the case where the estimated and predicted CSI at the UE will be the same. It is, nevertheless, possible when training data is reported to BS with excessive overhead, i.e., no compression error. This approach is infeasible due to excessive OTA training overhead. In light of the above issues, we summarize the one-sided CSIFB-PNet, denoted with CSIFB-PNet1, below.

In CSIFB-PNet1, we exploit the uncompressed CSI for training the ML model, i.e., $\mathcal{S} \triangleq \{\widehat{\mathbf{H}}(t), \widehat{\mathbf{H}}(t-1), \widehat{\mathbf{H}}(t-2), \cdots, \widehat{\mathbf{H}}(t-d), \cdots, \widehat{\mathbf{H}}(t-S)\}$, which is possible at the UE. More specifically, we train the ML model only at the UE and report the trained model to the BS for twin predictions. CSIFB-PNet1 benefits over two-sided model training, e.g., synchronization of weight parameters and hyperparameters to initiate the training process is not required. Further, the requirement of reporting the training data to BS is eliminated. Furthermore, this helps to exploit the uncompressed CSI at the UE for training the ML model. The considerable benefit of such an approach is the elimination of feedback requirements from UE when estimated and predicted CSI at the UE are identical. The rest of feedback mechanism remains the same as in CSIFB-PNet2. In the following, we present pseudo-code of CSIFB-PNet.

The pseudo-code of CSIFB-PNet is given in Algorithm 1. In the first step of Algorithm 1, the training data is accumulated at the BS and UE or only at the UE in the case of one-sided CSIFB-PNet. Then ML model is trained. In step-2, the trained model is reported to BS, and CSI is predicted on both sides. Importantly, in CSIFB-PNet2, reporting of trained model is not required. Later, the CSI is acquired at the BS, which is summarized in step-3 and step-4.

CSIFB-PNet needs to be supported by the standard as the invention requires the establishment of a protocol between the BS and UE, which is currently not part of the 3GPP specification. To this end, different prediction algorithms can be tabled and standardized. Furthermore, the possible standardization points include algorithm(s), memory, message exchanges, etc.

2.3.3 Remarks on CSIFB-PNet

There are threefold advantages of using CSIFB-PNet over the conventional approach. First, the errors caused by compression can be smaller in $\widehat{\mathbf{H}}_p^Q(t)$ than in $\widehat{\mathbf{H}}_c^Q(t)$ due to having low amplitude entries in $\widehat{\mathbf{H}}_p^Q(t)$, thanks to CSI prediction. Therefore, a more accurate version of the true CSI can be acquired at the BS. The second benefit can be attained in quantization bits, B_Q, required to send feedback from the UE. The better

Algorithm 1: CSIFB-PNet

Input: $\widehat{\mathbf{H}}_c^Q$
Output: $\bar{\mathbf{H}}_{\text{BS}}(t)$

// Step-1: Aggregate training data at BS and UE (or UE only in case of CSIFB-PNet1), create a set $\mathcal{S}$, and reverse its order w.r.t. time index, i.e., most fresh entry will go at the end.

1: **for** $s = 1$ **to** $S - 1$ **do**
2: Train the ML model.
3: **end for**

// Step-2: Report the trained model to BS (this is not required in CSIFB-PNet2). Also, predict the CSI at time t at BS and UE by exploiting d past CSI realizations of $\widehat{\mathbf{H}}_c^Q$.
// Step-3: Estimate the CSI, $\widehat{\mathbf{H}}_{\text{UE}}(t)$, at UE at time t.
// Step-4: Calculate $\Lambda_p, \Lambda_c, \rho_p$, and ρ_c, and acquire $\bar{\mathbf{H}}_{\text{BS}}(t)$.

4: **if** $\Lambda_p < \Lambda_c \ \& \ \rho_p > \rho_c$ **then**
5: Acquire $\bar{\mathbf{H}}_{\text{BS}}(t)$ using Equation (2.8) or Equation (2.9).
6: **else**
7: $\bar{\mathbf{H}}_{\text{BS}}(t) = \widehat{\mathbf{H}}_c^Q(t)$
8: **end if**

the prediction at the UE, the fewer bits would be needed to send the feedback. Thus, CSIFB-PNet can significantly reduce feedback overhead. Lastly, if the prediction at the UE is perfect, feedback is unnecessary; hence, feedback-related overhead is completely eliminated. This benefit can be attained in an environment where the UEs are static (e.g., in a stadium or moving at a predictable speed). This is because when the UEs are static, variations in the CSI can be very small; hence, prediction can be more accurate.

If the compression function, $Q(\cdot)$, uses an infinite amount of bits to send the feedback, i.e., $B_Q = \infty$, then there is no advantage of using CSIFB-PNet. In short, compression followed in the conventional approach and CSIFB-PNet will be the same, i.e., $Q(\bar{\boldsymbol{x}}) = \bar{\boldsymbol{x}}$, where $\bar{\boldsymbol{x}}$ is the input data for compression. In other words, we can say that there is no compression while sending feedback. Hence, the real benefits can be acquired when the CSI is highly compressed, which is true in a practical wireless environment, i.e., type-I and type-II CSI feedback used in 3GPP are highly compressed [18], as explained in Chapter 1.

2.4 CSI Prediction Models

Strictly speaking, CSI prediction predicts future CSI realizations by exploiting past observations. For CSI prediction, two statistical algorithms, i.e., autoregressive (AR) and parametric models, have been addressed in [62] and [63], respectively. Both models are based on the statistical modeling of a wireless channel. Broadly, a parametric model esti-

mates Doppler shift, the number of scattering resources, angles of arrival and departure, and amplitude. The assumption is that a wireless channel is a superposition of a finite number of complex sinusoids. Finally, CSI is predicted based on estimated parameters. However, estimated parameters can expire quickly in a highly dynamic environment; therefore, iterative re-estimation of parameters is required, increasing computation cost [63]. In contrast, the AR model approximates the wireless channel as an AR process [64], where the future CSI is extrapolated using a weighted linear combination of current and past CSI [7]. The downside of AR is its vulnerability to impairments, e.g., additive noise, making the AR model infeasible. Therefore, seeing the promising benefits of ML in Chapter 1, we utilize ML for CSI prediction [2]. In the rest of this section, we highlight the channel prediction models[2] that we used in our study [9]. They are divided into RNN, BiLSTM, and a hybrid model. In the following, we explain these models.

2.4.1 RNN for CSI Prediction

With the rapid advancements in ML algorithms, their applications have been considered in almost every field. For example, RNN, a class of ML, is a powerful technique for time-series predictions [65]. RNNs differ from standard feed-forward NN since their training and prediction are based on current and past information. In other words, their decisions are influenced by past observations too. Contrarily, a standard NN's output is independent of previous input, which may not be the case in many scenarios, e.g., consider correlated fading channels in wireless communications. Channel realizations, determined by the physical propagation of electromagnetic waves in the medium, show a large correlation in the time domain. Therefore, RNN can be the right ML tool to predict CSI [2].

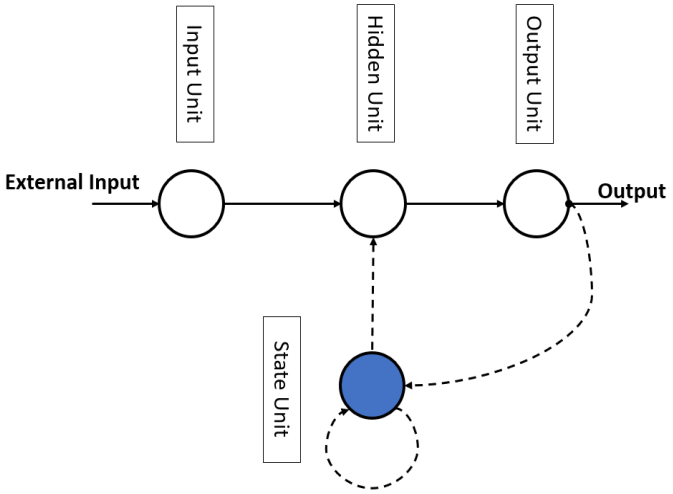


Figure 2.3: The topology of a simple *Jordan* RNN [66].

[2]The proposed models are used to predict the channel of single-UE, and we will discuss CSI prediction of multiple UEs in Chapter 4.

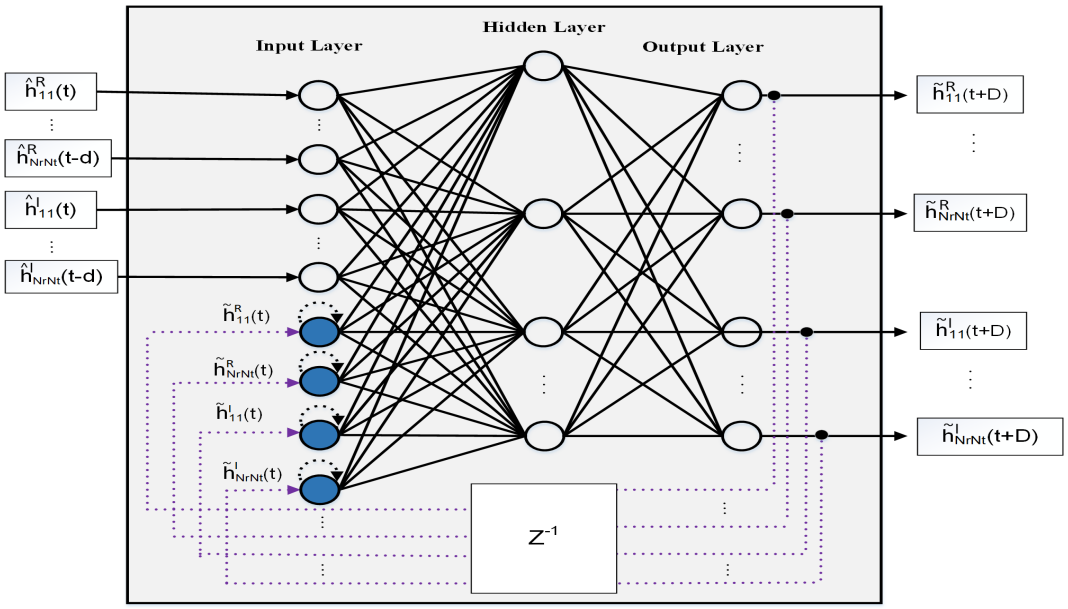


Figure 2.4: Pictorial representation of a fully connected *Jordan* RNN for massive MIMO channel prediction. The external inputs of the RNN are the real and imaginary parts of the compressed CSI. In contrast, the internal inputs (denoted with blue units) are the recurrent components. The output shows the multi-step predicted real and imaginary parts of compressed CSI for each antenna element.

The RNN has several variants [66], [67], among which we use a *Jordan network* [66]. The basic topology of a *Jordan network* is drawn in Fig. 2.3. The network consists of an input layer, a hidden layer, and an output layer. The layers are connected in a feed-forward configuration[3]. The hidden and output layers are also connected with a state unit (internal input); this recurrent connection of the internal input provides a short-term memory to the hidden unit. The hidden unit is influenced by external input, and its past state obtaining via internal input, which helps the RNN find a time-series relationship between the various inputs; thus, making it different from a standard NN.

By following the same structure of *Jordan network*, we reconstructed[4] a massive MIMO RNN, as drawn in Fig. 2.4, for the prediction of CSI [2]. The objective of the RNN is to forecast D-step values of CSI, denoted by $\widetilde{\mathbf{H}}(t+D)$, which are as close as possible to true values, i.e., $\widehat{\mathbf{H}}^{Q}(t+D)$. Here we explain the CSI prediction models with respect to $\widehat{\mathbf{H}}^{Q}$, however, models are also trained using $\widehat{\mathbf{H}}$ as followed in one-sided CSIFB-PNet. We assume the total (external and internal) numbers of neurons in the input layer as I_u, the hidden layer as J_u, and the output layer as K_u. In terms of massive

[3]It is important to note that *Jordan* used the terms plan unit and state unit for external and internal input, respectively, in his paper [66]. However, we use them for the reader's understanding as external and internal input.

[4]A similar configuration is also followed in [68] for SISO configuration.

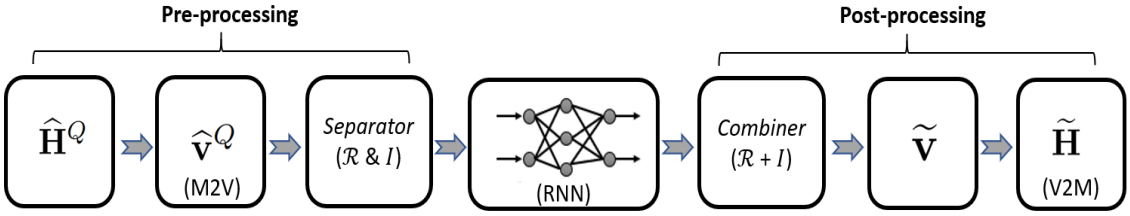


Figure 2.5: Pre and post-processing of CSI matrix for training and prediction using the RNN. The real and imaginary parts of CSI are represented by $\mathcal{R}$ and I, respectively.

MIMO channel prediction, the input of the RNN is the compressed channel, written as

$$\widehat{\mathbf{H}}^Q(t) = \begin{bmatrix} \widehat{h}^Q_{11}(t) & \cdots & \widehat{h}^Q_{1N_t}(t) \\ \widehat{h}^Q_{21}(t) & \cdots & \widehat{h}^Q_{2N_t}(t) \\ \vdots & \ddots & \vdots \\ \widehat{h}^Q_{N_r1}(t) & \cdots & \widehat{h}^Q_{N_rN_t}(t) \end{bmatrix}, \tag{2.10}$$

where $\widehat{h}^Q_{n_rn_t}(t) \in \mathbb{C}^{1\times 1}$ is the compressed estimated channel between n_t^{th} and n_r^{th} transmit and receive antennas, respectively. Along with the compressed channel, $\widehat{\mathbf{H}}^Q(t)$, its d-step delayed versions, denoted by $[\widehat{\mathbf{H}}^Q(t-1), \widehat{\mathbf{H}}^Q(t-2), \cdots, \widehat{\mathbf{H}}^Q(t-d)]$, are also fed as an external input to the RNN. RNN's total external input is given as

$$\widehat{\mathbf{H}}^Q = \big[\widehat{\mathbf{H}}^Q(t), \widehat{\mathbf{H}}^Q(t-1), \widehat{\mathbf{H}}^Q(t-2), \cdots, \widehat{\mathbf{H}}^Q(t-d)\big]. \tag{2.11}$$

The classic RNN does not take matrices as input; hence pre-processing is required. Below we provide some pre-processing steps, which are also depicted in Fig. 2.5. First, the combined input $\widehat{\mathbf{H}}^Q$ is converted from matrix-to-vector (M2V) form as

$$\widehat{\mathbf{v}}^Q = \mathrm{M2V}(\widehat{\mathbf{H}}^Q) = \big[\widehat{h}^Q_{11}, \quad \widehat{h}^Q_{12}, \quad \cdots, \quad \widehat{h}^Q_{N_rN_t}\big], \tag{2.12}$$

which represents a vector having complex entries. Currently, the ML algorithms are not well implemented for complex values [69]; therefore, we use real-valued RNNs in our study. For this purpose, the above complex vector can be converted into separate real and imaginary parts through a dedicated function; we call it *separator*; the outcome can be represented in a vector form as

$$\widehat{\mathbf{v}} = \big[\widehat{h}^R_{11}(t), \quad \cdots, \quad \widehat{h}^R_{N_rN_t}(t-d), \widehat{h}^I_{11}(t), \quad \cdots, \quad \widehat{h}^I_{N_rN_t}(t-d)\big], \tag{2.13}$$

where $\widehat{h}^R_{n_rn_t} \in \mathbb{R}^{1\times 1}$ and $\widehat{h}^I_{n_rn_t} \in \mathbb{R}^{1\times 1}$ denote[5] the real and imaginary parts of a complex channel $\widehat{h}^Q_{n_rn_t} \in \mathbb{C}^{1\times 1}$. These inputs can be seen in Fig. 2.4, where real and imaginary

[5]For notational convenience, we removed the superscript Q, which depicts the compressed channel.

parts are fed as an external input to the RNN. Besides, the recurrent[6] (feedback or internal inputs) components for time t, as depicted with blue units (or neurons) in Fig. 2.4, are expressed in the vector form as

$$\widetilde{\mathbf{v}}(t) = \left[\widetilde{h}^{R}_{11}(t), \quad \cdots, \quad \widetilde{h}^{R}_{N_r N_t}(t), \widetilde{h}^{I}_{11}(t), \quad \cdots, \quad \widetilde{h}^{I}_{N_r N_t}(t)\right], \tag{2.14}$$

where $\widetilde{h}^{R}_{n_r n_t}(t)$ and $\widetilde{h}^{I}_{n_r n_t}(t)$ show the real and imaginary parts of the channel, $\widetilde{h}_{n_r n_t}(t) \in \mathbb{C}^{1\times 1}$, predicted at time t, which acts as a recurrent component for the next iteration. The above vector is fed into RNN internally, which can be seen as feedback in Fig. 2.4. The total inputs, i.e., external and internal, of the RNN can be represented in a vector form as

$$\mathbf{q}(t) = \left[\widehat{\mathbf{v}}, \widetilde{\mathbf{v}}(t)\right]. \tag{2.15}$$

In a nutshell, by feeding the total input into the RNN, the D-step ahead prediction of CSI can be written as

$$\begin{aligned}\widetilde{\mathbf{h}}^{R,I}(t+D) = \big[&\widetilde{h}^{R}_{11}(t+D), \quad \cdots \quad ,\widetilde{h}^{R}_{N_r N_t}(t+D),\\ &\widetilde{h}^{I}_{11}(t+D), \quad \cdots \quad ,\widetilde{h}^{I}_{N_r N_t}(t+D)\big].\end{aligned} \tag{2.16}$$

Next, by applying the post-processing steps, shown in Fig. 2.5, first, we combine the real and imaginary parts using a function called *combiner*. Hence, the complex predicted channel vector is given as

$$\widetilde{\mathbf{h}}(t+D) = \left[\widetilde{h}_{11}(t+D), \quad \cdots \quad ,\widetilde{h}_{N_r N_t}(t+D)\right]. \tag{2.17}$$

Last, by performing vector-to-matrix (V2M) conversion, the above vector can be written into the matrix form as

$$\widetilde{\mathbf{H}}(t+D) = \begin{bmatrix} \widetilde{h}_{11}(t+D) & \cdots & \widetilde{h}_{1N_t}(t+D) \\ \widetilde{h}_{21}(t+D) & \cdots & \widetilde{h}_{2N_t}(t+D) \\ \vdots & \ddots & \vdots \\ \widetilde{h}_{N_r 1}(t+D) & \cdots & \widetilde{h}_{N_r N_t}(t+D) \end{bmatrix}, \tag{2.18}$$

which represents the massive MIMO predicted channel at time step $t+D$. Besides, $\widetilde{h}_{n_r n_t}(t+D) \in \mathbb{C}^{1\times 1}$ denotes the predicted channel for the n_t^{th} transmit and n_r^{th} receive antennas.

The prediction of the RNN is based on the weights (generally denoted with w) of input and hidden layers and the applied activation function to the linear equation. In summary, the weight, w, is randomly chosen for the connection between the output of a neuron in the predecessor layer and the input of a neuron in the successor layer. Let us denote the weight of i^{th} input and j^{th} hidden neuron as w_{ji}, and w_{kj} depicts the weight for the connection between k^{th} output neuron and j^{th} hidden neuron, where $1 \leq i \leq I_u$,

[6]To get a deeper understanding of the recurrent block, the interested readers can refer to [66].

$1 \leq j \leq J_u$, and $1 \leq k \leq K_u$. The output of a neuron is calculated by applying the activation function to the linear equation (combination of input and weight value); such activation function introduces non-linearity. Generally, four common activation functions are used in the ML domain: sigmoid, hyperbolic tangent, linear, rectified linear, leaky rectified linear, and threshold. Experiments showed that the hyperbolic tangent function gives the best result. However, addressing the results of different activation functions is not the objective of our work, and the choice of activation function is problem-dependent. The chosen activation function is expressed as

$$\textit{tanh}(\beta) = \frac{e^{\beta} - e^{-\beta}}{e^{\beta} + e^{-\beta}} \,. \tag{2.19}$$

where β is the value on which activation is applied, and it can be written as

$$\beta = \mathbf{w}_j \cdot \mathbf{q}(t) \tag{2.20}$$

where $\mathbf{w}_j = [w_{j1}, \cdots, w_{jJ_u}]$ represents the weights of hidden layer. The output of the j^{th} hidden layer neuron, at time instant t, can be calculated as

$$o_j(t) = \textit{tanh}\left(\mathbf{w}_j \cdot \mathbf{q}(t)\right) \,. \tag{2.21}$$

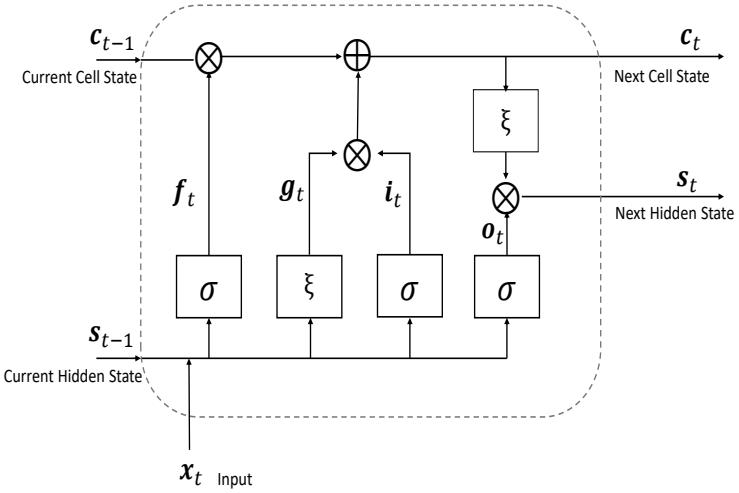
Consequently, the above activation function is sent as an input to the next layer, which in our case, is the output layer (see Fig. 2.4). In summary, the D-step ahead predicted output at the k^{th} output neuron can be written as

$$O^k(t+D) = \sum_{j=1}^{J_u} w_{kj} \cdot o_j(t) \,. \tag{2.22}$$

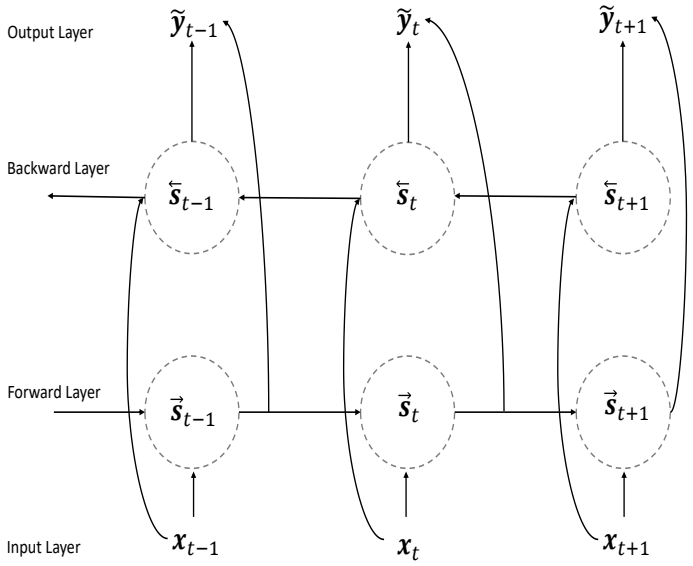
In the massive MIMO RNN model, the above output represents the real or imaginary part of the n_t^{th} and n_r^{th} transmit and receive antennas, respectively.

2.4.2 BiLSTM for CSI Prediction

BiLSTM is composed of two independent long short-term memory (LSTM) networks. In a BiLSTM model, information is learned from both ends of the input data vector, which results in better prediction performance than traditional unidirectional LSTM. LSTM is an advanced version of RNN. During model training, RNNs suffer from vanishing and exploding gradient problems in backpropagation. In the training phase, the gradient often becomes smaller and smaller when the backpropagation algorithm advances backwards, i.e., from the output to the input layer of RNN. By the time gradient reaches the input layer, it sometimes results in zero, not updating the weights of RNN; consequently, RNN cannot learn. Contrarily, exploding gradient turns to increase the gradient, which results in increasing weights of RNN; hence, gradient descent diverges. To this end, an LSTM-based NN was developed by Hochreiter and Schmidhuber [70]. The core idea of an LSTM is the introduction of a memory cell and multiplicative *gates*, which regulate the



(a) Single-cell of LSTM.



(b) Fully connected BiLSTM-based NN.

Figure 2.6: Graphical illustration of time-series NNs, i.e., LSTM and BiLSTM. Fig. 2.6a shows single-cell of an LSTM, where input is real-valued CSI realization for time instant t. Fig. 2.6b depicts a fully connected BiLSTM-based NN, where inputs are learned in two ways. Circular shapes given in Fig. 2.6b denote a single-cell of LSTM, which is given in Fig. 2.6a. The outputs of BiLSTM portray predicted CSI realizations. Source: [9].

flow of information (see Fig. 2.6a). Briefly, *forget gate* decides the amount of information to be stored in the cell by utilizing the current input, $\mathbf{x}_t$, and the output of the previous

LSMT cell, denoted by $\mathbf{s}_{t-1}$. Mathematically,

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{V}_f \mathbf{s}_{t-1} + \mathbf{b}_f) \,, \tag{2.23}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the *sigmoid* activation function, $\mathbf{W}$ and $\mathbf{V}$ are the weight matrices, $\mathbf{b}$ is the bias vector, subscript f is associated with the *forget gate*. The *input gate* determines the amount of information to be added into cell state $\mathbf{c}_{t-1}$ by exploiting $\mathbf{x}_t$ and $\mathbf{s}_{t-1}$. Mathematically,

$$\begin{aligned} \mathbf{i}_t &= \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{V}_i \mathbf{s}_{t-1} + \mathbf{b}_i) \,, \\ \mathbf{g}_t &= \textit{tanh}(\mathbf{W}_g \mathbf{x}_t + \mathbf{V}_g \mathbf{s}_{t-1} + \mathbf{b}_g) \,, \end{aligned} \tag{2.24}$$

where subscript i and g are associated with *input gate*. By utilizing *input* and *forget gates*, LSTM can determine the amount of information to be retained and removed. Finally, the *output gate* calculates the output of an LSTM cell by using an updated cell state, $\mathbf{c}_t$, and $\mathbf{x}_t$; the resultant output, given below, is then passed to the next LSTM cell of the network:

$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{V}_o \mathbf{s}_{t-1} + \mathbf{b}_o) \,. \tag{2.25}$$

As a result of the operations above, few information is dropped and a few is added, this updates the next long-term state as follows:

$$\mathbf{c}_t = (\mathbf{f_t} \otimes \mathbf{c}_{t-1} + \mathbf{i}_t \otimes \mathbf{g}_t) \,, \tag{2.26}$$

where $\otimes$ represents Hadamard product. Lastly, short-term memory state, $\mathbf{s}_t$, is calculated by passing long-term memory, $\mathbf{c}_t$, through *output gate* as

$$\mathbf{s}_t = \mathbf{o}_t \otimes \xi(\mathbf{c}_t) \,. \tag{2.27}$$

In BiLSTM architecture, as depicted in Fig. 2.6, input information is learned in two directions, i.e., left-to-right (forward layer) and right-to-left (backward layer). Importantly, notation $t+1$ in BiLSTM architecture is only used for the illustration purpose, such indexes are based on passed CSI observations. The output information of each direction, denoted by $\vec{s}$ and $\overleftarrow{s}$, respectively, is passed simultaneously to the output layer, where predicted output is calculated as

$$\widetilde{\mathbf{y}}_t = \vec{\mathbf{s}}_t \otimes \overleftarrow{\mathbf{s}}_t \,. \tag{2.28}$$

2.4.3 Hybrid Model for CSI Prediction

In the hybrid model, as shown in Fig. 2.7, we utilize an RNN-based CP, explained in Section 2.4.1, NP, explained in the following subsection. In the beginning, the dataset is cleaned, i.e., to check if there are corrupted/duplicate/missing data in the dataset. Additionally, data splitting is performed, in which the dataset is divided into three sets,

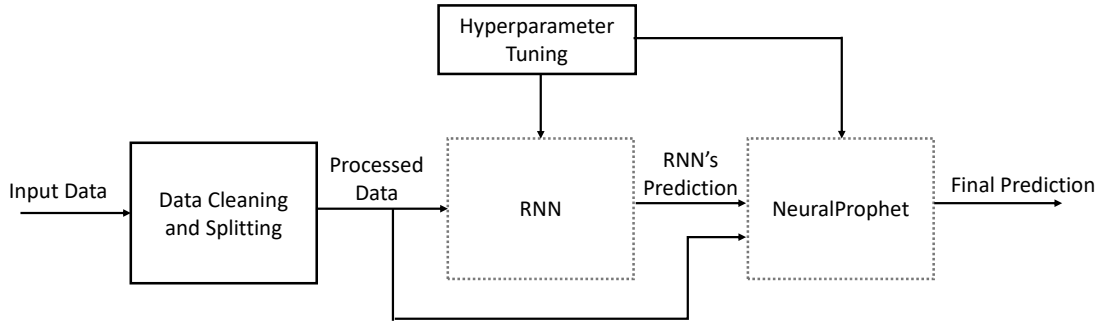


Figure 2.7: Flow diagram of the hybrid model, where dotted boxes show the prediction models used for the hybrid model. Processed data is the real-valued CSI realizations, and final prediction denotes the multi-step ahead predicted CSI realizations. Source: [9].

i.e., training, validation, and testing. Further, ML models transform the input sequences into an acceptable format. The processed dataset is fed to the input of RNN and NP. Then, we consider the predicted channel vector of RNN[7], which is fed to NP along with input feature vector[8]. NP learns to correct the predicted output with RNN's prediction. Later in Section 2.6, we demonstrate that NP can predict output more precisely when used with RNN and outperforms all standalone models, notably BiLSTM, for different cases. In the following, we explain the working functionality of NP.

2.4.3.1 NeuralProphet

Within a short span of time, NeuralProphet (NP) has emerged as a promising choice for different time-series prediction tasks [71]. Several time-series models, e.g., RNN and BiLSTM, have been developed in the context of DL. However, their internal functioning is still a question mark despite demonstrating promising results. In contrast, NP is an explainable and scalable prediction framework composed of statistical models, e.g., AR.

It is sometimes important to analyze the performance of a prediction model in the form of different components. DL-based models are difficult to interpret due to their black-box nature. Contrarily, NP is composed of different components[9], where each component contributes additively to predicted output, and their behaviour is well interpretable. Further, each component is composed of individual inputs and modelling methodology. The output of each component is D-step ahead future CSI realizations. For the notational convenience, we summarize the model for $D = 1$ and $\widetilde{h}_{n_r n_t}(t) \in \mathbb{R}^{1\times 1}$.

[7]It is, however, important to mention that the predicted output of BiLSTM can also be considered. But for the sake of lower computational complexity of the hybrid model, we used RNN, as BiLSTM is computationally more expensive than RNN. The rationale behind this is two-way learning of BiLSTM and use of *gates*, e.g., *input*, *forget*.

[8]In all CPs, input features, and corresponding labels are same.

[9]In the documentation of NP, it is composed of six components. However, concerning our application of CSI prediction, we dropped a few as some of them are irrelevant for CSI prediction, e.g., holidays. For more details, interested readers can refer to [71].

In the context of CSI prediction, the predicted value of NP for time instant t can be written as [71]

$$\widetilde{h}_{n_r n_t}(t) = R_t + A_t \tag{2.29}$$

where R_t represents the trend function for the input data [71], and A_t is the AR effect for time t based on previous CSI realizations [64]. The trend function, R_t, captures the overall variation in the input data. It tries to learn the points where clear variation in the data occurs; these points are called change-points, represented by $\{n_1, n_2, \ldots, n_{\mathsf{m}}\}$, composed of a total of m change-points (tuned using *grid search* [72]). A trend function can be expressed as

$$R_t = (\zeta^0 + (\mathbf{\Gamma}_t)^\dagger \boldsymbol{\zeta}) \cdot t + (\rho^0 + (\mathbf{\Gamma}_t)^\dagger \boldsymbol{\rho}) \,, \tag{2.30}$$

where $\boldsymbol{\zeta} = \{\zeta^1, \zeta^2, \ldots, \zeta^{\mathsf{m}}\}$, and $\boldsymbol{\rho} = \{\rho^1, \rho^2, \ldots, \rho^{\mathsf{m}}\}$, are the vectors of growth rate and offset adjustments, respectively, and $\boldsymbol{\zeta} \in \mathbb{R}^{\mathsf{m}\times 1}$ and $\boldsymbol{\rho} \in \mathbb{R}^{\mathsf{m}\times 1}$. Besides, ζ^0 and ρ^0 are the initial growth rate and offset values, respectively. And, $\mathbf{\Gamma}_t = \{\Gamma_t^1, \Gamma_t^2, \ldots, \Gamma_t^{\mathsf{m}}\}$, where $\mathbf{\Gamma}_t \in \mathbb{R}^{\mathsf{m}\times 1}$, which represents whether a time t is past each change-point. For a m^{th} change-point, Γ_t^m is defined as

$$\Gamma_t^j = \begin{cases} 1, & \text{if } t \geq n_m \\ 0, & \text{otherwise} \end{cases} . \tag{2.31}$$

A classic AR process of order d can be modeled as

$$A_t = \Omega + \sum_{e=1}^{d} \theta_e \cdot A_{t-e} + \epsilon_t \,, \tag{2.32}$$

where Ω and ϵ_t are the intercept and white noise, respectively, and θ are the coefficients of AR process. The classic AR model can only make one-step ahead prediction, and to make multi-step ahead prediction, D distinct AR models are required to fit. To this end, we utilize feed-forward NN along with AR (built-in feature of NP), termed as *AR-Net* [73], to model AR process dynamics. NP-based *AR-Net* can produce multi-step future CSI realizations by using one AR model. *AR-Net* mimics a classic AR model, with the only difference of data fitting. *AR-Net* is a feed-forward NN that maps AR model. In the *AR-Net*, d last observations of CSI realizations are given as input $\mathbf{z} = \{A_{t-1}, \cdots, A_{t-d}\}$, which are processed by the first layer and then passed through each hidden layer. Correspondingly, D-step ahead future CSI realizations, denoted by $\widetilde{\mathbf{z}} = \{A_t, A_{t+1}, \ldots, A_{t+D}\}$, can be obtained at the output layer. Mathematically, $\widetilde{\mathbf{z}}$ is obtained as

$$\begin{aligned} \boldsymbol{\omega}_1^{\mathsf{out}} &= \alpha(\mathbf{U}_1 \mathbf{z} + \mathbf{b}_1^{\mathsf{NP}}) \,, \\ \boldsymbol{\omega}_{\mathsf{i}}^{\mathsf{out}} &= \alpha(\mathbf{U}_{\mathsf{i}} \boldsymbol{\omega}_{\mathsf{i}-1}^{\mathsf{out}} + \mathbf{b}_{\mathsf{i}}^{\mathsf{NP}}) \,, \qquad \text{for} \quad \mathsf{i} \in [2, 3, \ldots, l] \\ \widetilde{\mathbf{z}} &= \mathbf{U}_{l+1} \boldsymbol{\omega}_l^{\mathsf{out}} \,, \end{aligned}$$

Figure 2.8: Image of the Nokia campus in Stuttgart, Germany. The transmit array's location (resp. boresight direction) is marked on the left by a blue bar (resp. blue arrow). The measurement tracks along which the vehicle (mimicking a UE) was moved are depicted by black lines. The corresponding arrowheads and numbers indicate the direction of movement and the measurement track number. Source: [2].

where $\alpha(\cdot)$ is the *rectified linear unit* (ReLu) activation function, written as

$$\alpha(\gamma) = \begin{cases} \gamma, & \gamma \geq 0 \\ 0, & \gamma < 0 \end{cases} . \tag{2.33}$$

Further, l is the number of hidden layers having n_h hidden units in each layer, $\mathbf{b}^{\text{NP}} \in \mathbb{R}^{n_h \times 1}$ is the vector of biases, $\mathbf{U} \in \mathbb{R}^{n_h \times n_h}$ is the weight matrix for hidden layers, except for the first $\mathbf{U}_1 \in \mathbb{R}^{n_h \times d}$ and last $\mathbf{U}_{l+1} \in \mathbb{R}^{D \times n_h}$ layers. In the AR component of NP, an important selection parameter is the order of AR, i.e., d, which is hard to select in practice. In general, d is chosen such that $d = 2D$, i.e., twice the number of the prediction horizon.

The operation of an ML-based channel predictor is divided into two phases: training and prediction. After selecting the hyperparameters (e.g., learning rate, optimization algorithm), the training phase can be started in the first phase. In this phase, channel values are given at the input of CP, along with corresponding output labels. Consequently, the CP processes each sample (or a batch of samples depending on the batch size) and compares the predicted value with the true label. The weights are updated through backpropagation, which is repeated until a certain convergence condition, e.g., mean-squared error (MSE), is minimized.

2.5 Measurement Campaign

Massive MIMO channel measurement campaign (see Figs. 2.8, 2.9) was conducted on the Nokia campus in Stuttgart. In that area, the buildings of approximately 15 meter height are primarily arranged along streets, acting as reflectors and blockers for

(a) Image of Track-1 of the measurement campaign.

(b) BS antenna with 64 antenna elements is shown here.

(c) A trolley, mimicking a UE, is moving on track-1.

Figure 2.9: A closer view of measurement track-1 and network entities is shown in these figures. Buildings of approximately 15 m height can also be seen, acting as reflectors and blockers for the radio waves. Source: [3].

the radio waves from the transmit antenna array, which was placed on the rooftop of one of these buildings. The environment resembles well a NLoS urban-micro-like propagation scenario. The geometry of 64-element transmit array had been adapted to the propagation scenario; i.e., 4 rows with 16 (single-pol.) patch antennas each were used, with a horizontal antenna spacing of $\lambda/2$, and a vertical separation of λ.

The array antennas transmitted 64 time-frequency orthogonal pilot signals at 2.18 GHz carrier frequency, using OFDM waveforms that followed 10 MHz LTE numerology (i.e., 600 subcarriers with 15 kHz spacing). The pilot signals had been arranged such that the sounding on 50 separate subbands (each consisting of 12 consecutive subcarriers) required 0.5 ms. Within that pilot burst period, the propagation channel was assumed to be time-invariant. The pilot bursts were sent continuously with a periodicity of 0.5 ms.

The receiver cart (mimicking a UE), as shown in Fig. 2.9, consisted of a single monopole antenna mounted at 1.5 meter height, a Rohde & Schwarz TSMW receiver and a Rohde & Schwarz IQR hard disc recorder, which continuously captured the received baseband signal. Both the transmit array and the receiver were frequency synchronized via global positioning system. During the measurements, the receiver cart was moved along several routes at walking speed (5 kmph), which corresponds to a spatial channel sampling distance of less than 0.1 mm.

2.6 Results and Analysis

By exploiting the dataset given in Section 2.5, this section evaluates the performance of the proposed work. The results are divided into two parts, where the first focuses on the performance of CSIFB-PNet and later on the proposed CSI prediction models.

To train CPs, we used dataset of track-1 (shown in Figs. 2.8, 2.9), which is composed of 116k consecutive CSI realizations, i.e., $\{\mathbf{H}(t)|t = 1, \ldots, 116\text{k}\}$. The dataset (normalized) is passed through necessary pre-processing and formatting steps using custom-built input pipelines to be parsed through each ML model easily. Further, the dataset of track-1 is divided into three parts: training (80%), validation (10%), and test (10%). Similarly, the 10% dataset of track-2 and track-6 is taken randomly for the test. Unless stated otherwise, the performance of CSIFB-PNet is evaluated on track-2, and simulation parameters are given in Table 2.1. The training process starts from an initial state, where weights and biases are randomly initialized. At the t^{th} time iteration, pre-processed[10] channel vector is fed as an input to CPs. To train CPs, we use Huber loss as a cost function, which is defined as

$$L_{\text{huber}}(\mathbf{H}, \widetilde{\mathbf{H}}) = \begin{cases} \frac{1}{2\tau}(\mathbf{H} - \widetilde{\mathbf{H}})^2, & \text{for } |\mathbf{H} - \widetilde{\mathbf{H}}| \leq \tau \\ |\mathbf{H} - \widetilde{\mathbf{H}}| - \frac{\tau}{2}, & \text{otherwise} \end{cases} \tag{2.34}$$

where $\mathbf{H}$ and $\widetilde{\mathbf{H}}$ are the true labels and predicted CSI realizations, respectively. It is important to highlight that the CPs are trained on compressed and true channels,

[10] Real and imaginary parts of channel are separated, delayed channel realizations and corresponding labels are created, etc.

Table 2.1: Parameters of CSIFB-PNet

Parameter	Value	Parameter	Value
$\{l, \tau\}$	$\{4, 1\}$	$\{\lrcorner, J_u\}$(NP)	$\{0.001, 32\}$
Epochs	50	$\{d, D\}$	$\{20, 10\}$
m	50	$\{\lrcorner, J_u\}$(RNN & BiLSTM)	$0.001, 200$

e.g., CSIFB-PNet2 exploited compressed CSI, and CSIFB-PNet1 exploited true CSI. For brevity, here we write the cost function with respect to true CSI. $\mathbf{H}$ will be replaced by $\widehat{\mathbf{H}}_c^Q$ in case of exploiting compressed CSI. By using Huber loss as a cost function, a batch of 32 samples is fed into each CP, the predicted outcome is compared with true labels, and error is backpropagated to update weights and biases using *adaptive moment estimation* (*Adam*) as an optimizer [74]. The training iterations are repeated until the cost function goes below a threshold value. Furthermore, open-source libraries such as *TensorFlow*, *Keras*, and *Scikit-learn* are used for the implementation of CPs.

Selection of optimal training parameters, e.g., learning rate $(\lrcorner)$, number of hidden layers (l), and hidden neurons (J_u), play a pivotal role in enhancing prediction accuracy. As a model does not directly learn hyperparameters, we manually define them before fitting the model. Therefore, a well-known automated strategy, that is, *grid search* [72], is used for hyperparameter tuning. For NP, *linear growth* function and 95% *change-point range* gave the best results [71]. The remaining tuned parameters for NP and other CPs are listed in Table 2.1. Also, the standard NP model only takes uni-variate data to produce its output. In our hybrid model, we adapted the standard NP model for multivariate data by adding an extra future regressor into the NP model and using multiple parallel models depending on the MIMO configuration. Future regressors are the variables which are known for the future. In our case, predictions of RNN are known to us, and we included this information as a future regressor of the NP model. Thus, we feed both RNN predictions and true sequences as multivariate inputs to the NP model to enhance its prediction capability. Besides, in RNN and BiLSTM models, the dropout layer is adapted, which drops hidden units randomly with a probability of 0.2, to prevent over-fitting.

2.6.1 Performance of CSIFB-PNet

In this subsection, we compare CSIFB-PNet with the benchmark schemes by using various evaluation parameters, which are given below. We utilize RNN as a channel predictor with $D = 1$; later in Section 2.6.2, we will address the results of proposed CPs with different values of D. Furthermore, to compare the performance of CSIFB-PNet with the benchmark schemes, we present the results using the one-sided CSIFB-PNet methodology. Later, we give a comparison between the one-sided and two-sided CSIFB-PNet.

2.6.1.1 NMSE

To observe the effectiveness of CSIFB-PNet, we use normalized mean-squared errorr (NMSE), denoted by Υ^{FB}, which is the difference between the true channel and the acquired channel at the BS. Mathematically,

$$\Upsilon^{\text{FB}} = \mathsf{E}\left\{ \frac{\left\| \mathbf{H} - \bar{\mathbf{H}}_{\text{BS}} \right\|_{\text{FRO}}^2}{\|\mathbf{H}\|_{\text{FRO}}^2} \right\} , \tag{2.35}$$

where $\bar{\mathbf{H}}_{\text{BS}}$ denotes the acquired channel at the BS. Importantly, in the case of CSIFB-PNet, the acquired channel at the BS is the one given in Equation (2.9), whereas in the case of the conventional approach, the acquired channel at the BS is a compressed estimated channel, i.e., Equation (2.3). In some results, Υ^{FB} is expressed in decible (dB), calculated as $10\log_{10}(\Upsilon^{\text{FB}})$.

2.6.1.2 Precoding Gain

As the feedback CSI is used for precoding, we also measure the performance using precoding gain. Let us denote the acquired or reconstructed complex channel vector at the BS as $\bar{\mathbf{h}}_{\text{BS}}$ and true complex channel vector as $\mathbf{h}$. By using this information, the equivalent channel can be represented as

$$\mathbf{h}_{\text{eq}} = \left(\frac{\bar{\mathbf{h}}_{\text{BS}}}{\left\| \bar{\mathbf{h}}_{\text{BS}} \right\|_2} \right)^* \times \left(\frac{\mathbf{h}}{\|\mathbf{h}\|_2} \right) . \tag{2.36}$$

By using (2.36), precoding gain is calculated as

$$\Psi^{\text{FB}} = \mathsf{E}\left\{ (\mathbf{h}_{\text{eq}})^* \times \mathbf{h}_{\text{eq}} \right\} . \tag{2.37}$$

2.6.1.3 Cosine Similarity

To measure the quality of the precoding vector, we consider the cosine similarity, denoted by ϱ^{FB}, of the two vectors, calculated as

$$\varrho^{\text{FB}} = \cos\theta = \mathsf{E}\left\{ \frac{|(\bar{\mathbf{h}}_{\text{BS}})^* \cdot \mathbf{h}|}{\left\| \bar{\mathbf{h}}_{\text{BS}} \right\|_2 || \ \mathbf{h} \ ||_2} \right\} . \tag{2.38}$$

Fig. 2.10 shows the gain of CSIFB-PNet1 over the conventional approach, where Υ^{FB} is plotted against different numbers of overhead bits used to feedback CSI. It can be seen that CSIFB-PNet1 provides a gain of approximately 1.00 under high compression. However, a smaller gain is observed when the feedback overhead is increased. For example, the gain is low when $B_Q = 5$, i.e., low compression. In short, using massive feedback bits brings no advantage, which verifies our remark from Section 2.3.3. The feedback followed in 3GPP is highly compressed; therefore, CSIFB-PNet1 is beneficial. In a nutshell, CSIFB-PNet1 reduces OTA feedback overhead and improves acquired CSI accuracy at the BS when feedback CSI from the UE is massively compressed.

Cosine similarity is another important evaluation parameter plotted in Fig. 2.11 for both schemes. The results show that CSIFB-PNet1 gives cosine similarity of 0.95 by using

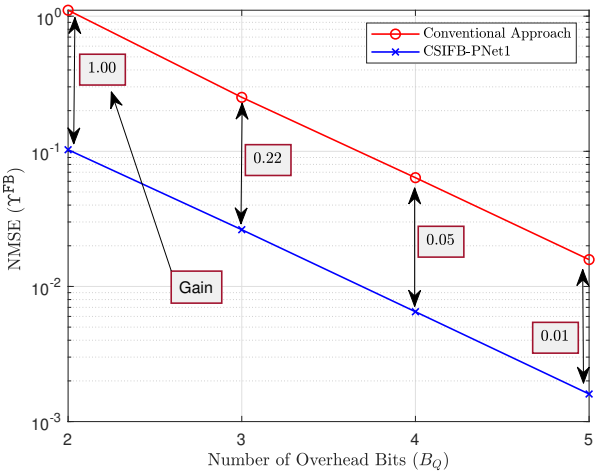


Figure 2.10: Comparison of conventional CSI feedback approach vs. CSIFB-PNet1 with different overhead bits used to feedback CSI.

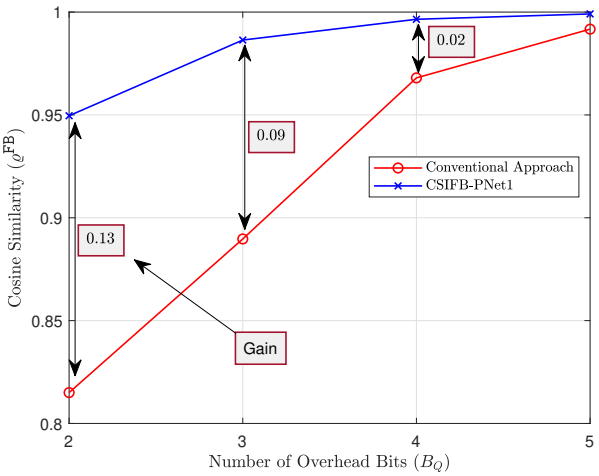


Figure 2.11: Performance evaluation using cosine similarity.

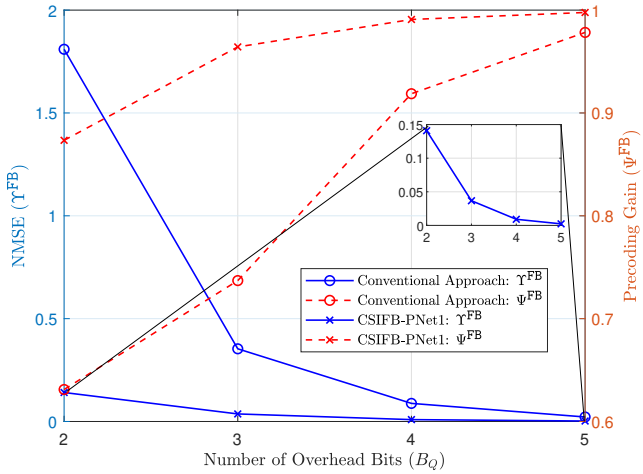


Figure 2.12: Robustness of CSIFB-PNet1. Performance evaluation on track-6 of the measurement campaign.

2 overhead bits. In contrast, the conventional approach achieves a cosine similarity of approximately 0.82. Similar to Fig. 2.10, the cosine similarity gain also reduces between the two schemes with an increased overhead bit. It can be concluded from the results that CSIFB-PNet1 improves CSI accuracy by approximately 16% when CSI is highly compressed. And the gain reduces to approximately 3%, when CSI feedback is sent with high overhead, i.e., 4 overhead bits.

To verify the robustness of CSIFB-PNet1, Fig. 2.12 reveals the performance using NMSE and precoding gain on track-6, which is away from the trained track. The trend of Υ^{FB}, plotted in blue color on the left y-axis, shows that CSIFB-PNet1 provides a huge gain of approximately 1.66 under low overhead, i.e., $B_Q = 2$. And the gain reduces with the increase in overhead. Thus, CSIFB-PNet1 is highly beneficial when feedback information is massively compressed. In contrast, a similar gain can be observed for precoding, plotted in red on the right y-axis. For instance, when $B_Q = 2$, then CSIFB-PNet1 has $\Psi^{\text{FB}} = 0.87$ whereas the conventional approach reaches CSIFB-PNet1 by using 5 overhead bits.

To compare CSIFB-PNet1 with state-of-the-art, i.e., autoencoder-based CSI feedback, called as CsiNet in [35], Fig. 2.13 shows the comparison in terms of NMSE and cosine similarity. We used a pre-trained CsiNet [75] to have a fair comparison. Accordingly, we have trained CSIFB-PNet1 on the same dataset as utilized in [75]. The dataset is generated using Matlab 5G toolbox function *nrCDLChannel*, composed of 15k samples. To get further details of the dataset and pre-trained CsiNet, please refer to [75]. We compare the performance of CsiNet and CSIFB-PNet1 by using different overhead bits used to compress the CSI[11]. In Fig. 2.13, we plot NSME and cosine similarity

[11] In CsiNet, generated codewords by the encoder are compressed. In CSIFB-PNet1, an update function, i.e., the difference between prediction and estimation, is compressed.

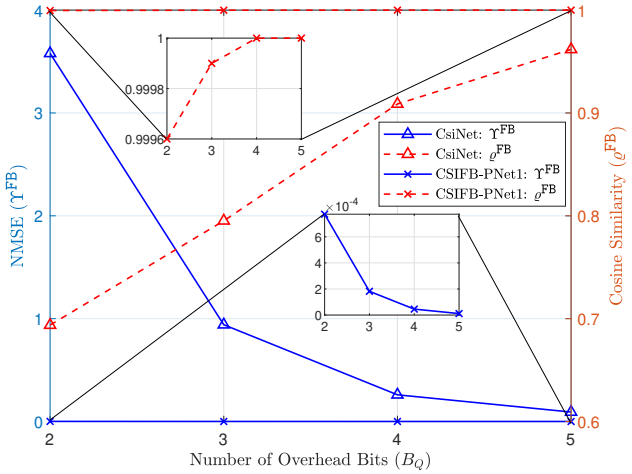


Figure 2.13: Comparison of CSIFB-PNet1 with CsiNet [35], [75].

between the recovered and the true channel. NMSE is plotted on the left y-axis in blue color, and cosine similarity is plotted on the right y-axis in red color. It is evident from the results that CSIFB-PNet1 recovers the CSI using only 2 overhead bits, and CsiNet requires 6 bits predictably. The rationale behind this is CSIFB-PNet1 compresses the update function (see Equation (2.7)); hence, less quantization noise. A negligible NMSE shows that feedback from the UE can be completely eliminated as the prediction is aligned with the estimated channel at UE. On the other hand, CsiNet compresses the generated codewords, through which the decoder network cannot recover the CSI. During the simulations, it is observed that without compressing the codewords, CsiNet gives a similar performance as CSIFB-PNet1. The presented results corroborate that CSIFB-PNet1 outperforms CsiNet.

Fig. 2.14 reveals the performance of one-sided CSIFB-PNet and two-sided CSIFB-PNet with the number of overhead bits. The results show that CSIFB-PNet1 outperforms CSIFB-PNet2. This is because of CSIFB-PNet1 training on the uncompressed CSI, which is done at the UE. Thus, a more accurate CSI prediction model is obtained at the UE, improving performance. Numerically, we observed a performance gain of approximately $3.25\,\mathrm{dB}$ using CSIFB-PNet1 when the feedback CSI is highly compressed, i.e., $B_Q = 2$. The issue with CSIFB-PNet1 is the reporting of the trained model to BS. In Chapter 3, we will address a solution to overcome this issue.

In Table 2.2, a comprehensive comparison is given by using the dataset from the measurement campaign (see Section 2.5). The best results are presented in bold font. CsiNet is retrained using the dataset of track-1, where training parameters follow their default setting in [75] and transmitted codewords are not compressed. It can be observed that CSIFB-PNet1 outperforms benchmark schemes. For instance, CSIFB-PNet1 obtains the lowest NMSE values and outperforms benchmark schemes at all compression levels. In comparison to CSIFB-PNet2, CSIFB-PNet1 also provides significant gains due

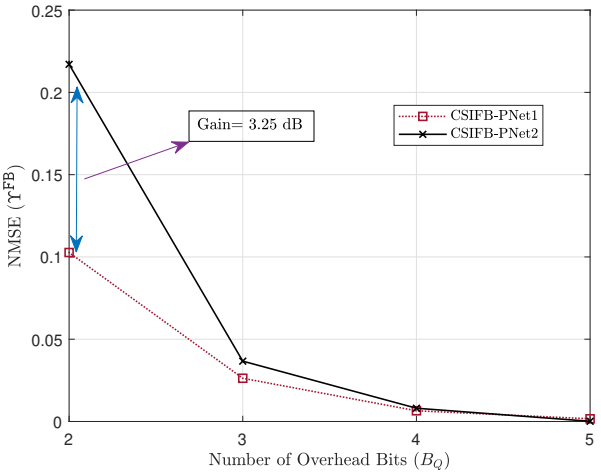


Figure 2.14: One-sided versus two-sided CSIFB-PNet. The dataset of track-2 is used.

to the sophisticated ML architecture and utilizing uncompressed CSI at the UE to train channel predictors. When the number of overhead bits is increased, benchmark schemes give similar results to CSIFB-PNet1, showing that they require more overhead. Further, CsiNet performs better than the conventional approach under low overhead, and conventional performs better under high overhead cost. The reason for the worst performance of CsiNet in high overhead bits is that it is not well trained on the measurement campaign, and perhaps more effort is required to train CsiNet; nevertheless, even well-trained CsiNet has lower performance in comparison to CSIFB-PNet1 (see Fig. 2.13). In a nutshell, we have learned from the presented results that CSIFB-PNet gives the best performance with having low overhead cost.

2.6.2 Performance of CSI Prediction Models

The performance of four CSI prediction models, i.e., NP, RNN, BiLSTM, and hybrid model, is evaluated using NMSE and cosine similarity. NMSE of a CP is defined as

$$\Upsilon^{\text{CP}} = \mathsf{E}\left\{ \frac{\left\| \mathbf{H} - \widetilde{\mathbf{H}} \right\|_{\text{FRO}}^2}{\|\mathbf{H}\|_{\text{FRO}}^2} \right\} . \tag{2.39}$$

Similarly, by exploiting $\mathbf{H}$ and $\widetilde{\mathbf{H}}$, cosine similarity of a CP, denoted by ϱ^{CP}, is computed.

Fig. 2.15 shows the comparison of four CPs in terms of NMSE (Υ^{CP}). Particularly, performance is evaluated on three tracks unseen by the CPs during the training phase. The trend reveals that RNN when used standalone, does not perform well; hence, it gives the worst performance. On the other hand, BiLSTM outperforms RNN because BiLSTM can retain information in their memory for longer periods and learns the input data in both directions, thereby performing better. In contrast, the hybrid model performs better than other CPs. The rationale is that NP learns better when RNN's predicted values

Table 2.2: Performance comparison using different evaluation parameters and benchmark schemes

B_Q	Method	Υ^{FB} (dB)	Ψ^{FB}	ϱ^{FB}
2	Conventional	0.44	0.66	0.81
	CSIFB-PNet1	$\mathbf{-9.88}$	$\mathbf{0.90}$	$\mathbf{0.94}$
	CSIFB-PNet2	-6.63	0.81	0.90
	CsiNet [35], [75]	-2.46	0.44	0.66
3	Conventional	-6.00	0.79	0.88
	CSIFB-PNet1	$\mathbf{-15.80}$	$\mathbf{0.97}$	$\mathbf{0.98}$
	CSIFB-PNet2	-14.35	0.96	0.97
	CsiNet [35], [75]	-2.46	0.44	0.66
4	Conventional	-11.95	0.93	0.96
	CSIFB-PNet1	$\mathbf{-21.86}$	$\mathbf{0.99}$	$\mathbf{0.99}$
	CSIFB-PNet2	-20.93	0.98	0.98
	CsiNet [35], [75]	-2.46	0.44	0.66
5	Conventional	-18.02	0.98	0.98
	CSIFB-PNet1	-27.87	$\mathbf{0.99}$	$\mathbf{0.99}$
	CSIFB-PNet2	$\mathbf{-38.92}$	$\mathbf{0.99}$	$\mathbf{0.99}$
	CsiNet [35], [75]	-2.46	0.44	0.66

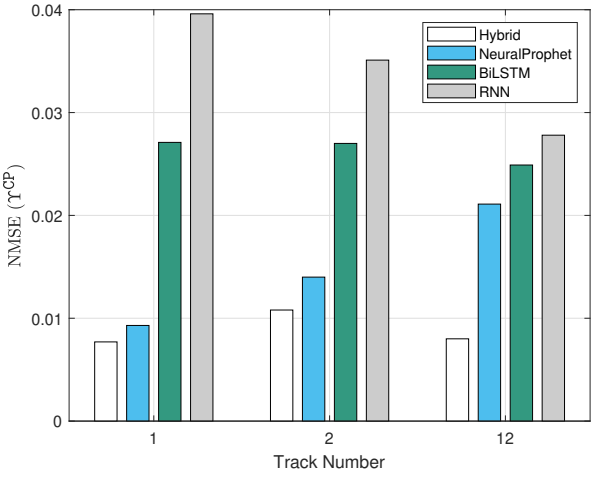


Figure 2.15: Performance of the CPs on different tracks followed by UE, where NMSE is independent apropos track number as each track has different channel strength. The number of predicted CSI realizations is $D = 10$, and uncompressed CSI is used for prediction.

Table 2.3: Performance comparison of different CPs under various parameters

$B_Q \downarrow$	**Model** $\rightarrow$	**Hybrid**		**NP**		**BiLSTM**	
	Horizon $\rightarrow$	$D = 10$	$D = 1$	$D = 10$	$D = 1$	$D = 10$	$D = 1$
∞	Υ^{CP}	$\mathbf{0.0077}$	0.0022	0.0093	$\mathbf{0.0014}$	0.0271	0.0052
	ϱ^{CP}	$\mathbf{0.9975}$	0.9993	0.9974	$\mathbf{0.9996}$	0.9923	0.9988
5	Υ^{CP}	$\mathbf{0.0337}$	0.0081	0.0413	$\mathbf{0.0080}$	0.0398	0.0111
	ϱ^{CP}	$\mathbf{0.9895}$	0.9971	0.9864	$\mathbf{0.9972}$	0.9874	0.9964
3	Υ^{CP}	0.2188	0.0267	0.1550	$\mathbf{0.0259}$	$\mathbf{0.0502}$	0.0281
	ϱ^{CP}	0.9416	0.9904	0.9435	$\mathbf{0.9907}$	$\mathbf{0.9838}$	0.9902

are passed as future regressors. For instance, on track-1, there is approximately 80% reduction in NMSE when the hybrid model is used compared to RNN.

Table 2.3 presents a detailed comparison of CPs, where the best results are written in bold numbers. The results are obtained using a test dataset of track-1 and for different prediction horizons, i.e., D, and under different overhead bits used to compress CSI. NMSE (Υ^{CP}) and cosine similarity (ϱ^{CP}) are the two evaluation parameters used to analyze the performance of CPs. The results show that NP is the best for small prediction horizons, i.e., $D = 1$, and under any compression level. However, for long prediction horizons, i.e., $D = 10$, the hybrid model is useful when the compression level is low, e.g., 5 or no-compression (∞). The rationale behind the superior performance of the hybrid model is the combination of RNN and NP. For instance, in the case of $D = 10$, NP had shown bad performance when used standalone, but when combined with RNN, it learned the information of RNN's prediction to improve accuracy. That is why the hybrid model has superior performance. Nonetheless, under high compression, i.e., 1 and 3 compression bits, and $D = 10$, BiLSTM is the best choice, which is because BiLSTM is suitable for long prediction horizons and handling non-linearities, thanks to the memory cell and multiplicative *gates* (see Section 2.4.2).

2.6.3 Computational Complexity of CSIFB-PNet

To measure the complexity of CSIFB-PNet, we calculate the number of complex multiplications required for the training and prediction of channel predictors. To get a single-step prediction of the massive MIMO channel, the hidden layer has to perform $I_u \times J_u$ complex multiplications, where I_u and J_u are the total number of neurons of the input and the hidden layer, respectively. For two hidden layers, $J_u \times J_u$ complex multiplications are performed. The complex multiplications at the output layer are $J_u \times K_u$, where K_u are the total output neurons. Hence, the total complex multiplications are

$$X = J_u \times (I_u + J_u + K_u)\,. \tag{2.40}$$

The total number of input neurons, I_u, depends on the number of massive MIMO sub-channels, recurrent components, and the delayed channel matrix. Hence, I_u is calculated

as

$$
\begin{aligned}
I_u &= \underbrace{N_r \cdot N_t}_{\widehat{\mathbf{H}}_c^{Q}(t)} + d \cdot \underbrace{(N_r \cdot N_t)}_{\widehat{\mathbf{H}}_c^{Q}(t-d)} + \underbrace{N_r \cdot N_t}_{Z^{-1}} \\
&= (d+2) \cdot N_r \cdot N_t \, .
\end{aligned} \tag{2.41}
$$

Conversely, the output layer's neurons are equivalent to massive MIMO sub-channels; therefore, $K_u = N_r \cdot N_t$. By substituting Equation (2.41) and $K_u = N_r \cdot N_t$ into Equation (2.40), X can be written in the simplified form as

$$
\begin{aligned}
X &= J_u \times [(d+2) \cdot N_r \cdot N_t + J_u + N_r \cdot N_t] \\
&= J_u \times \left[N_r \cdot N_t \left(d + 3 + \frac{J_u}{N_r \cdot N_t} \right) \right] \, .
\end{aligned} \tag{2.42}
$$

It is important to mention that the number of hidden layers and their neurons are dependent on the optimization problem; they are called hyperparameters of an ML algorithm. Generally, the same number of J_u is used in all hidden layers. Let us denote the size of massive MIMO configuration by $M = N_r \cdot N_t$ and the scale of the channel predictor by $C = 2dJ_u$. Then the one-step prediction complexity of the channel predictor is $\mathcal{O}(MC)$. The training complexity depends on the number of training examples (samples), denoted by N_s, and the number of training epochs, represented by N_e. Therefore, the training complexity of the channel predictor is $\mathcal{O}(MCN_sN_e)$. Besides, the total number of weight parameters for single-step prediction is 0.35 M. Furthermore, the number of FLOPs for single-step prediction is 7.8 M, which are very few in comparison to CsiNet (58.52 M) [36].

2.7 Conclusion

Motivated by the inaccurate CSI acquisition at the BS due to compression, this chapter addressed an ML-based massive MIMO CSI feedback mechanism coined as CSIFB-PNet. The presented results corroborated the validity of CSIFB-PNet. Specifically, the evaluated results using empirical data and the 3GPP channel model showed that CSIFB-PNet can reduce OTA overhead and effectively recover the compressed CSI in comparison to benchmark schemes. Furthermore, feedback is not necessary for a 3GPP channel model as the prediction at the UE is approximately the same as the estimated channel. The results showed the effectiveness of CSIFB-PNet compared to benchmark schemes. Experimental evaluations on track-2 of the measurement campaign demonstrated an increase of approximately 23% in precoding gain than without ML (conventional approach) when 2 bits are used to provide feedback information, and CSIFB-PNet2 is exploited. This gain increases to 36% when one-sided CSIFB-PNet is used. Similarly, an approximately 60% decrease in NMSE is observed for long-range uncompressed CSI prediction when the performance of the hybrid model is compared to the advanced time-series model, i.e., BiLSTM.

Chapter 3

CSI Feedback Enhancement in Multiuser

3.1 Introduction

In the previous chapter and particularly in the literature work, ML/DL has been opted at the BS and UE to improve the precision of acquired CSI. However, ML/DL implementation at UE can be infeasible for various reasons, such as UE power consumption. In addition to that, we have studied in the previous chapter that one-sided ML can help to enhance the CSI precision but at the cost of reporting the entire ML model, which includes a massive number of trained parameters and ML architecture etc. To overcome these issues, this chapter proposes a CSI learning mechanism at BS, called CSILaBS, to avoid ML at UE. Specifically, we exploit CP at BS to compute a light-weight PF. This PF is reported to UE for feedback evaluation. The use of such light-weight PF can reduce the computation cost of a UE and PF reporting overhead. Broadly, this chapter proposes an alternative scheme for massive MIMO CSI feedback that circumvent the need for power-hungry ML implementation at the UE while maintaining the overall ML gains. Besides, in a multiuser environment, we propose various mechanisms to select the feedback by exploiting PF while aiming to improve CSI accuracy. The major contributions of this chapter are:

- Motivated by the aforementioned issues, this paper considers a light-weight PF for CSI feedback overhead reduction and to improve acquired CSI accuracy. More specifically, assuming the limited power of a UE, we address the question: *How to remove ML from the UE while maintaining ML gains?*
- In light of the above question, we consider the implementation of CSI prediction at the BS for massive MIMO CSI feedback, where a light-weight PF is computed by exploiting predicted and reported CSI realizations. Later on, PF, reported by BS, can be used at the UE for feedback evaluation. Hence, training and storage of a fully functional NN can be eliminated at the UE. The reporting of PF can help

to create a set of matrices, i.e., a codebook, which can have a standardization impact. Thus, an index can be reported to avoid the transmission overhead of PF. To this end, we also cover possible standardization points of CSILaBS[1].

- By extending the work to multiuser, we address feedback selection methodologies to acquire more accurate CSI at the BS by exploiting PF at the UE. Particularly, we propose CSI feedback schemes, which are based on, for instance, *probabilistic* method, to avoid data loss because of collisions. Through simulations, we show that the proposed methodologies can effectively improve CSI precision when feedback is intelligently selected. We learn from the extensive simulations that the prediction error threshold is an important feedback evaluation parameter in a multiuser environment that can be carefully selected to improve CSI quality.

3.1.1 Chapter Organization and Notations

The rest of the chapter is organized as follows. In Section 3.2, the system model is presented. CSILaBS is detailed in Section 3.3. The proposed feedback selection methodologies are addressed in Section 3.4. Results are analyzed in Section 3.5. The conclusion is made in Section 3.6.

Throughout this chapter, matrices and vectors are represented by boldface upper and lower-case, respectively. Also, scalars are denoted by normal lower and upper-case. $\mathbb{R}$ and $\mathbb{C}$ denote the real and complex numbers, respectively. The superscripts $[\cdot]^\dagger$ and $[\cdot]^*$ denote the transpose, and the conjugate transpose of a matrix/vector, respectively. $\mathsf{E}\{\cdot\}$ denotes expectation operator, $\|\cdot\|^2_{\mathsf{FRO}}$ represents squared Frobenius norm, and $|\cdot|$ shows absolute value. $\mathbf{h}_k$ denotes the true channel for k^{th} UE, where $k = \{1, 2, \cdots, K\}$. The acquired and predicted channel at the BS for k^{th} UE are represented by $\bar{\mathbf{h}}_k^{\mathsf{BS}}$ and $\widetilde{\mathbf{h}}_k^{\mathsf{BS}}$,respectively. Similarly, the predicted and estimated channel at a UE is denoted by $\widetilde{\mathbf{h}}_k$ and $\widehat{\mathbf{h}}_k$, respectively. The matrix form of these vectors is represented in boldface upper-case.

3.2 System Model

Consider a massive MIMO cellular network, where a BS is serving using $M \gg 1$ transmit antennas to K single-antenna UEs. Without loss of generality, the received signal, per subcarrier and OFDM symbol, at the k^{th} UE can be expressed as

$$r_k(t) = \mathbf{h}_k(t)\boldsymbol{\nu}(t) + \varpi(t)\,, \tag{3.1}$$

where t is the time-index, $\mathbf{h}_k(t) \in \mathbb{C}^{M\times 1}$is the channel vector for k^{th} UE, $\boldsymbol{\nu}_k(t) = [\nu_1(t), \nu_2(t) \ldots, \nu_M(t)]^\dagger$ is the pilot vector, and $\varpi(t)$ is the noise. As discussed in the previous chapter, estimated $\mathbf{h}$ should be fed back to the BS, and it requires high OTA overhead. By assuming perfect CSI estimation, in the following, we address the CSI feedback scheme coined CSILaBS.

[1]UE can also perform ML training for various reasons, e.g., BS does not have a dataset for training ML algorithm(s). This approach is subject to UE being capable of running ML [3].

3.3 CSILaBS

This section details the proposed CSI feedback mechanism, CSILaBS, by exploiting the predicted channel. The core idea is to have the same PF at both ends, that is, at BS and at each UE. Such PF will be generated at the BS and reported to all UEs along with the CSI-RS. The UEs will compute an update function by exploiting PF and CSI-RS, which is then fed back to BS. Hence, a precise version of CSI can be acquired at the BS by using a smaller OTA overhead or zero-OTA overhead if the prediction is perfect. For the sake of simplicity, we explain CSILaBS for one UE, remarking that the same process will be followed by all UEs. CSILaBS provides an efficient and light-weight way for the UE and the BS to implement the same PF without having to report a massive number of ML weights, as followed in CSIFB-PNet2 (see Chapter 2). To implement CSILaBS, there are seven different stages: • Assessment • ML training at BS • PF reporting • PF verification • CSI estimation • CSI compression and feedback • CSI retrieval. We explain them in the following.

3.3.1 Assessment

In the first stage, BS informs the use of PF to UE and requests for the training data. The UE sends acknowledgment(ACK)/negative acknowledgement (NACK) for the availability of training data. If NACK is reported, then BS transmits CSI-RSs, which UE uses for channel estimation. Consequently, a compressed version of the estimated CSI is reported to BS. In the case of ACK, pre-existing training data can be reported to BS.

3.3.2 ML Training at BS

At the beginning of this stage, training data for ML-based CP is acquired at the BS. The training and prediction of CP have already been explained in Section 2.4 of Chapter 2. For brevity, let us assume that the predicted channel[2] at the BS for time instant t is denoted by $\widetilde{\mathbf{h}}_k^{\mathsf{BS}}(t)$, which we use to explain rest of the scheme.

3.3.3 PF Reporting

At this stage, BS reports a PF by exploiting the predicted channel from the previous stage. PF reporting can be of different forms, which we address below.

3.3.3.1 Model-Based Representation

This method evaluates a certain number of matrix-based models for channel evolution, and the matrix composition is returned. For example, an AR model of order d can be adopted as

$$\widetilde{\mathbf{h}}_k^{\mathsf{BS}}(t) = \sum_{e=1}^{d} \hat{\mathbf{F}}_{k,e}(Q[\widehat{\mathbf{h}}_k(t-e)])^{\dagger}\,, \tag{3.2}$$

[2]Recurrent neural network (RNN)-based CSI predictor is assumed in this chapter.

where transition matrix $\hat{\mathbf{F}}$ (a PF) is reported to UE, and $Q[\cdot]$ is a standard element-wise quantization/compression function. The reporting of the matrix (or set of matrices) can be done by using a codebook (similar to PMI reporting in type-I/II followed in the standards) and henceforth reporting only the index. For this purpose, predictor matrices codebook can be established through standards. The BS reports the index that approximates the set of matrices $\hat{\mathbf{F}}$.

3.3.3.2 Full Function Reporting

In this methodology, a NN is converted to an equation. This is simply done by writing the input-output relationship, considering weights and activation functions. The methods to convert NNs into equations are known in the literature and reported here for completeness.

$$\mathbf{Y} = \varkappa(\mathbf{W}\mathbf{X} + \mathbf{b}) \,, \tag{3.3}$$

where $\mathbf{W}$ and $\mathbf{b}$ represent the weights of a NN, $\mathbf{X}$ and $\mathbf{Y}$ depict the inputs (in the context of ML, input features of a NN) and outputs of a NN, respectively. Also, $\varkappa(\cdot)$ represents the activation function, which can be ReLu, hyperbolic tangent (*tanh*), sigmoid, etc. This is, however, a discouraged method, as the function itself can be too complex to be implemented and reported. Therefore, in this study, we consider this function reporting as a benchmark scheme, exploited in CSIFB-PNet1 explained in Chapter 2. To reduce reporting overhead, the function index can be reported as follows. A certain number of functions are reported into a codebook, which will be set at a standard level. A NN is converted into an equation, and the function in the set that is closest, i.e., giving the best prediction, to the NN is selected.

3.3.4 PF Verification

It is possible that the BS predicts the CSI with sufficient accuracy; however, the generated PF is not accurate due to, e.g., changing network conditions. Therefore, PF verification is essential before starting the CSI feedback mechanism. PF can be verified by reporting it to UE. For example, UE can compute the error based on reported PF and CSI-RS, using the former for CSI prediction and later for the latest CSI estimation. The error can be computed in the form of, for instance, MSE between the predicted and estimated CSI. The outcome can be sent to BS, which can retrain the NN if the error is sufficiently larger than a threshold value.

3.3.5 CSI Estimation

If PF is nearly accurate, the BS can utilize the predicted channel for CSI acquisition. To this end, BS transmits CSI-RS at, for example, time instant t, where UE can estimate the channel by exploiting CSI-RS. As we assume perfect channel estimation, therefore, in the following, we address CSI compression once the channel is estimated.

3.3.6 CSI Compression and Feedback

In CSILaBS, channel prediction at the UE is necessary to reduce CSI feedback overhead and improve CSI precision acquired at the BS. One of the possible ways is to use

ML at UE as well (see Chapter 2); however, it will increase computational cost. Also, an extra amount of information will be required to synchronize the ML algorithm used at both ends, i.e., BS and UE. To avoid this, we exploit reported PF, explained in Section 3.3.3. Let us assume that the estimated CSI at the k^{th} UE is denoted by $\widehat{\mathbf{h}}_k(t)$. Further, by exploiting the reported PF, explained in Section 3.3.3, predicted CSI at the k^{th} UE is represented by $\widetilde{\mathbf{h}}_k(t)$. Finally, CSI can be feedback to BS as

$$\bar{\mathbf{h}}_k(t) = Q\Big[f\{\widetilde{\mathbf{h}}_k(t), \widehat{\mathbf{h}}_k(t)\}\Big] , \tag{3.4}$$

where $f\{\cdot\}$ is an update function, which is simply a difference between the predicted and the estimated channel, as followed in Chapter 2. However, the channel prediction is based on a light-weight PF, which significantly reduces the computation cost at UE and reporting overhead of PF. The benefit of such compression is the reduction in OTA overhead to feedback CSI. For example, if $f\{\cdot\} = 0$, which is subject to perfect prediction, then the feedback-related overhead is eliminated; hence, UE does not need to feedback anything. Also, if $f\{\cdot\} \neq 0$ then quantization function, $Q[\cdot]$, will add less noise, thanks to $\widetilde{\mathbf{h}}_k(t)$. It is important to mention that the above equation can also serve as a verification for PF before compression due to the fact that UE can check if the error is sufficiently large.

3.3.7 CSI Retrieval

CSI for time instant t can be acquired at the BS by exploiting predicted CSI at BS, $\widetilde{\mathbf{h}}_k^{\mathsf{BS}}(t)$, and the reported update $\bar{\mathbf{h}}_k(t)$. Therefore, if $f\{\cdot\} \neq 0$, then BS recovers the CSI as

$$\bar{\mathbf{h}}_k^{\mathsf{BS}}(t) = \widetilde{\mathbf{h}}_k^{\mathsf{BS}}(t) + f^{-1}\{\bar{\mathbf{h}}_k(t)\} , \tag{3.5}$$

where f^{-1} represents inverse of f. In this study, $f^{-1} = -f$. In the second scenario, i.e., $f\{\cdot\} = 0$, BS does not receive anything, indicating that the predicted CSI at BS is perfect. Thus, the BS assumes that $\bar{\mathbf{h}}_k^{\mathsf{BS}}(t) = \widetilde{\mathbf{h}}_k^{\mathsf{BS}}(t)$.

3.3.8 Remarks on CSILaBS

The major advantage of CSILaBS is the elimination of ML at the UE. This helps the UE to use a light-weight PF to make CSI prediction; hence computation cost has been reduced. If the training data is unavailable at BS or does not have sufficient resources due to certain reasons, then UE can run ML at its end, similar to what is followed in Chapter 2, helping the BS to not do ML training. Furthermore, in the case of using ML at UE, UE will be responsible for reporting PF to BS for channel prediction at BS. In that case, PF verification, discussed in Section 3.3.4, will be done locally at UE, and BS has only to make predictions by exploiting reported PF. To reduce the reporting overhead of training data to BS, data augmentation algorithm(s) can be considered at BS to generate training data [76]. We shed light on reporting the training data to BS later in this chapter. CSILaBS also reduces feedback overhead if the prediction is not good enough and eliminates feedback if the prediction is perfect. Considering that

the accurate PF is achieved, CP at the BS can keep running in the background, using the newly acquired channel, $\bar{\mathbf{h}}_{\text{BS}}$, as an input. The CP will be updated accordingly or retrained if necessary. The BS will trigger an update in the PF whenever a threshold is surpassed, and the updated PF will be reported to UE, where codebook entries can be updated etc. This allows to track sudden changes in the ML model. For instance, if the underlying channel evolution changes due to, e.g., UE passing behind a building. In the following, we propose different algorithms to select the feedback in the multiuser environment.

3.4 CSI Feedback Selection

In the multiuser scenario of CSILaBS, feedback selection becomes crucial when the objective is to enhance the CSI feedback quality. In this section, we address the question: *How to efficiently select the feedback while improving the CSI accuracy of the overall system?* The problem of CSI feedback selection is similar to random access scheme, where a UE senses the *channel* before data transmission. For example, in a random access scheme, multiple users can transmit at the same time. In the case when two neighbouring users transmit at the same time then we say that their messages *collide* with each other, resulting in degrading the SINR of the overall system. Therefore, the users require a distributed medium access control (MAC) for efficient scheduling such that their messages do not *collide*. A plethora of research work has been proposed on random access protocols, e.g., carrier sense multiple access protocols, which is an important class of MAC protocols because of their simplicity, and have been widely used in IEEE 802.11 wireless-fidelity. In random access protocols, one potential solution is to use global dynamics. In global dynamics, the users transmit with a given probability, e.g., $\frac{\exp(\widetilde{\omega}_l(t))}{1+\exp(\widetilde{\omega}_l(t))}$ as given in [77], where $\widetilde{\omega}_l(t)$ is the weight of link l at time slot t. In the literature, random access-based approaches have proven to be throughput optimal [77], [78].

In CSI feedback selection, one may also see the problem as a random access scheme but the goal of our study is different, i.e., we do not transmit at a given rate to improve throughput. As indicated in the objective of this thesis that we aim to enhance the CSI feedback, therefore, we focus on using random access model to acquire precise CSI at the BS. In a nutshell, we use random access model in CSILaBS to reduce the errors due to compression imposed at the UE and to minimize OTA overhead cost. In the following, we propose various algorithms to select feedback.

3.4.1 Probabilistic Feedback

In this scheme, the feedback from each UE is evaluated with a certain probability, which depends on the error in the update function computed at the k^{th} UE. Without loss of generality, let us denote the set of available resource blocks to feedback the CSI by $\mathcal{N} = \{1, 2, \cdots, N\}$. In the probabilistic feedback, the k^{th} UE selects $n \in \mathcal{N}$ resource block randomly and evaluates the feedback in the selected resource block with

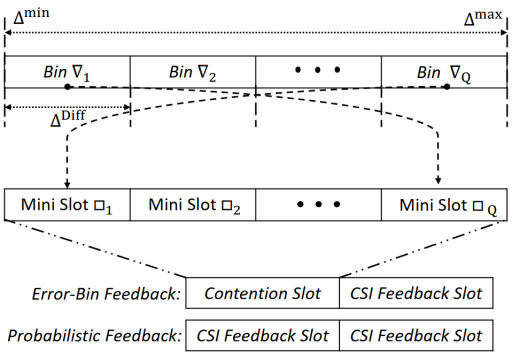


Figure 3.1: Pictorial representation of two feedback resources in probabilistic and error-bin feedback, where former utilizes both resources for CSI feedback and later uses one to check contention and second to feedback CSI of UE winning the contention.

a probability

$$\mathcal{P}_k^n = \frac{\exp(\Delta_k)}{1+\exp(\Delta_k)}\,, \tag{3.6}$$

where

$$\Delta_k = \mathsf{E}\big\{|\widetilde{\mathbf{h}}_k(t) - \widehat{\mathbf{h}}_k(t)|\big\} \tag{3.7}$$

is the error between the predicted and the estimated channel at the k^{th} UE. It can be observed from (3.6) that higher Δ_k will result in higher $\mathcal{P}_k^n$. This will be helpful to enhance CSI feedback as users having high error can be given priority so that an accurate version of CSI can be retrieved at the BS. Importantly, feedback at a UE is evaluated only when $\Delta_k > \wp$, where $\wp$ is the error threshold. The constraint $\Delta_k > \wp$ is put due to nature of (3.6), e.g., a user may select the feedback with 50% success rate even when there is no need to feedback CSI; in other words, when $\Delta_k = 0$. Later in Section 3.5, we will show that the optimal value of $\wp$ is an important design parameter for feedback selection, and that probabilistic feedback does help to enhance CSI feedback by using CSILaBS.

3.4.2 Error-Bin Feedback

In the probabilistic feedback, we have learned that UEs with high error must be selected more frequently; hence, we revised the random access methodology accordingly. In the error-bin method, we tackle a similar problem as followed in the random access schemes, i.e., avoiding *collisions* but our objective is to acquire CSI at the BS with high precision. Considering the objective of acquiring precise CSI at the BS, we propose to have a contention slot in addition with data slot or CSI feedback slot. The contention slot can be further divided into mini slots, where the objective of mini slots is to prioritize CSI feedback of UEs having high error. And the UEs winning the contention will be selected to feedback CSI. To illustrate error-bin feedback, we have drawn Fig. 3.1, where the k^{th} UE uses one resource to check contention and another to feedback CSI. The contention slot is composed of $\square_Q$ mini slots. Correspondingly, there are a total of ∇_Q bins, where

each bin is equally spaced Δ^{Diff} and the error window of all bins is predefined, ranging between Δ^{min} and Δ^{max}. For instance, Δ^{min} and Δ^{max} can be mapped between 0 and 1, respectively. And accordingly, $\Delta^{\text{Diff}} = 0.10$.

Focusing on the contention slot, the k^{th} UE verifies the constraint $\Delta_k > \wp$ and feedback 1 bit in a mini slot, depending on the error level. For example, if Δ_k lies in the first bin ∇_1, i.e., $0 < \Delta_k < 0.10$, then the UE transmits 1 bit in the last mini slot $\square_Q$ as error is minimum. Similarly, if Δ_k belongs to last bin ∇_Q, showing high error; the UE feedback 1 bit in the first mini slot $\square_1$. Pictorial illustration of these allocations is given in Fig. 3.1. We can learn from this allocation that UEs with high error will be given priority to feedback. Besides, in the case where two or more UEs transmit in the same mini slot, this indicates *collision*, but the advantage is we only lose 1 bit.

At the end of the contention slot, the UE winning the contention will send the CSI in the CSI feedback slot. In the scenario where two or more UEs win the contention, the UEs that transmitted in the earliest mini slots will be scheduled. This is because the UE transmitting in the earliest slot indicates that its error is high. Notably, the selection of winning UEs is dependent on the number of available data slots. For instance, if there are two data slots and three UEs win the contention, the first two UEs with the highest error will be selected, and the third UE will be dropped due to the unavailability of the resource block. In this way, we prioritize UEs having high error, consequently, CSI feedback performance can be enhanced. In contrast to error-bin feedback, probabilistic feedback uses all resources for CSI feedback, i.e., there is no contention slot (see Fig. 3.1).

3.4.3 Deterministic Feedback

In the deterministic feedback, all the UEs transmit at the same time irrespective of their error value and without any resource selection. Intuitively, there will be a high number of *collisions*. Hence, CSI feedback performance can not be enhanced as the BS will rely on open-loop CSI prediction, i.e., without any update from a UE. Later in Section 3.5, we will show that deterministic feedback performs the worst in comparison to error-bin and probabilistic feedback.

3.4.4 Periodic Feedback

This approach is used as a benchmark scheme to observe the performance gain of proposed CSI feedback mechanisms. Briefly, we consider the feedback by exploiting the conventional CSI feedback scheme, i.e., without ML, where the feedback from a UE is simply the compressed version of the estimated channel. In other words, there is no channel predictor in the network; thus, $\bar{\mathbf{h}}_k^{\text{BS}}(t) = Q[\widehat{\mathbf{h}}_k(t)]$. The feedback without ML is evaluated in a round-robin fashion, i.e., every UE feedback CSI on its turn without any error threshold limit. For this purpose, the total UEs K are divided into the available number of resources. For instance, if there are $N = 10$ resources and $K = 30$, then $K = 10$ UEs are selected in each round; hence, the first group of UEs will get the opportunity to retransmit in the fourth round and so on. In Section 3.5.2, we will discuss the results of above CSI feedback selection mechanisms.

3.5 Results and Analysis

In this section, we verify the effectiveness of CSILaBS. To give the comparison of CSILaBS with the CSIFB-PNet (proposed in Chapter 2), we utilize the same dataset as of Chapter 2, i.e., the dataset of measurement campaign performed at Nokia Bell-Labs. In the case of multiuser, the dataset is generated using MATLAB, considering that the underlying model is evolving using an AR process [64]. We compare the performance of CSILaBS with two other schemes, i.e., CSIFB-PNet and without ML. In the case of without ML, $\bar{\mathbf{h}}_k^{\mathsf{BS}}(t) = Q[\widehat{\mathbf{h}}_k(t)]$, which has been explained in Chapter 2, and we called it conventional approach. Besides, in CSILaBS and CSIFB-PNet, $\bar{\mathbf{h}}_k^{\mathsf{BS}}(t) = \widetilde{\mathbf{h}}_k^{\mathsf{BS}}(t) + f^{-1}\{\bar{\mathbf{h}}_k(t)\}$. For an abuse of notation, we denote $\bar{\mathbf{H}}^{\mathsf{BS}} \in \mathbb{C}^{M\times K}$ and $\mathbf{H} \in \mathbb{C}^{M\times K}$ as the matrix form of $\bar{\mathbf{h}}_k^{\mathsf{BS}}$ and $\mathbf{h}_k$, respectively, for all K UEs, and considering one-step ahead prediction. By using this information, precoding gain can be obtained as

$$\Psi^{\mathsf{FB}} = \mathsf{E}\left\{\mathsf{Trace}(\mathbf{H}_{\mathsf{eq}}^{*} \times \mathbf{H}_{\mathsf{eq}})\right\} , \tag{3.8}$$

where

$$\mathbf{H}_{\mathsf{eq}} = \left(\frac{\bar{\mathbf{H}}^{\mathsf{BS}}}{\left\|\bar{\mathbf{H}}^{\mathsf{BS}}\right\|_{\mathsf{FRO}}}\right)^{*} \times \left(\frac{\mathbf{H}}{\|\mathbf{H}\|_{\mathsf{FRO}}}\right) \tag{3.9}$$

is the equivalent channel. Similarly, by using $\bar{\mathbf{H}}^{\mathsf{BS}}$ and $\mathbf{H}$, NMSE, denoted by Υ^{FB}, is calculated as

$$\Upsilon^{\mathsf{FB}} = \mathsf{E}\left\{\frac{\left\|\bar{\mathbf{H}}^{\mathsf{BS}} - \mathbf{H}\right\|_{\mathsf{FRO}}^{2}}{\|\mathbf{H}\|_{\mathsf{FRO}}^{2}}\right\} . \tag{3.10}$$

Likewise, cosine similarity, denoted by ϱ^{FB}, is calculated by using $\bar{\mathbf{H}}^{\mathsf{BS}}$ and $\mathbf{H}$.

3.5.1 Comparison of CSILaBS with CSIFB-PNet

Fig. 3.2 presents the accuracy of the acquired channel at BS in the form of cosine similarity (ϱ^{FB}). The trend shows that the accuracy of the acquired channel increases with the number of overhead bits used to feedback CSI. However, CSILaBS brings a huge performance gain when feedback information is highly compressed, i.e., when 2 bits are used. Numerically, CSILaBS provides a gain of 11% in comparison to without ML. Thus, CSILaBS is beneficial in practical wireless communications environments as feedback evaluation methods, i.e., type-I/II, followed in the standards, are massively compressed due to exploitation of the codebook. Further, CSILaBS retains ML gain, i.e., CSILaBS gives nearly the same gain as using ML at both ends (CSIFB-PNet). Nevertheless, the benefit of CSILaBS is the elimination of ML training at UE, which can be costly in terms of UE's power consumption. Also, CSILaBS uses a light-weight PF at UE for feedback evaluation, and the overhead cost of PF reporting is small in comparison to CSIFB-PNet. For example, CSIFB-PNet requires 0.35 M parameters to report, whereas CSILaBS requires only $M \times M = 4096$.

Recalling that feedback CSI is used for precoding, we now inspect the performance of CSILaBS by using precoding gain, Ψ^{FB}, plotted on the right-side of Fig. 3.3. Furthermore,

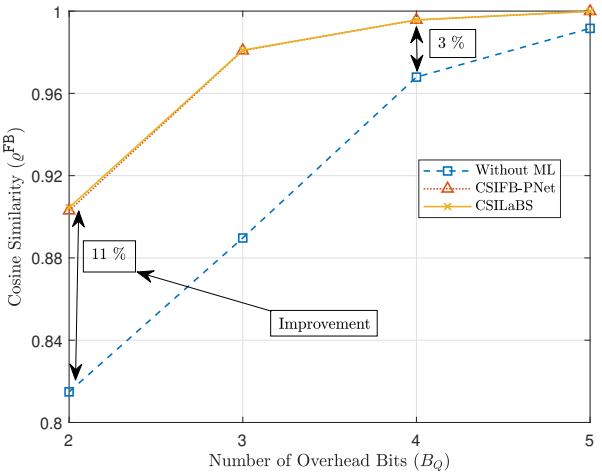


Figure 3.2: Cosine similarity (ϱ^{FB}) between the acquired channel at BS and corresponding true channel. The figure demonstrates that ϱ^{FB} increases with the number of overhead bits used to feedback CSI from UE to BS.

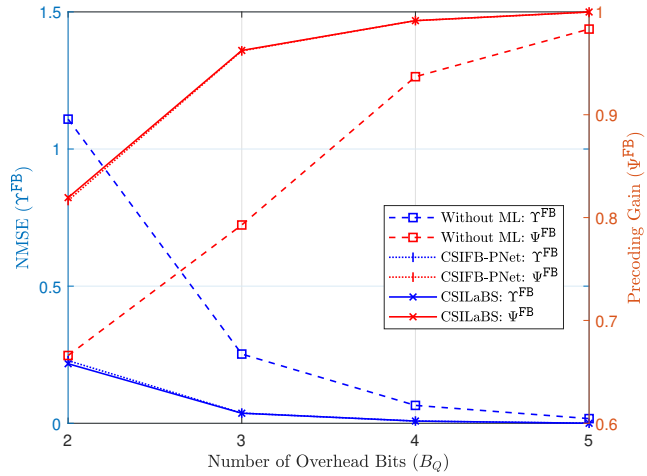


Figure 3.3: NMSE (Υ^{FB}) and precoding gain (Ψ^{FB}) of different CSI feedback schemes. Similar to Fig. 3.2, this figure reveals that the performance of feedback CSI improves with the number of overhead bits.

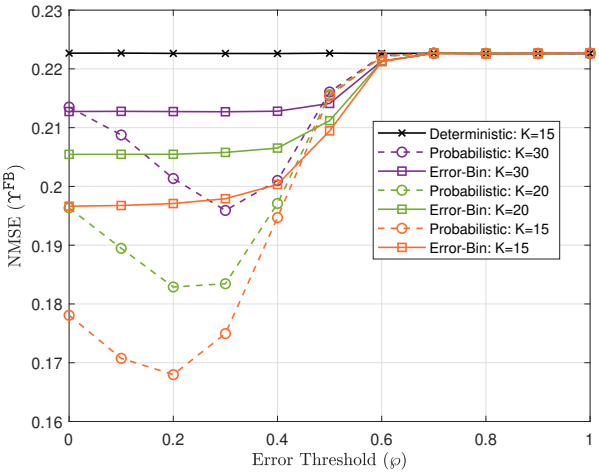


Figure 3.4: NMSE (Υ^{FB}) of different CSI feedback schemes versus the error threshold ($\wp$). A comparison is provided with various numbers of UEs (K). $N = 10$.

the comparison is made by using NMSE, Υ^{FB}, as an evaluation parameter. Fig. 3.3 reveals that the CSILaBS gives high precoding gain using fewer overhead bits. This gain reduces with the number of overhead bits. On the other hand, a similar trend can be observed for Υ^{FB}. Further, CSIFB-PNet has comparable performance with CSILaBS; nevertheless, the former requires implementation of ML at UE too.

3.5.2 Performance of CSILaBS under Feedback Selection

Fig. 3.4 reveals the performance of Υ^{FB} against the error threshold ($\wp$) selected to evaluate the feedback at a UE. It is important to mention that lower values of $\wp$ result in high numbers of feedback and vice versa. A comparison is provided with various numbers of UEs (K) and different feedback selection algorithms. First, we can observe that Υ^{FB} by using the deterministic method gives the worst performance, which is due to an excessive number of collisions. Hence, the BS relies on the predicted channel, i.e., $\bar{\mathbf{H}}^{\text{BS}} = \widetilde{\mathbf{H}}^{\text{BS}}$. Also, Υ^{FB} remains unchanged over different values of $\wp$, which is because $\bar{\mathbf{H}}^{\text{BS}} = \widetilde{\mathbf{H}}^{\text{BS}}$ is independent of $\wp$. In contrast, error-bin has slightly lower Υ^{FB}, which increases with $\wp$ as feedback frequency has reduced when $\wp$ is increased. Contrary to error-bin and deterministic feedback, probabilistic feedback shows the best performance. This is due to the fact that probabilistic feedback selects the feedback intelligently, i.e., when Δ and $\mathcal{P}$ are high. It can also be observed that with an appropriate value of $\wp$, CSI acquisition can be improved. For example, when $K = 30$, then $\wp = 0.30$ is the optimal choice to select the feedback using the probabilistic method. Numerically, we observe a gain of approximately 8.7% and 13% in comparison to error-bin and deterministic feedback, respectively. And this gain increases with the reduction in the number of UEs, as a lower value of K results in fewer collisions. Importantly, in the case of deterministic feedback, Υ^{FB} will remain constant despite different values of K; hence, we only plotted

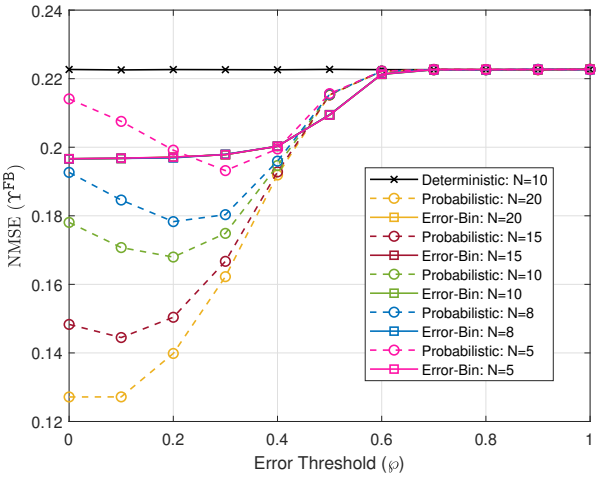


Figure 3.5: Comparison with various number of resource blocks (N). $K = 15$.

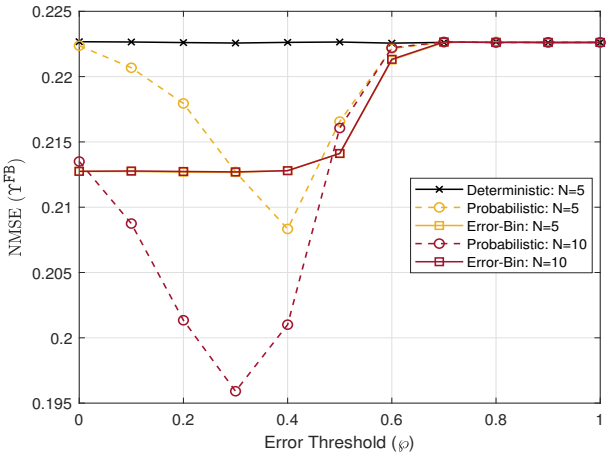
for $K = 15$ for illustration purposes.

Fig. 3.5 reveals the performance when the number of resources is varied and $K = 15$. The results show that probabilistic feedback outperforms. And the performance of probabilistic feedback improves with the number of resource blocks (N). However, in the case of error-bin, Υ^{FB} does not vary with the increase in the number of N, though it remains low for lower values of $\wp$. Besides, we can see that the optimal value of $\wp$ can be selected for probabilistic feedback depending on the available number of resources. For instance, when $N = 15$, then $\wp = 0.10$ is the optimal choice. In contrast, deterministic feedback is once again showing the worst performance.

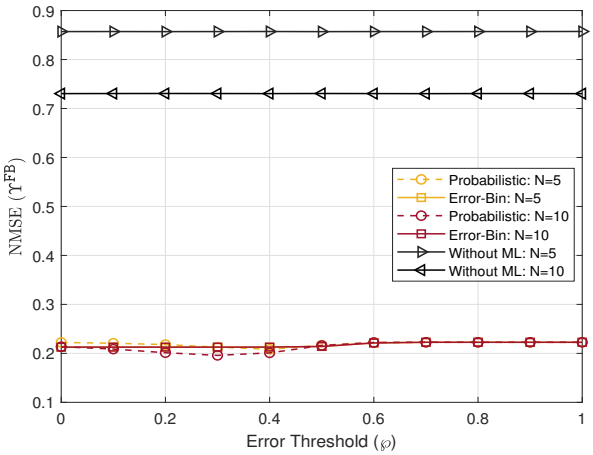
Fig. 3.6 shows the performance when the users are increased to $K = 30$ and when the performance is compared with periodic feedback, i.e., without ML as explained in Section 3.4.4. For illustration purposes, we have plotted the results of without ML on Fig. 3.6b. The results show that the proposed feedback schemes, which are based on channel predictors, give the best performance, as drawn on Fig. 3.6a.

3.6 Conclusion

This chapter investigated how ML can be avoided at UE for CSI feedback. In Chapter 2, ML has been considered at BS and UE, which makes it hard to train computationally expensive ML algorithm(s) at UE. In contrast, in this chapter, we addressed a novel method, coined CSILaBS, to efficiently recover compressed CSI without implementing ML at the UE. When applied ML at the BS, the generated function raised the idea of developing a codebook for downlink overhead reduction, which has shown highly standard-relevant implementation. The results showed the effectiveness of CSILaBS compared to benchmark schemes. The evaluations demonstrated an increase of approximately 23% in precoding gain than without ML when 2 bits are used to feed-



(a) Comparison of proposed schemes.



(b) Comparison with the benchmark.

Figure 3.6: A comparison of all CSI feedback schemes with the different number of resource blocks (N). $K = 30$.

back information. Furthermore, CSILaBS retains ML gains; that is, the performance of CSIFB-PNet and CSILaBS is approximately similar. In the case of the multiuser scenario, we observed that CSILaBS with the proposed probabilistic feedback method brings a gain of approximately 32% compared to deterministic feedback when $K = 15$ and $N = 10$ are considered.

Chapter 4

Role of RIS in CSI Prediction

4.1 Introduction

RIS, made of electromagnetic material, is capable of reconfiguring the properties of a wireless signal impinging upon it [79]–[82]. This feature can improve the performance of wireless PHY channel between transmitters and receivers, which usually suffers from severe fading, degrading the overall system efficiency. An intelligently configured RIS can help, for instance, to null out interference in a multiuser environment [83]. Implementation of RIS can be done through software-defined metasurfaces [80], composed of reconfigurable sub-wavelength-sized elements. Though RIS has been applied in various aspects of wireless communications, we focus on utilizing quasi-RIS for CSI prediction, as we have studied from Chapter 2 and Chapter 3 that channel prediction plays a pivotal role in enhancing CSI feedback.

Albeit ML has been utilized to improve the precision of channel prediction in Chapter 2, however requiring computationally complex and time-consuming implementation. For example, deployment of such an ML algorithm at a UE in CSIFB-PNet can be costly. In addition, in a highly time-varying channel, simultaneous prediction of the multiuser channel can become nearly impossible. In this chapter, we investigate a novel way to utilize RIS by asking the following question: *In a highly dynamic multiuser communication environment, is it possible to configure a quasi-RIS surface to make the channel predictable for desired prediction length?* The main outcome of this chapter shows that this is indeed possible.

The motivation of this chapter is to predict the entire multiuser composite channel for desired time slots by exploiting M RIS elements out of N, i.e., quasi-RIS. The benefit is the prediction of the whole composite channel arriving from N RIS elements; hence complexity and overhead for channel estimation can be reduced, which is the major bottleneck in RIS. Further, using fewer RIS elements, i.e., M, the remaining RIS elements can be utilized for other purposes, e.g., interference nulling in a multiuser environment. In some recent studies, e.g., [84], channel estimation in RIS is done using an exhaustive grid-search algorithm, which converts the composite channel from fast

to slow. Similarly, Doppler effect and multipath fading mitigation are achieved in [85], where Doppler frequency and RIS phases are assumed to be known. In [86], Doppler compensation is done in a non-RIS environment by formulating a compressive sensing problem, and the solution is obtained using Bayesian inference. We propose a completely different solution to mitigate fading effects by making the channel predictable using fewer RIS elements. The major contributions of this chapter are [5]:

- This chapter aims to enhance the performance of the channel predictor. To achieve this, we focus on maximizing the time correlation of the composite channel received at the UE by exploiting quasi-RIS.
- In a dynamic multiuser environment, we justify that the amalgamation of RIS and ML can help to minimize the performance loss of a channel predictor. Furthermore, to meet the desired objective function, we define different methodologies to partition the tuning part of RIS for multiuser.

4.1.1 Chapter Organization and Notations

The rest of this chapter is organized as follows. System model is presented in Section 4.2. Proposed mechanism is explained in Section 4.3. Section 4.4 focuses on results and Section 4.5 concludes the chapter.

Throughout this chapter, matrices and vectors are represented by boldface upper and lower-case, respectively. Also, scalars are denoted by normal lower and upper-case. $\widehat{\mathbf{H}}$, $\widetilde{\mathbf{H}}$, and $\mathbf{H}$, denote the RIS-assisted optimized composite channel, predicted channel by ML, and corresponding true channel. Furthermore, the notations $\mathbb{R}$ and $\mathbb{C}$ are representing the real and complex numbers, respectively. $\mathsf{E}\{\cdot\}$ denotes expectation operator, and$\|\cdot\|^2_{\mathsf{FRO}}$ represents squared Frobenius norm.

4.2 System Model

Consider a single-antenna BS and K single-antenna UEs communicating via RIS. We assume that direct-path between the BS and UEs is blocked. The RIS is a surface consisting of N elements. Without loss of generality, the received signal at the k^{th} UE and i^{th} time-index is expressed as

$$y_i^k = H_i^k x_i^k + n_i^k \,, \tag{4.1}$$

where $H_i^k = (\mathbf{g}_i^k)^* \mathbf{\Phi} \mathbf{g}_i^r \in \mathbb{C}$ shows the composite channel, which is phase dependent. $\mathbf{g}_i^r = [\mathsf{g}_i^{r,1}, \mathsf{g}_i^{r,2}, \ldots, \mathsf{g}_i^{r,N}] \in \mathbb{C}^N$ and $\mathbf{g}_i^k = [\mathsf{g}_i^{k,1}, \mathsf{g}_i^{k,2}, \ldots, \mathsf{g}_i^{k,N}] \in \mathbb{C}^N$ denote the channels of the link between BS and RIS and the link between RIS and k^{th} UE, respectively. x_i^k is the transmitted signal and $n_i^k \sim \mathcal{N}_{\mathbb{C}}(0, \sigma^2)$ is the noise variance. $\mathbf{\Phi} \triangleq \mathsf{diag}(e^{-j\phi_1}, e^{-j\phi_2}, \ldots, e^{-j\phi_N})$ is the diagonal phase shift matrix for RIS, where $\phi_n \in [0, 2\pi)$ is the phase shift of the n^{th} reflecting element.

4.3 Proposed Mechanism

The proposed mechanism aims to enhance the channel prediction performance by amalgamating RIS and ML. In particular, we utilize a portion of RIS elements to make the composite channel, H^k, predictable for desire time intervals while leaving remaining surface for other applications. As discussed in the previous section, the composite channel is dependent on the phases of N RIS elements; therefore, we optimize phases for $M < N$ selective RIS elements to make the channel predictable. The proposed mechanism is composed of two stages. In the first stage, we maximize prediction factor (Ω), which represents the time correlation among I consecutive time instances of the composite channel for $\{1, \cdots, K\}$ UEs. To define Ω, let us denote $\mathbf{H}_{\mathsf{a}} \in \mathbb{C}^{(I-1)\times K}$ as the composite channel for time instances $\{1, \cdots, I-1\}$ and K UEs. Also, let us represent one-step ahead version (in time) of the composite channel by $\mathbf{H}_{\mathsf{b}} \in \mathbb{C}^{(I-1)\times K}$ for time instances $\{2, \cdots, I\}$. Further, $\mathbf{\Omega} \in \mathbb{R}^{K\times K}$ represents the prediction factor matrix, and an element of $\mathbf{\Omega}$, denoted by $\Omega^{p,q}$, can be calculated as

$$\Omega^{p,q} = \frac{\sum_{i=1}^{I-1}\left[\left(v_i^p - \bar{v}^p\right)\left(v_i^q - \bar{v}^q\right)\right]}{\sqrt{\sum_{i=1}^{I-1}\left(v_i^p - \bar{v}^p\right)^2 \sum_{i=1}^{I-1}\left(v_i^q - \bar{v}^q\right)^2}}, \tag{4.2}$$

which represents the similarity between the p^{th} and q^{th} columns, $\mathbf{v}^p$ and $\mathbf{v}^q$, of $\mathbf{H}_{\mathsf{a}}$ and $\mathbf{H}_{\mathsf{b}}$, respectively, and $I-1$ is the length of each column. $v_i^{(\cdot)}$ represents an entry of column vector $\mathbf{v}^{(\cdot)}$, and $\bar{v}^{(\cdot)}$ denotes mean of $\mathbf{v}^{(\cdot)}$. Finally, $\Omega = \frac{1}{K}\mathsf{Trace}(\mathbf{\Omega})$ represents the prediction factor of observed CSI realizations, and $\mathsf{Trace}(\cdot)$ is depicting the trace of matrix $\mathbf{\Omega}$. Ω is maximized by intelligently configuring M RIS elements, explained in following subsection. In the second stage, the optimized composite channel is fed as an input of the ML algorithm for channel prediction. For instance, in the case of four instances, i.e., $L = 4$, of the composite channel, the fourth instance is predicted by an ML model while exploiting past three observations $(I = L - 1)$. To maximize Ω, we assume that $I < L$ instances of the composite channel are recorded at K UEs. In the following, we explain each stage.

4.3.1 Stage-1: RIS Configuration

As alluded to earlier, this stage concerns configuring RIS elements to maximize Ω. Mathematically, the optimization problem of stage-1 is formulated as

$$\begin{aligned} \text{P1:} \quad & \max_{\mathbf{\Phi}'} \quad \Omega = \frac{1}{K}\mathsf{Trace}(\mathbf{\Omega}) \\ & \text{s.t.} \quad 0 \le \phi_m^{'} \le 2\pi, \qquad \forall m = 1, 2, \cdots, M \end{aligned} \tag{4.3}$$

where $0 \le \phi_m^{'} \le 2\pi$ is the unit modulus constraint of RIS phase shifts. $\mathbf{\Phi}^{'} = [e^{-j\phi_1^{'}}, e^{-j\phi_2^{'}}, \ldots, e^{-j\phi_M^{'}}]$, where $\mathbf{\Phi}^{'} \in \mathbf{\Phi}$ is the subset containing M RIS elements selected for optimization. We explain the selection of M elements in the following.

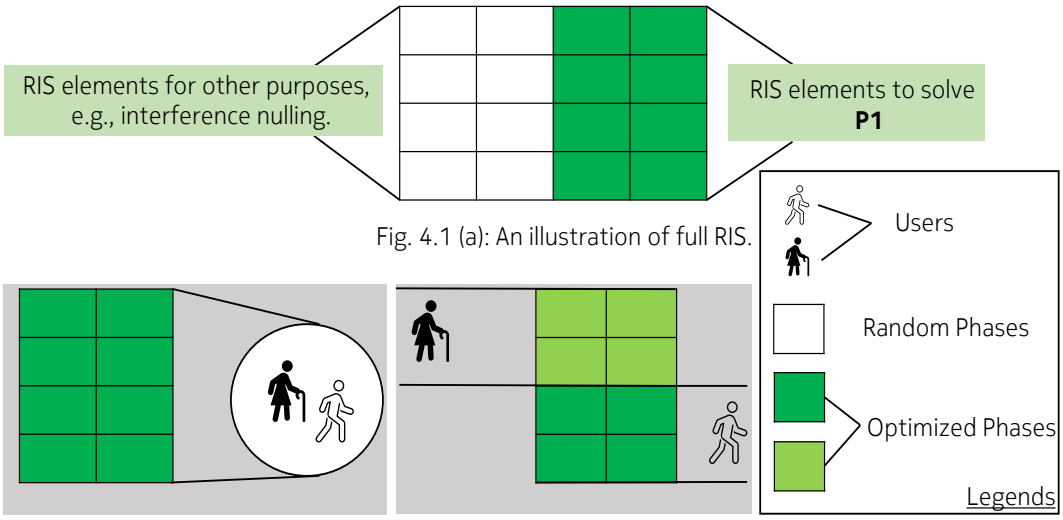


Fig. 4.1 (a): An illustration of full RIS.

Fig. 4.1 (b): Non-partitioned RIS (NP-RIS). Fig. 4.1 (c): Partitioned-RIS (P-RIS).

Figure 4.1: An illustration of RIS surface used in the study. (a) Showing a complete RIS surface having N RIS elements, with random phases $(N - M)$, denoted with white color, and optimized phases (M), denoted with green color. (b) Representing the optimized portion of RIS used for all UEs. (c) Depicting the division of the optimized portion of RIS used for each UE. In both partitions (NP-RIS and P-RIS), random phases are also present but dropped for illustration purposes.

To solve P1, first, we select M RIS elements using two different methodologies. In the first methodology, coined as *partitioned-RIS* (P-RIS), we allocate a block of quasi-RIS elements for each UE. Each block's size depends on the total number of UEs, K, and the number of chosen RIS elements for optimization, i.e., M. For instance, in the case of $M = 10$ and $K = 2$, each block will be composed of 5 RIS elements, meaning that the UE $k \in K$ will use 5 RIS elements to optimize P1. The optimization of P1 will consider the remaining 5 elements for k^{th} UE with random phases, which can be used for other applications, e.g., interference cancellation. In the second method, called *non-partitioned-RIS* (NP-RIS), the optimization problem sees a block of M RIS elements as one. In other words, the optimization of K UEs is done collectively. It is important to mention that, in both methods, NP-RIS and P-RIS, a part of the RIS surface is present to use for other purposes. Hence, the composite channel arrived at a UE is comprised of N RIS elements, as mentioned in Section 4.2. An illustration of RIS partitioning is shown in Fig. 4.1.

Besides, to optimize phases for P1, we use a meta-heuristic optimization algorithm [87], [88]. Nevertheless, exploration of various ML-based solution(s) can also be possible, which we leave as future direction. Briefly, we use a genetic algorithm (GA), where M random phases of RIS are initialized, $\mathbf{\Phi}^{'}$. Then an initial population of size M_{p}, where each row (called a chromosome) represents a possible set of M RIS phases, is created. The fitness function, Ω, is evaluated on each chromosome, where 5% elite chromosomes remain unchanged, and the rest are mutated for the next generation. The algorithm converges when the change in $\Omega \leq \lambda$, where λ is the tolerance value. Otherwise, the algorithm halts when a maximum number of iterations, T_{it}, are completed. The pseudo-code of the optimization process for stage-1 is summarized in lines 1-19 of Algorithm 2. Briefly, the optimized RIS phases maximize the objective function, i.e., Ω. Consequently, the optimized composite channel for K UEs with L instances of the channel can be expressed as $\widehat{\mathbf{H}}$. To generate training data for stage-2, various examples with L instances of the composite channel are created for the channel prediction, described in the following subsection.

4.3.2 Stage-2: Channel Prediction using ML

Channel prediction[1] is the process of forecasting CSI realizations by exploiting past observations. In this stage, we simultaneously predict the channel of K UEs by exploiting RNN (explained in Chapter 2 for $K = 1$). The optimization problem of stage-2 is given below.

$$\text{P2:} \quad \min \quad \sum_{k=1}^{K} \mathsf{E}\Big[\Big\| \widehat{\mathbf{h}}^k - \widetilde{\mathbf{h}}^k \Big\|^2 \Big] \tag{4.4}$$

s.t.

$$\widetilde{\mathbf{h}}^k = c_p(\widehat{h}^k_{i=1}, \cdots, \widehat{h}^k_I), \qquad \forall k = 1, 2, \cdots, K \tag{4.5}$$

[1] For details, please see Chapter 2.

where $\{\widehat{h}^k_{i=1}, \cdots, \widehat{h}^k_I\} \in \widehat{\mathbf{H}}$ is the optimized channel, having I past observations, for the k^{th} UE obtained in stage-1. $c_p(\cdot)$ represents the single channel predictor for all UEs. $\widehat{\mathbf{h}}^k \in \mathbb{C}^D$ and $\widetilde{\mathbf{h}}^k \in \mathbb{C}^D$ are the true and predicted channel, respectively, for the k^{th} UE, where $D = L - I$ is the length of predicted CSI realizations.

The operation of channel predictor is divided into two phases: training and prediction. In the first phase, after selecting the hyperparameters, e.g., learning rate and optimization algorithm, training data is fed into the channel predictor. Consequently, the channel predictor processes each sample (or a batch of samples depending on the batch size) and compares the predicted value with the true label. The objective of P2 is to forecast multi-step ahead values of CSI, which are as close as possible to true values. To do this, weights of RNN are recursively updated through backpropagation until a certain convergence condition is met, i.e., MSE is minimized. The training phase is summarized in lines 20-27 of Algorithm 2. The trained channel predictor can make multi-step ahead predictions. Hence, at a time instant l, the channel of K UEs for D future steps, denoted by $\widetilde{\mathbf{H}}_{l+D} \in \mathbb{C}^{D \times K}$, is predicted. In the following section, we will show that the channel predictor can learn the channel with minimum training cost, i.e., epochs, thanks to stage-1.

4.4 Simulation Results

This section evaluates the performance of the proposed mechanism. More specifically, we validate the performance under two methodologies: i) when RIS is randomly configured (random phases) and ii) when RIS is intelligently configured (optimized phases). Unless stated otherwise, the BS is serving $K = 10$ UEs via a RIS placed at a distance r of 50 m. The channel of BS-RIS is modelled using the 3D Saleh-Valenzuela channel model [89], where 2 NLoS paths are assumed. The channel gain is taken as $\sim \mathcal{N}_{\mathbb{C}}(0, 10^{-0.1\Lambda(r)})$, where $\Lambda(r) = 20 + 0.10\log_{10}(r) + \xi$, with $\xi \sim \mathcal{N}(0, \sigma^2_\xi)^3$. The transmit antenna gain is 9.60 dBi, $\sigma_\xi = 3.8$ dB. The channel between RIS-UE is tapped delay line (TDL) channel model in 3GPP 5G NR [90]. The channel coefficients used in our study are generated with Matlab 5G Toolbox function *nrTDLChannel*, where the implementations are exactly following the aspects of 3GPP 5G NR standard protocol TR 38.901 [90]. In *nrTDLChannel*, the UE velocity is set to 100 km/h, which indicates that the channel is highly time-varying; the rest of the parameters are set as default. The rest of the simulation parameters are: $L = 4, I = 3, N = 320, f_c = 28$ GHz, $\lambda = 0.01, \gamma = 10^{-3}, M = \frac{N}{2}, T_{it} = 100 \times M$, unless otherwise stated. For the training of channel predictor, 3-layer RNN architecture is used: 1 input, 1 output, and 1 hidden layer. The input and output neurons are chosen according to the values of I, D, K, whereas the hidden layer has 50 neurons. The dropout layer is adapted to prevent over-fitting, which drops hidden units randomly with a probability of 0.2. *Adam* optimizer is used to update weights of RNN, where the learning rate is set to 10^{-3}, activation function in the hidden layer is *hyperbolic tangent*, the number of epochs $\mathsf{E}_{\mathsf{pochs}} = 50$, and the batch size is set to 20. The dataset in

Algorithm 2: Pseudo-Code of the Proposed Mechanism

Input: $\mathbf{H}^{\mathsf{a}}, \mathbf{H}^{\mathsf{b}}, L,\ N,\ M,\ K, \gamma\ , \lambda, M_{\mathsf{p}}$
Output: $\widetilde{\mathbf{H}}_{l+D}$

Choose number of RIS elements (M) to solve P1 $\rightarrow\ M = \frac{N}{2}$;
Select partitioning method to divide quasi-RIS$\rightarrow$ P-RIS or NP-RIS;
if P-RIS **then**
  $M = \frac{M}{K}$
else
  M
end if
`Initialize Population:` $(0, 2\pi, M, M_p)$
Compute Ω for each chromosome: $\Omega = \frac{1}{K}\mathsf{Trace}(\mathbf{\Omega})$;
for $t_{it} = 2 : T_{it}$ **do**
  select 5% elite chromosomes from $t_{it} - 1$
  select remaining chromosomes from $t_{it} - 1$ for mutation
  Compute Ω for each chromosome
  if $change$ in $\Omega \leq \lambda$ **then**
    break for loop
  end if
end for
Select the chromosome having max. Ω: $\mathbf{\Phi}^{'}$
Compute composite channel using $\mathbf{\Phi}^{'} \rightarrow \widehat{\mathbf{H}}$
for $e_{\mathsf{tr}} = 1$ **to** $\mathsf{E}_{\mathsf{pochs}}$ **do**
  Input features and output labels of RNN;
  Calculate objective function using P2
  Update weights of RNN [2];
  if $change$ in P2 $\leq \gamma$ **then**
    break for loop
  end if
end for
Obtain $\widetilde{\mathbf{H}}_{l+D}$ using I past observations of $\widehat{\mathbf{H}}$ and trained channel predictor
return $\widetilde{\mathbf{H}}_{l+D}$

stage-2 is composed of $20\,$k examples and data distribution: $70\%, 10\%, 20\%$, for training, validation, and test dataset, respectively. Furthermore, all the simulation results are obtained with TensorFlow $2.3.0$, Python $3.6.13$, and Matlab R2022b. The performance of the proposed mechanism is evaluated using the prediction factor (Ω) of the composite channel over I different CSI observations.

Fig. 4.2 plots the comparison between two partitioning methods of RIS, i.e., P-RIS and NP-RIS. Further, the performance is compared with randomly and intelligently configured RIS elements. We observe that the intelligent configuration of RIS shows better

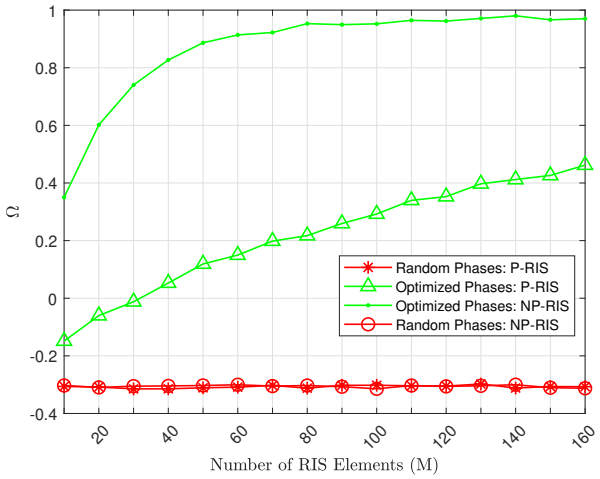


Figure 4.2: Comparison between RIS partitioning methods.

performance in both partitioning methods. However, NP-RIS outperforms P-RIS, which is due to the fact that RIS elements are collectively optimized for all UEs. Also, increasing the number of optimized RIS elements (M) clearly improves the performance. For example, NP-RIS requires nearly $M = 140$ to maximize Ω, whereas P-RIS achieves $\Omega \approx 0.41$ for same value of M. In contrast, randomly configured RIS does not improve Ω, and it remains unchanged. In the rest of this section, we evaluate performance using NP-RIS, and we drop the curves of random phases for the illustration purposes.

To observe the performance under different numbers of UEs, we have plotted the results in Fig. 4.3[2] against the number of optimized RIS elements. The increasing trend in the prediction factor can be observed with increasing M. Importantly, it can be observed that fewer UEs, i.e., $K = 2$, require fewer RIS elements for optimization. For instance, in the case of $M = 10$, $K = 2$ improves the prediction factor by approximately 31% in comparison to $K = 4$. Nevertheless, higher values of M depict no gain. We can conclude that a higher number of UEs require more intelligently configured RIS elements.

In Fig. 4.4, we show the performance of observation intervals, I, by using different values of M. It can be observed from Fig. 4.4 that the prediction factor increases with the increase of M. Nevertheless, $I = 3$ requires only $M = 25$ RIS elements to maximize Ω, the rationale behind this is channel predictability is only maximized among 3 consecutive time instances of the channel. In contrast, to maximize channel predictability among 10 consecutive instances of the channel, approximately $M = 70$ intelligently configured RIS elements are required. In summary, we have learned that a higher number of M is required to maximize channel predictability among multiple instances of the channel.

During the training of channel predictor, we observed that intelligently configured

[2]For better illustration, we have dropped the curves of randomly configured RIS. As it is evident from Fig. 4.2, randomly configured RIS does not improve performance.

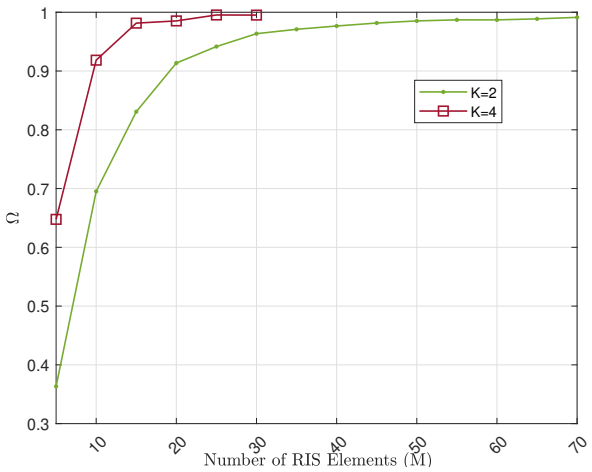


Figure 4.3: A comparison with different numbers of UEs. $N = 140$.

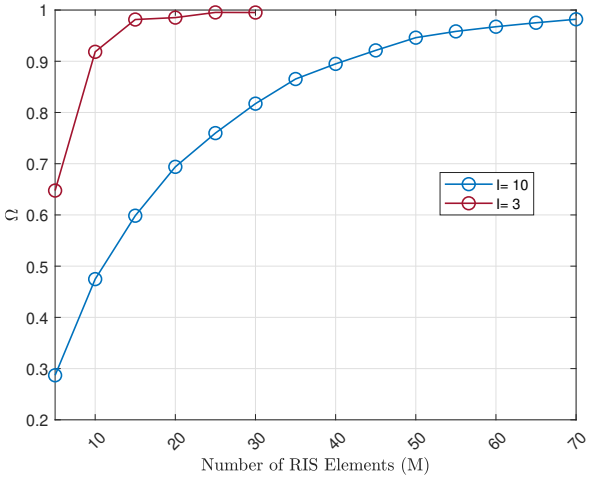


Figure 4.4: Performance of Ω with different L and M. $K = 2$, $N = 140$.

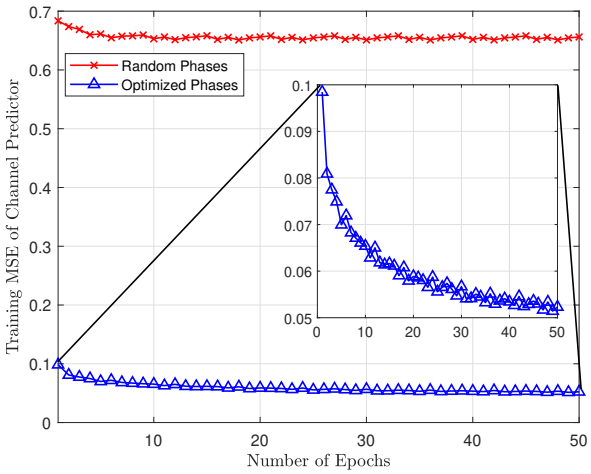


Figure 4.5: Performance evaluation during the training of channel predictor. $K = 2, N = 140, \Omega = 0.99$.

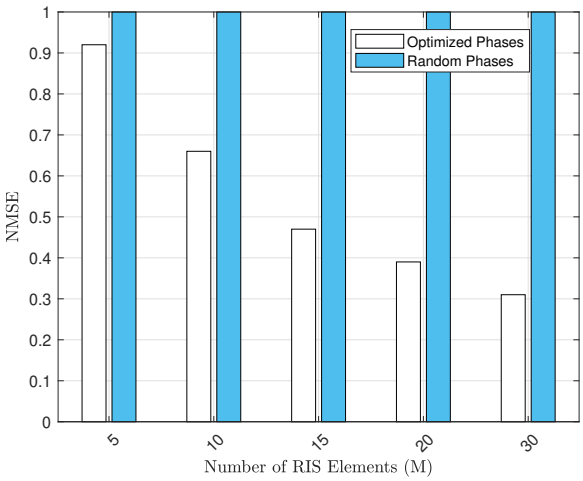


Figure 4.6: Performance of channel prediction with the number of RIS elements. $K = 2, N = 2M$.

RIS can quickly learn future channel realizations. To demonstrate this, in Fig. 4.5, we have drawn MSE (between the true and predicted channel realizations) obtained during the training of channel predictor against the number of epochs. The observations show that at the end of the first epoch, intelligently configured RIS brings approximately 85% reduction in MSE in comparison to randomly configured RIS. Further, by the end of 50 epochs, 92% reduction in the MSE is observed. Fig. 4.6 reveals the performance of the channel predictor on the test dataset with various numbers of RIS elements. The results are plotted for randomly as well as intelligently configured RIS elements. The performance is evaluated using:

$$\text{NMSE} = \text{E}\left\{ \frac{\left\| \widehat{\mathbf{H}} - \widetilde{\mathbf{H}} \right\|_{\text{FRO}}^2}{\left\| \widehat{\mathbf{H}} \right\|_{\text{FRO}}^2} \right\}, \tag{4.6}$$

which is calculated between the true and the predicted channel. The results of Fig. 4.6 show that the intelligently configured RIS can significantly predict future channel realizations, whereas randomly configured RIS is unable to predict the channel. Another important observation is the accuracy of the predicted channel improves with the increase in the number of intelligently configured RIS elements because higher M brings a higher prediction factor, as explained in previous results. In contrast, randomly configured RIS shows NMSE $= 1$ irrespective of the number of RIS elements. In summary, we have concluded from the presented results that the proposed mechanism can improve the performance of channel predictor by using fewer epochs and exploiting simple neural network architecture.

4.5 Conclusion

A highly time-varying multiuser channel will always be challenging to predict, even using a large-RIS-assisted communication system with random phase configuration. We have provided an efficient mechanism to make the channel predictable by carefully configuring a small portion of RIS, where a considerable part is dedicated to applications other than channel prediction. The simulation results corroborated the validity of the proposed mechanism. Thus we encourage using such an approach where RIS is configured by considering other applications. The outcome showed that a simple channel prediction model can learn the channel evolution of multiple UEs when RIS is intelligently configured.

Chapter 5

Conclusions and Outlooks

5.1 Conclusions

We have learned from the thesis that ML plays a pivotal role in future cellular networks, and there is high potential to propose ML algorithms from standardization perspective. In particular, we have shown that the performance of CSI feedback enhances by using ML techniques. The novel idea of using channel prediction not only resulted in CSI feedback overhead reduction but also improved the quality of CSI acquired at the BS. In Chapter 1, motivated by the promise of the use of ML algorithms, we presented an overview of the practical challenges of deploying ML techniques in wireless networks and how to deal with them. We believe that overcoming these challenges, both in research as well as at standardization levels, will pave the way for next-generation wireless communication to be effective and sustainable.

Chapter 2 borrowed the idea of channel prediction and introduced its use towards enhancing CSI feedback in a single-user massive MIMO system. To this end, a novel algorithm, coined as CSIFB-PNet, is developed. We have learned that CSIFB-PNet can be used on both sides as well as one-side of the communication system. Generally speaking, higher gain in the acquired CSI at BS is achieved by using a minimum amount of overhead bits, that is, 2. In contrast, the gain is negligible when ≥ 5 overhead bits are considered. The rationale behind nearly zero-gain for ≥ 5 is minimum information loss while compressing the CSI. Thus, CSIFB-PNet is the best choice when there is a high compression error. We have also concluded that exploiting ML at the UE-only brings a gain of approximately 3.25 dB in comparison to when using at both ends. The improvement is due to exploiting uncompressed CSI at the UE. The chapter also evaluated the capability of ML-based channel predictors to correctly predict CSI realizations using real-world measurements. The analyzed ML techniques proved that prediction is accurate when facing real-world data, and it can work under different settings, such as quantization levels and representations. In addition, the channel predictors have proved to be disentangled from the specific channel track, creating a prediction abstraction layer that can be exploited to predict also physically diverse channels (e.g., a channel predictor

trained in a LoS area and exploited in a NLoS area). Such techniques can hence pave the way to advanced channel acquisition methods that will reduce dependency on channel sounding and estimation.

Chapter 3 addressed a novel method, coined CSILaBS, to efficiently recover compressed CSI without implementing ML at the UE. When ML was applied at the BS-only, the generated function raised the idea of developing a codebook for downlink overhead reduction, which has shown highly standard-relevant implementation. The results showed the effectiveness of CSILaBS compared to benchmark schemes. The results demonstrated an increase of approximately 23% in precoding gain than without ML when 2 bits are used to feedback information. Thus, CSILaBS is the best choice when there is a high compression error and UE is incapable of ML training and storage. By extending the work towards multiuser, we proposed different techniques to evaluate feedback from UE while aiming to improve acquired CSI accuracy. Particularly, we observed that CSILaBS with the proposed probabilistic feedback method brings a gain of approximately 32% when 15 users are considered and there are 10 resource blocks. Furthermore, the gain drops to almost half when the number of users is doubled. In a nutshell, we have learned that CSILaBS can retain ML gains without implementing ML on both sides or reporting massive ML parameters while doing one-sided ML training.

Chapter 4 focused on exploiting RIS to enhance the performance of channel predictor. By considering a highly time-varying channel of multiuser communication environment, we proposed a scalable idea of utilizing quasi-RIS to make the channel predictable. The outcomes of the work showed that RIS plays a predominant role in not only improving the performance of channel prediction but also requiring fewer epochs to train the ML model. We also showed that intelligently configured RIS can predict multi-step future channel realizations while saving sufficient RIS elements for other applications. We have learned that the number of required RIS elements to maximize the prediction factor is dependent on the number of users within the network. For instance, in the experiments, we observed that approximately 25 RIS elements are required to acquire perfect channel prediction when 2 users are considered. The required RIS elements grow to 45 and 110 when there are 4 and 10 users, respectively. Nevertheless, performance improvement is observed by saving sufficient RIS elements for other applications.

5.2 Future Research Directions

Although the thesis work showed promising results, however, there are still many possible future directions that one can take as a continuation of this thesis. First, the comparison of proposed works with standardized codebook-based techniques, that is, type-I and type-II, can be a promising future direction. Second, the performance evaluation of CSI feedback schemes can be tested on a variety of datasets. This is due to the reason that in a practical environment, the users do not stay in a cell, and hence underlying channel distribution may change. Thus, there is need for a standard dataset which can be used as a benchmark to evaluate proposed work. Correspondingly, ML

algorithms can be trained and designed to work in changing propagation conditions. Also, system-level simulations can be carried out to observe the outcomes of deploying ML. There is need to compare the proposed ML-based channel predictors against statistical methods, for instance, Kalman filter [91], [92]. Besides, in wireless communication, we have to deal with complex numbers; however, the proposed models are evaluated on real-valued ML models. Therefore, a comparison of complex-valued ML models against real-valued models can be a possible future direction [93], [94].

The design of a single-ML model for multiuser is the next big challenge, which can be addressed in the continuation of this thesis work. This will help the network to manage the ML training and probably reduce the computation cost. Thus, there is room to propose adaptive ML algorithms to tackle the challenge of CSI feedback overhead. Second, the existing standards for CSI feedback cannot be completely changed with the deployment of ML. Hence, there is gap for revising existing standards from the perspective of ML models. Besides, the selection of feedback in a multiuser scenario can be revised and improved, for example, by proposing a hybrid framework composed of error-bin and probabilistic feedback. On the other hand, the synchronization of twin channel predictors is an important issue which has yet to be addressed. Though our work considered using the same dataset at each time step to make a prediction, there is a need to propose mechanisms to synchronize the ML models at BS and UE. For instance, how do both models use the same training parameters, weights, and datasets? There is a need to compare the results regarding the overhead required to train and test the models against the CSI acquisition accuracy. Besides, in the proposed work, we considered uniform quantization to compress CSI; however, a comparison of different quantization schemes can also be done.

Though our work gave the introduction of RIS deployment to enhance CSI feedback performance through channel prediction. Nevertheless, a detailed comparison with CSI feedback schemes is important to take an insight look on the challenges that we can face. For example, the CSI dimension will massively increase with the number of RIS elements. Hence, ML-based frameworks must be revised to tackle such huge dimensions. For instance, a RIS with 100 elements per dimension will require ML to tackle $10\,\mathrm{k}$ CSI dimensions for a single-antenna UE. This number is huge in comparison to a massive MIMO system. Therein, ML techniques can be revised to reduce the computation cost. In addition to that, the design of an ML model can be one of the future directions to fine-tune phases of RIS, which are handled through evolutionary algorithm in this thesis. There is also requirement of proposing closed-form solutions (if possible) to make the channel predictable. Besides, quasi-RIS is exploited to achieve the desired objective in the thesis, one can analyze the performance by solving two different problems. For example, interference mitigation and channel prediction in multiuser can be jointly solved by exploiting entire RIS, where a portion of RIS is dedicated for channel prediction and rest for interference mitigation. The proposed solution addressed in Chapter 4 can also be applied to achieve channel hardening in RIS, similar to what is done in [95].

Besides, the reporting and verification of training data is a crucial issue in the pro-

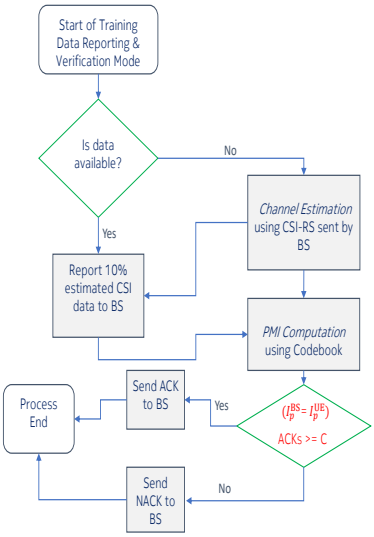
posed CSI feedback frameworks. For example, in the case of CSILaBS, addressed in Chapter 3, training data must be acquired at the BS. However, there is a huge cost to report this training data. In the following, we propose a standard-relevant solution to acquire training data in CSILaBS and CSIFB-PNet.

5.2.1 How to Acquire Training Data for CSILaBS and CSIFB-PNet?

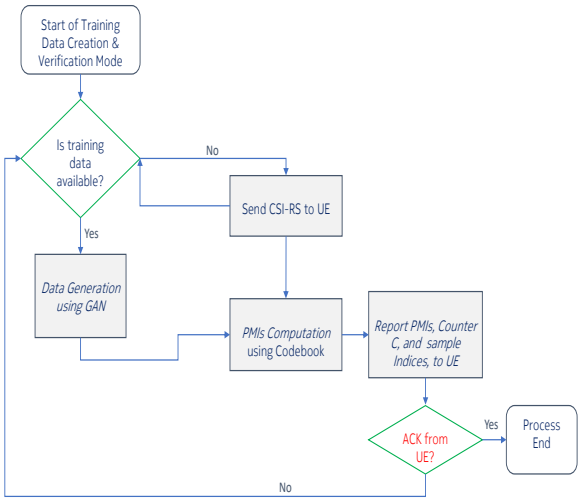
As we have learned from Chapter 1 and proposed CSI feedback schemes, that is, CSILaBS and CSIFB-PNet, the availability of ML algorithms' training data is a crucial problem in a wireless network. For instance, CSILaBS, deployed at BS for CSI feedback, requires training data to be reported by UE. This reporting of training data can cause a huge OTA overhead. In other words, the problem to solve is to have identical training data for ML-based CSI prediction at both ends (UE and BS) at minimum OTA overhead without compromising much on the quality of CSI prediction. Considering such issues, we address the use of data augmentation either at BS or at both ends, where fewer CSI entries can help to generate huge training data by exploiting ML-based algorithm(s) [96]. Motivated by this idea, we present two embodiments to verify the data generated by data augmentation algorithms, e.g., GAN [96]. The embodiments can significantly reduce the OTA overhead of training data compared to state-of-the-art solutions, e.g., reporting a bunch of compressed estimated CSI to BS. The proposed idea, called CSI data for training ML (CSI-DTML), has the following main features:

- CSI-DTML focuses on the method of verification of data generated by GAN. To this end, CSI-DTML proposes two embodiments for verifying generated data: BS-Driven and UE-Driven.
- In embodiment-1 (BS-Driven), implementation of GAN is considered only at BS, and 3GPP standardized codebooks help verify newly generated data.
- In embodiment-2 (UE-Driven), implementation of GAN is considered at both ends, i.e., BS and UE. Also, twin GANs are considered. GAN at UE assists the BS in collecting accurate data, whereas data verification is done locally at UE.

Both embodiments can help to verify the generated data effectively, limiting OTA overhead, and have a substantial standard impact as it requires signalling and network support. Before explaining CSI-DTML, here we first summarize data augmentation. Data augmentation is "a process to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data" [96]. The well-known data augmentation techniques are random cropping, shearing, rotation, flipping, adding Gaussian noise, color shifting ($+20$ to -20 etc.), and GAN. GAN is a popular ML algorithm that can generate new data by using existing data and random noise (e.g., flipped CSI). GAN comprises two stages, i.e., *generator* and *discriminator*. The *generator* generates new data, and *discriminator* verifies the newly generated data by comparing it with the existing data. In the following, we address the two embodiments.



(a) Implementation at UE.



(b) Implementation at BS.

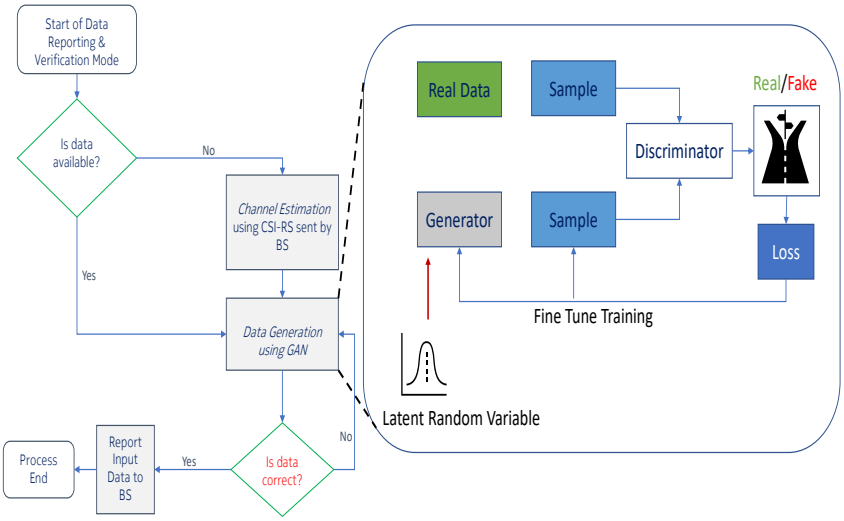
Figure 5.1: Implementation flowchart of BS-Driven CSI-DTML.

5.2.1.1 BS-Driven CSI-DTML

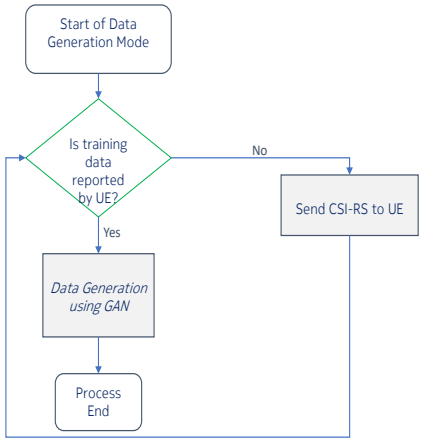
The main steps of embodiment-1, BS-Driven CSI-DTML, are summarized below.

- BS informs the use of data augmentation algorithm at its end. Therefore, it requests UE to send some initial samples of the CSI (compressed or uncompressed; it depends on how much overhead a UE can afford).
- The UE feedback ACK/NACK for the request sent previously. An ACK indicates that UE has the initial samples to share. In the case of NACK (indicating the unavailability of initial samples), BS sends CSI-RS so that UE can generate some initial samples using a CSI estimation algorithm.
- Once the data is available at UE, UE does not feedback the entire data as it causes OTA overhead. Instead, it divides data into B batches such that the data properties in each batch reflect the original distribution of data. Hence, UE feedback one batch (size of each batch can be, e.g., 10% of entire data) out of B batches. The reason to send the data in batches is to reduce overhead. Also, if GAN can generate the rest of the batches by only using one batch, then it's better to report only one batch and save OTA training overhead by 90%. By exploiting the reported batch, GAN generates the rest of the batches at the BS.
- At this stage, BS computes a few PMIs of the generated batches by exploiting 3GPP standardized codebook [18]. Then the BS reports PMIs, corresponding true indices of the generated channel samples, and a reliability counter C (to count the number of ACKs).
- By exploiting the information sent in step-4, UE computes the PMIs for the data available at its end. Next, PMIs sent by the BS, denoted by I_p^{BS}, and calculated by the UE, I_p^{UE}, are compared, and the number of ACKs is added to the counter C. If the counter C reaches the value sent by the BS, UE transmits an ACK signal or NACK (in case C does not reach the threshold set by BS).
- At this stage, UE sends an ACK to BS, which indicates that the generated data at the BS is correct, and it can be used for training of an AI/ML algorithm, e.g., for channel prediction in the case of CSILaBS. Alternatively, in the case of NACK (indicating that data generated by BS is incorrect), UE sends another batch from the data, which can help GAN generate better data.
- Steps 4-6 are repeated (in case of NACK) until the BS receives an ACK. If ACK is not received after reporting all batches, this shows that GAN did not bring any advantage. Hence, we fall back to the conventional approach, i.e., the entire data is reported to the BS by UE.

The implementation flowchart of BS-Driven CSI-DTML is given in Fig. 5.1.



(a) Implementation at UE.



(b) Implementation at BS.

Figure 5.2: Implementation flowchart of UE-Driven CSI-DTML.

5.2.1.2 UE-Driven CSI-DTML

The main steps of embodiment-2, UE-Driven CSI-DTML, are summarized below.

- BS and UE agree to use twin GAN, where twin means that both algorithms have the same initialization parameters, e.g., weights.
- The UE feedback ACK/NACK to initiate the process at its end. An ACK indicates that UE has the initial samples to share.
- In the case of NACK (indicating the unavailability of initial samples), BS sends CSI-RS so that UE can generate some initial samples using a channel estimation algorithm.
- Once the data is available at UE, UE tries various combinations of batches to generate new data by exploiting GAN. The newly generated data can be verified locally at UE through different parameters, e.g., MSE and cosine similarity.
- Once the right combination of batches is computed in step- 3, UE reports the correct number of batches to BS and informs to generate the remaining batches. This is helpful for the BS as it does not need to ask for more data to train GAN. Alternatively, BS does not need to send the signals to UE to verify data as the right combination of training data has already been reported by the UE.
- BS generates the new data. Once the data is generated, it is unnecessary to verify it as it has already been verified at UE. The newly generated data can be used to train channel predictor for CSILaBS and CSIFB-PNet.

The implementation flowchart of UE-Driven CSI-DTML is given in Fig. 5.2. The implementation process of GAN has also been provided, which will also be followed in BS-Driven CSI-DTML when BS has to use GAN. Nonetheless, the performance gains of data augmentation are still need to be tested and investigated. In the future, such algorithms can be evaluated by exploiting the dataset used in our study.

References

Related Own References

International Journals with peer review

[1] M. K. Shehzad, L. Rose, M. M. Butt, I. Z. Kovács, M. Assaad, and M. Guizani, "Artificial intelligence for 6G networks: Technology advancement and standardization," *IEEE Vehicular Technology Magazine*, vol. 17, no. 3, 2022.

[2] M. K. Shehzad, L. Rose, S. Wesemann, and M. Assaad, "ML-based massive MIMO channel prediction: Does it work on real-world data?" *IEEE Wireless Communications Letters*, vol. 11, no. 4, 2022.

[5] M. K. Shehzad, M. Assaad, and L. Rose, "Reconfigurable intelligent surfaces: How to enhance multiuser channel prediction?" *IEEE Wireless Communications Letters*, 2023.

International conferences with peer review

[6] M. K. Shehzad, L. Rose, and M. Assaad, "Dealing with CSI compression to reduce losses and overhead: An artificial intelligence approach," in *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2021.

[7] M. K. Shehzad, L. Rose, and M. Assaad, "A novel algorithm to report CSI in MIMO-based wireless networks," in *ICC 2021 - IEEE International Conference on Communications*, 2021.

[8] M. K. Shehzad, L. Rose, and M. Assaad, "RNN-based twin channel predictors for CSI acquisition in UAV-assisted 5G+ networks," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021.

[9] M. K. Shehzad, L. Rose, M. F. Azam, and M. Assaad, "Real-time massive MIMO channel prediction: A combination of deep learning and NeuralProphet," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, 2022.

Submitted Journals

[3] M. K. Shehzad, L. Rose, S. Wesemann, M. Assaad, and S. A. Hassan, "Design of an efficient CSI feedback mechanism in massive MIMO systems: A machine learning approach using empirical data," *in IEEE Transactions on Cognitive Communications and Networking (TCCN)*, 2023.

[4] M. K. Shehzad, L. Rose, and M. Assaad, "Massive MIMO CSI feedback using channel prediction: How to avoid machine learning at UE?" *in IEEE Transactions on Wireless Communications (TWC)*, 2023.

Other References

[10] 3GPP, "Study of enablers for network automation for 5G (Release 16)," Technical Report (TR) 23.791, Jun. 2019.

[11] I. Union, "IMT traffic estimates for the years 2020 to 2030," *Report ITU*, 2015.

[12] J. Hoydis, F. A. Aoudia, A. Valcarce, and H. Viswanathan, "Toward a 6G AI-native air interface," *IEEE Communications Magazine*, vol. 59, no. 5, 2021.

[13] A.-A. A. Boulogeorgos, E. Yaqub, M. Di Renzo, A. Alexiou, R. Desai, and R. Klinkenberg, "Machine learning: A catalyst for THz wireless networks," *Frontiers in Communications and Networks*, 2021.

[14] 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, 2017.

[15] *5G/NR - CSI RS Codebook*, https://www.sharetechnote.com/html/5G/5G_CSI_RS_Codebook.html, Accessed on 2020-09-22. [online].

[16] https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3216, Accessed on 2020-09-22. [online].

[17] H. Holma, A. Toskala, and T. Nakamura, *5G technology: 3GPP new radio*. John Wiley & Sons, 2020.

[18] Z. Qin and H. Yin, "A review of codebooks for CSI feedback in 5G new radio and beyond," *arXiv preprint arXiv:2302.09222*, 2023.

[19] D. J. Love, R. W. Heath, V. K. Lau, D. Gesbert, B. D. Rao, and M. Andrews, "An overview of limited feedback in wireless communication systems," *IEEE Journal on selected areas in Communications*, vol. 26, no. 8, 2008.

[20] D. J. Love and R. W. Heath, "Limited feedback diversity techniques for correlated channels," *IEEE Transactions on Vehicular Technology*, vol. 55, no. 2, 2006.

[21] N. Ravindran, N. Jindal, and H. C. Huang, "Beamforming with finite rate feedback for LOS MIMO downlink channels," in *IEEE GLOBECOM 2007-IEEE Global Telecommunications Conference*, IEEE, 2007.

[22] B. Lee, J. Choi, J.-Y. Seol, D. J. Love, and B. Shim, "Antenna grouping based feedback compression for FDD-based massive MIMO systems," *IEEE Transactions on Communications*, vol. 63, no. 9, 2015.

[23] M. S. Sim, J. Park, C.-B. Chae, and R. W. Heath, "Compressed channel feedback for correlated massive MIMO systems," *Journal of Communications and Networks*, vol. 18, no. 1, 2016.

[24] W. Shen, L. Dai, B. Shim, Z. Wang, and R. W. Heath, "Channel feedback based on AoD-adaptive subspace codebook in FDD massive MIMO systems," *IEEE Transactions on Communications*, vol. 66, no. 11, 2018.

[25] R. Ahmed, K. Jayasinghe, and T. Wild, "Comparison of explicit CSI feedback schemes for 5G new radio," in *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, 2019.

[26] Z. Qin, J. Fan, Y. Liu, Y. Gao, and G. Y. Li, "Sparse representation for wireless communications: A compressive sensing approach," *IEEE Signal Processing Magazine*, vol. 35, no. 3, 2018.

[27] D. L. Donoho, A. Maleki, and A. Montanari, "Message-passing algorithms for compressed sensing," *Proceedings of the National Academy of Sciences*, vol. 106, no. 45, 2009, issn: 0027-8424. eprint: https://www.pnas.org/content/106/45/18914.full.pdf.

[28] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, vol. 57, no. 11, 2004.

[29] P.-H. Kuo, H. Kung, and P.-A. Ting, "Compressive sensing based channel feedback protocols for spatially-correlated massive antenna arrays," in *2012 IEEE Wireless Communications and Networking Conference (WCNC)*, IEEE, 2012.

[30] X. Rao and V. K. Lau, "Distributed compressive CSIT estimation and feedback for FDD multi-user massive MIMO systems," *IEEE Transactions on Signal Processing*, vol. 62, no. 12, 2014.

[31] P. Cheng and Z. Chen, "Multidimensional compressive sensing based analog CSI feedback for massive MIMO-OFDM systems," in *2014 IEEE 80th Vehicular Technology Conference (VTC2014-Fall)*, 2014.

[32] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, 1966.

[33] A. Beck and M. Teboulle, “A fast iterative shrinkage-thresholding algorithm for linear inverse problems,” *SIAM journal on imaging sciences*, vol. 2, no. 1, 2009.

[34] C. A. Metzler, A. Maleki, and R. G. Baraniuk, “From denoising to compressed sensing,” *IEEE Transactions on Information Theory*, vol. 62, no. 9, 2016.

[35] C.-K. Wen, W.-T. Shih, and S. Jin, “Deep learning for massive MIMO CSI feedback,” *IEEE Wireless Communications Letters*, vol. 7, no. 5, 2018.

[36] J. Guo, C.-K. Wen, S. Jin, and G. Y. Li, “Overview of deep learning-based CSI feedback in massive MIMO systems,” *IEEE Transactions on Communications*, vol. 70, no. 12, 2022.

[37] C. Lu, W. Xu, H. Shen, J. Zhu, and K. Wang, “MIMO channel information feedback using deep recurrent network,” *IEEE Communications Letters*, vol. 23, no. 1, 2018.

[38] Q. Yang, M. B. Mashhadi, and D. Gündüz, “Deep convolutional compression for massive MIMO CSI feedback,” in *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, 2019.

[39] Y. Sun, W. Xu, L. Liang, N. Wang, G. Y. Li, and X. You, “A lightweight deep network for efficient CSI feedback in massive MIMO systems,” *IEEE Wireless Communications Letters*, 2021.

[40] Y. Sun, W. Xu, L. Fan, G. Y. Li, and G. K. Karagiannidis, “AnciNet: an efficient deep learning approach for feedback compression of estimated CSI in massive MIMO systems,” *IEEE Wireless Communications Letters*, vol. 9, no. 12, 2020.

[41] Z. Liu, M. del Rosario, and Z. Ding, “A Markovian model-driven deep learning framework for massive MIMO CSI feedback,” *IEEE Transactions on Wireless Communications*, vol. 21, no. 2, 2022.

[42] H. Shan, X. Chen, H. Yin, L. Chen, and G. Wei, “Joint sparse autoencoder based massive MIMO CSI feedback,” *IEEE Communications Letters*, 2023.

[43] T. Wang, C.-K. Wen, S. Jin, and G. Y. Li, “Deep learning-based CSI feedback approach for time-varying massive MIMO channels,” *IEEE Wireless Communications Letters*, vol. 8, no. 2, 2019.

[44] X. Li and H. Wu, “Spatio-temporal representation with deep neural recurrent network in MIMO CSI feedback,” *IEEE Wireless Communications Letters*, vol. 9, no. 5, 2020.

[45] B. Zhang, H. Li, X. Liang, X. Gu, and L. Zhang, “Multi-task training approach for CSI feedback in massive MIMO systems,” *IEEE Communications Letters*, 2022.

[46] J. Guo, C.-K. Wen, and S. Jin, "Deep data hiding-based CSI feedback overhead elimination: An initial investigation," in *ICC 2022-IEEE International Conference on Communications*, IEEE, 2022.

[47] J. Shen, X. Liang, X. Gu, and L. Zhang, "Clustering algorithm-based quantization method for massive MIMO CSI feedback," *IEEE Wireless Communications Letters*, vol. 11, no. 10, 2022.

[48] S. Mourya, S. Amuru, and K. K. Kuchi, "A spatially separable attention mechanism for massive MIMO CSI feedback," *IEEE Wireless Communications Letters*, 2022.

[49] J. Cui, Q. Hu, F. Tan, W. Guo, and S. Wang, "IALNet: An integration attention lightweight neural network for massive MIMO CSI feedback," *IEEE Wireless Communications Letters*, 2023.

[50] Y. Ma, W. Yu, X. Yu, J. Zhang, S. Song, and K. B. Letaief, "Lightweight and adaptive FDD massive MIMO CSI feedback with deep equilibrium learning," *arXiv preprint arXiv:2211.15079*, 2022.

[51] Z. Lu, X. Zhang, R. Zeng, and J. Wang, "Better lightweight network for free: Codeword mimic learning for massive MIMO CSI feedback," *IEEE Communications Letters*, vol. 27, no. 5, 2023.

[52] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[53] B. Tolba, M. Elsabrouty, M. G. Abdu-Aguye, H. Gacanin, and H. M. Kasem, "Massive MIMO CSI deedback based on generative adversarial network," *IEEE Communications Letters*, vol. 24, no. 12, 2020.

[54] A. Aggarwal, M. Mittal, and G. Battineni, "Generative adversarial network: An overview of theory and applications," *International Journal of Information Management Data Insights*, vol. 1, no. 1, 2021.

[55] D. Bank, N. Koenigstein, and R. Giryes, *Autoencoders*, 2020.

[56] J. Guo, C.-K. Wen, S. Jin, and X. Li, *AI for CSI feedback enhancement in 5G-advanced*, 2022. arXiv: 2206.15132 [cs.IT].

[57] E. G. Larsson, O. Edfors, F. Tufvesson, and T. L. Marzetta, "Massive MIMO for next generation wireless systems," *IEEE communications magazine*, vol. 52, no. 2, 2014.

[58] T. L. Marzetta, "Noncooperative cellular wireless with unlimited numbers of base station antennas," *IEEE transactions on wireless communications*, vol. 9, no. 11, 2010.

[59] J. Zhang, C.-K. Wen, S. Jin, X. Gao, and K.-K. Wong, "On capacity of large-scale MIMO multiple access channels with distributed sets of correlated antennas," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 2, 2013.

[60] J. Kazemitabar and H. Jafarkhani, "Multiuser interference cancellation and detection for users with more than two transmit antennas," *IEEE Transactions on Communications*, vol. 56, no. 4, 2008.

[61] E. Björnson, J. Hoydis, and L. Sanguinetti, "Massive MIMO has unlimited capacity," *IEEE Transactions on Wireless Communications*, vol. 17, no. 1, 2017.

[62] C. Min, N. Chang, J. Cha, and J. Kang, "MIMO-OFDM downlink channel prediction for IEEE802.16e systems using Kalman filter," in *2007 IEEE Wireless Communications and Networking Conference*, 2007.

[63] R. O. Adeogun, P. D. Teal, and P. A. Dmochowski, "Parametric channel prediction for narrowband MIMO systems using polarized antenna arrays," in *2014 IEEE 79th Vehicular Technology Conference (VTC Spring)*, 2014.

[64] K. Baddour and N. Beaulieu, "Autoregressive modeling for fading channel simulation," *IEEE Transactions on Wireless Communications*, vol. 4, no. 4, 2005.

[65] J. T. Connor, R. D. Martin, and L. E. Atlas, "Recurrent neural networks and robust time series prediction," *IEEE transactions on neural networks*, vol. 5, no. 2, 1994.

[66] M. I. Jordan, "Serial order: A parallel distributed processing approach," in *Advances in psychology*, vol. 121, Elsevier, 1997.

[67] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, 1990.

[68] W. Liu, L.-L. Yang, and L. Hanzo, "Recurrent neural network based narrowband channel prediction," in *2006 IEEE 63rd Vehicular Technology Conference*, vol. 5, 2006.

[69] A. Hirose, *Complex-valued neural networks*. Springer Science & Business Media, 2012, vol. 400.

[70] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, 1997.

[71] O. Triebe, H. Hewamalage, P. Pilyugina, N. Laptev, C. Bergmeir, and R. Rajagopal, "NeuralProphet: Explainable forecasting at scale," *arXiv:2111.15397*, 2021.

[72] P. Liashchynskyi and P. Liashchynskyi, *Grid search, random search, genetic algorithm: A big comparison for NAS*, 2019.

[73] O. Triebe, N. Laptev, and R. Rajagopal, "*Ar-net*: a simple auto-regressive neural network for time-series," *arXiv preprint arXiv:1911.12436*, 2019.

[74] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2014.

[75] *CSI feedback with autoencoders*, https://fr.mathworks.com/help/comm/ug/csi-compression-autoencoder.html, Accessed on 2023-03-29. [online].

[76] X. Liang, Z. Liu, H. Chang, and L. Zhang, "Wireless channel data augmentation for artificial intelligence of things in industrial environment using generative adversarial networks," in *2020 IEEE 18th International Conference on Industrial Informatics (INDIN)*, vol. 1, 2020.

[77] J. Ghaderi and R. Srikant, "On the design of efficient CSMA algorithms for wireless networks," in *49th IEEE Conference on Decision and Control (CDC)*, IEEE, 2010.

[78] J. Ni, B. Tan, and R. Srikant, "Q-CSMA: Queue-length-based CSMA/CA algorithms for achieving maximum throughput and low delay in wireless networks," *IEEE/ACM Transactions on Networking*, vol. 20, no. 3, 2011.

[79] C. Pan, H. Ren, K. Wang, J. F. Kolb, M. Elkashlan, M. Chen, M. Di Renzo, Y. Hao, J. Wang, A. L. Swindlehurst, X. You, and L. Hanzo, "Reconfigurable intelligent surfaces for 6G systems: Principles, applications, and research directions," *IEEE Communications Magazine*, vol. 59, no. 6, 2021.

[80] O. Tsilipakos, A. C. Tasolamprou, A. Pitilakis, F. Liu, X. Wang, M. S. Mirmoosa, D. C. Tzarouchis, S. Abadal, H. Taghvaee, C. Liaskos, *et al.*, "Toward intelligent metasurfaces: The progress from globally tunable metasurfaces to software-defined metasurfaces with an embedded network of controllers," *Advanced optical materials*, vol. 8, no. 17, 2020.

[81] A. d. J. Torres, L. Sanguinetti, and E. Björnson, "Intelligent reconfigurable surfaces vs. decode-and-forward: What is the impact of electromagnetic interference?" In *2022 IEEE 23rd International Workshop on Signal Processing Advances in Wireless Communication (SPAWC)*, 2022.

[82] E. Björnson, H. Wymeersch, B. Matthiesen, P. Popovski, L. Sanguinetti, and E. de Carvalho, "Reconfigurable intelligent surfaces: A signal processing perspective with wireless applications," *IEEE Signal Processing Magazine*, vol. 39, no. 2, 2022.

[83] T. Jiang and W. Yu, "Interference nulling using reconfigurable intelligent surface," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 5, 2022.

[84] Z. Huang, B. Zheng, and R. Zhang, "Transforming fading channel from fast to slow: Intelligent refracting surface aided high-mobility communication," *IEEE Transactions on Wireless Communications*, vol. 21, no. 7, 2021.

[85] E. Basar, "Reconfigurable intelligent surfaces for Doppler effect and multipath fading mitigation," *frontiers in Communications and Networks*, vol. 2, 2021.

[86] G. Liu, A. Liu, R. Zhang, and M. Zhao, "Angular-domain selective channel tracking and Doppler compensation for high-mobility mmWave massive MIMO," *IEEE Transactions on Wireless Communications*, vol. 20, no. 5, 2020.

[87] T. Back, *Evolutionary algorithms in theory and practice: Evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[88] M. K. Shehzad, A. Ahmad, S. A. Hassan, and H. Jung, "Backhaul-aware intelligent positioning of UAVs and association of terrestrial base stations for fronthaul connectivity," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 4, 2021.

[89] M. R. Akdeniz, Y. Liu, M. K. Samimi, S. Sun, S. Rangan, T. S. Rappaport, and E. Erkip, "Millimeter wave channel modeling and cellular capacity evaluation," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 6, 2014.

[90] 3GPP, *Study on channel model for frequencies from 0.5 to 100 GHz*, https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3173, Accessed on 2023-12-12. [online].

[91] W. Jiang and H. D. Schotten, "A comparison of wireless channel predictors: Artificial intelligence versus Kalman filter," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019.

[92] H. Kim, S. Kim, H. Lee, and J. Choi, "Massive MIMO channel prediction: Machine learning versus Kalman filtering," in *2020 IEEE Globecom Workshops (GC Wkshps*, 2020.

[93] A. Hirose, *Complex-valued neural networks: theories and applications*. World Scientific, 2003, vol. 5.

[94] C. Lee, H. Hasegawa, and S. Gao, "Complex-valued neural networks: A comprehensive survey," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 8, 2022.

[95] E. Björnson and L. Sanguinetti, "Rayleigh fading modeling and channel hardening for reconfigurable intelligent surfaces," *IEEE Wireless Communications Letters*, vol. 10, no. 4, 2021.

[96] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data*, vol. 6, no. 1, 2019.