IEEE Open Journal of the Communications Society

Received XX Month, XXXX; revised XX Month, XXXX; accepted XX Month, XXXX; Date of publication XX Month, XXXX; date of current version 11 January, 2024.

Digital Object Identifier 10.1109/OJCOMS.2024.011100

Model-Free Robust Beamforming in Satellite Downlink using Reinforcement Learning

Alea Schröder[1,2] *(Student Member, IEEE)*, Steffen Gracla [1,2], Carsten Bockelmann [1] *(Member, IEEE)*, Dirk Wübben [1,2] *(Senior Member, IEEE)*, Armin Dekorsy [1,2] *(Senior Member, IEEE)*

[1]Department of Communications Engineering, University of Bremen, 28359 Bremen, Germany
[2]Gauss-Olbers Space Technology Transfer Center, University of Bremen, 28359 Bremen, Germany

CORRESPONDING AUTHOR: Alea Schröder (e-mail: schroeder@ant.uni-bremen.de).

This work was partly funded by the German Federal Ministry of Research, Technology and Space (BMFTR) under grant 16KISK016 (Open6GHub) and the European Space Agency (ESA) under contract number 4000139559/22/UK/AL (AIComS)

ABSTRACT Satellite-based communications are expected to be a substantial future market in 6G networks. As satellite constellations grow denser and transmission resources remain limited, frequency reuse plays an increasingly important role in managing inter-user interference. In the multi-user downlink, precoding enables the reuse of frequencies across spatially separated users, greatly improving spectral efficiency. The analytical calculation of suitable precodings for perfect channel information is well studied, however, their performance can quickly deteriorate when faced with, e.g., outdated channel state information or, as is particularly relevant for satellite channels, when position estimates are erroneous. Deriving robust precoders under imperfect channel state information is not only analytically intractable in general but often requires substantial relaxations of the optimization problem or heuristic constraints to obtain feasible solutions. Instead, in this paper we flexibly derive robust precoding algorithms from given data using reinforcement learning. We describe how we adapt the applied Soft Actor-Critic learning algorithm to the problem of downlink satellite beamforming and show numerically that the resulting precoding algorithm adjusts to all investigated scenarios. The considered scenarios cover both single satellite and cooperative multi-satellite beamforming, using either global or local channel state information, and two error models that represent increasing levels of uncertainty. We show that the learned algorithms match or markedly outperform two analytical baselines in sum rate performance, adapting to the required level of robustness. We also analyze the mechanisms that the learned algorithms leverage to achieve robustness. The implementation is publicly available for use and reproduction of the results.

INDEX TERMS 6G, beamforming, robustness, non-terrestrial networks, NTN, LEO, machine learning, reinforcement learning

I. INTRODUCTION

THIS paper examines the use of deep Reinforcement Learning (RL) for the problem of beamforming in satellite-to-earth downlink. In future mobile communications, satellites are expected to work in cooperation with traditional terrestrial networks, rather than being a competing mode of connection. Satellite-empowered Non-Terrestrial Networks (NTN) can bring benefits that complement the weaker points of terrestrial networks, primarily by offering direct connectivity to areas with low coverage, e.g., remote livings, ships at sea or aircrafts during flight, and disaster-stricken areas. The development of widely accessible satellite communications was spurred in recent years by the advancements in the field of smaller, less expensive Low Earth Orbit (LEO) satellites, with their lower orbit height between $600\,\mathrm{km}$ and $1000\,\mathrm{km}$ offering short round-trip latency. The satellite communications market is expected to grow rapidly, e.g., [1] estimating the space industry to

surge to a volume of $\$1$ Trillion by 2040, over half of that from mobile connectivity applications.

In order to achieve these goals, spectral efficiency is an ongoing topic of discussion. With competition for bandwidth and existing terrestrial network interference, satellites may opt to use beamforming to enable Spatial Division Multiple Access (SDMA). In beamforming, antenna arrays may be used to shape radiation patterns towards a desired spatial direction while suppressing interference in other directions. This enables satellites to reuse the same frequency for multiple users in different positions, thereby increasing spectral efficiency. Particularly with direct-to-handheld communications in mind, joint beamforming among multiple satellites allows very narrow transmission beams, separating users that are fairly close to each other. Among others, we have shown in prior work [2] that beamforming SDMA can indeed lead to increased spectral efficiency in the NTN context.

Mathematically, beamforming is performed by purposefully overlapping multiple simultaneous precoded transmissions, i.e., from multiple antennas, based on the available Channel State Information at Transmitter (CSIT) [3]. We describe this process in detail in the following section. The calculation of optimal precodings is well known from terrestrial communications, and typical precoders such as Minimum Mean Squared Error (MMSE) precoding do a serviceable job when applied to NTN conditions [4]. In satellite downlink, conventional CSIT estimation based on uplink Channel State Information (CSI) can be inaccurate, as the uplink conditions may not reliably reflect the downlink channel. Therefore, we assume that the CSIT in satellite downlink Line of Sight (LoS) channels is primarily obtained from positioning information instead. However, such position-based CSIT can become outdated quickly. This is due to the large distances, high satellite velocities, and synchronization delays. As a result, maintaining accurate channel knowledge becomes particularly challenging. Using such erroneous or outdated channel information will quickly degrade the performance of typical precoding algorithms. Conventionally, channel models can be extended with an error model that tracks reality as closely as possible, and new precoding algorithms tailored to these erroneous channels can be derived analytically, e.g., [5]–[7]. However, these error models are merely an approximation of reality, and mathematical intractability may force a relaxation of the problem regardless. This work looks particularly at the problem of sum rate maximization in the presence of erroneous CSIT, which is known to be NP-hard [5]. Further, exceeding computational complexity may preclude a precoder from consideration in the context of satellite communications even if it achieves better sum rate performance than less complex algorithms [8].

More recently, data-driven Machine Learning (ML) has become an attractive option to derive algorithms from observations rather than from a model. ML iteratively tries to find a best fit to given data, whatever the underlying real processes and error sources may be. This approach has been shown, both within the communications domain [9], [10] as well as other domains, to match or outperform traditional model-based algorithms on these types of problems. In the context of satellite precoding, e.g., [11] have used ML to approximate a lower complexity implementation of an existing robust precoder using supervised learning.

In this paper, we use the subdomain of RL [12] to tackle the problem of downlink satellite precoding. RL, as described in depth in subsequent sections, starts with a random precoding strategy and iteratively improves upon it by a process of scientific trial-and-error. Specifically, a learning agent will select a precoding for a given CSIT estimate, and receive a resulting performance metric, i.e., the sum rate. Based on such repeated interactions, the learning agent adapts its precoding strategy to promote precodings that lead to high sum rates and demote decisions that lead to low sum rates. This carries the major advantage that, in contrast to supervised learning, no pre-existing precoding algorithms, system models or pre-labeled training data are required to learn. Conversely, the problem of selecting precodings is also a particularly good fit for RL as there are no long term dependencies on the decisions, i.e., selecting a precoding at one time t will not influence future channel states at times $t + t'$. This is beneficial since long term time dependencies are a major factor of instability for RL convergence. In [13], the authors have applied a similar RL approach on a mixed Quality of Service (QoS) optimization objective using codebook style precoding. In this article, we instead demonstrate a learning algorithms that directly output precodings with no constraint to a codebook.

The predominant choices for deep RL algorithms on continuous action spaces are either on-policy like Trust Region Policy Optimization (TRPO) [14], Proximal Policy Optimization (PPO) [15], or off-policy like Deep Deterministic Policy Gradient (DDPG) [16], Soft Actor-Critic (SAC) [17]. These algorithms differ primarily not in the performance they achieve, but in how they integrate with a system. In satellite communications, we are 1) limited by satellite hardware, 2) want to limit communication overhead, and 3) want to limit unnecessary trial-and-error. Hence, the ability to learn more efficiently on ground-based computation clusters and sample efficiency are main concerns.

On-policy methods such as PPO and its variants learn stably but discard samples after only a few gradient updates, which is less efficient when generating new samples is costly. Off-policy methods, by contrast, can reuse past experiences more effectively. Among them, DDPG is a computationally lighter predecessor to SAC but tends to exhibit much slower convergence and severely lower sample efficiency [18]. Among other improvements, the SAC algorithm augments off-policy learning with a maximum-entropy objective to encourage more stable exploration and improved sample efficiency. This makes SAC particularly well suited for our offline, ground-based training setup, where computational

demands are acceptable but data efficiency is essential. We therefore select SAC and adapt it to learn precoding strategies for satellite networks. In our considered scenario, satellites collect precoding data and forward it to a ground station periodically. The ground station updates the parametrized precoding policy using a replay buffer and periodically transmits updated parameters back to the satellites.

The contributions of this work are thus:

- We adapt the vanilla SAC ML algorithm to downlink satellite SDMA precoding, for single satellite, cooperative multi-satellite with local CSIT, and cooperative multi-satellite with global CSIT. Using this, we directly output highly flexible precoding algorithms that maximize an arbitrary objective.
- We numerically evaluate learned algorithms that output high sum rate under erroneous CSIT, a NP-hard problem [5], extending first results presented in earlier works [2], [19]. We compare to analytical baseline precoders on simulated data, considering several combinations of satellite and user constellations as well as error models.
- We give particular detail on the ML-specific challenges in terms of convergence and learning sample design.
- We find that the learned precoders well adapt to diverse challenges in terms of availability and truthfulness of CSIT. Learning leads to high performance in all evaluated contexts while offering significantly increased flexibility over analytical approaches.

The paper is structured as follows. After this introduction, we establish the satellite downlink model, including satellite and user positioning as well as the wireless communications model. We then describe the models of uncertainty applied in this work and define what levels of information are available in distributed precoding. Finally, we state the sum rate maximization problem.

Following that, Section III introduces the use of RL in order to find a precoding algorithm. We step through the learning and inference processes in detail and describe the adaptations made to the baseline SAC RL algorithm. In Section IV, we define the analytical baseline precoders. They are used to gauge the learned precoders' performance, but also to show the upsides and potential downsides of analytically derived precoders that promote the use of a data-driven approach. Finally, Section V evaluates the precoders over a wide variety of scenarios and gives practical insight on the different mechanisms that promote robustness to uncertainty, followed by conclusions.

We assume prior knowledge of basic Neural Network (NN) function and wireless communication theory. Table 1 lists all abbreviations used.

Notations: In the following, we denote

- vectors in boldface lowercase letters: $\mathbf{a}$;
- matrices in boldface uppercase letters: $\mathbf{A}$;
- $\mathcal{U}, \mathcal{N}$ as continuous uniform and Normal distributions;
- sets as other uppercase calligraphic letters: $\mathcal{A}$;
- $|\cdot|$ and $\|\cdot\|$ as L1 and L2 norms;
- $\circ$ as Hadamard product;
- and $\{\cdot\}^{\mathrm{H}}$ as the Hermitian of a matrix.

Table 2 lists a key selection of denotations.

TABLE 1. List of Abbreviations

AcNN	Actor Neural Network
AoD	Angle of Departure
CrNN	Critic Neural Network
CSIT	Channel State Information at Transmitter
DL	Deep Learning
IUI	Inter-User Interference
LEO	Low Earth Orbit
LoS	Line of Sight
LR	Learning Rate
MIMO	Multiple Input Multiple Output
ML	Machine Learning
MMSE	Minimum Mean Squared Error
NN	Neural Network
NTN	Non-Terrestrial Networks
QoS	Quality of Service
RL	Reinforcement Learning
SAC	Soft Actor-Critic
SDMA	Spatial Division Multiple Access
SGD	Stochastic Gradient Descent
SINR	Signal to Interference and Noise Ratio
SLNR	Signal to Leakage and Noise Ratio
ULA	Uniform Linear Array

II. THE SATELLITE DOWNLINK MODEL

In this section, we first introduce the system setup, followed by the wireless communication model, including the LoS characteristics of the transmission and the signal design with precoding. We then discuss two uncertainty models for CSIT. One reflecting imperfect user position knowledge at the satellites, and another modeling satellite-specific phase errors, for instance due to synchronization mismatches. Building on this, we describe different models of locally available CSIT for cooperative beamforming. Here, we assume perfect synchronization between satellites, but consider that each satellite may only know its own CSIT while having limited or outdated information about others. Finally, we formulate the overall optimization objective, which is to maximize the achievable sum rate.

TABLE 2. List of Select Denotations

Variable	*Meaning*	$f(t)$
t	Simulation step	Yes
T	Training synchronization interval	No
$\mathcal{M}, M, m$	Set, number, index related to satellites	No
$\mathcal{K}, K, k$	Set, number, index related to users	No
N, n	Number, index related to a satellite's antennas	No
$\mathbf{v}, v$	Steering vector, steering vector entry	Yes
$d_{\mathcal{M}}, d_{\mathcal{K}}$	Average inter-satellite, inter-user distances	No
$\mathbf{u}, u$	Transmit data vector, data symbol	Yes
$\mathbf{x}, x$	Precoded data vector, data symbol	Yes
$\mathbf{W}, \mathbf{w}, w$	Precoding matrix, vector, entry	Yes
$\mathbf{H}, \mathbf{h}$	CSIT	Yes
$\tilde{\mathbf{H}}, \tilde{\mathbf{h}}$	Estimated CSIT	Yes
$\tilde{\mathbf{H}}_m, \tilde{\mathbf{H}}_{\text{L1},m}$	Local estimated CSIT	Yes
$\varepsilon_{\text{aod},k,m}, \varepsilon_{\text{ph},k,m}$	Error model 1, 2 realizations	Yes
$\Delta\varepsilon_{\text{aod}}, \sigma^2_{\text{ph}}$	Error model 1, 2 parametrization	No
$r, \bar{r}$	Sum rate, mean sum rate	Yes
$\boldsymbol{\mu}, \hat{Q}$	Parametrized actor, critic NN functions	on T
$\boldsymbol{\theta}_\mu, \boldsymbol{\theta}_{\hat{Q}}$	Actor, Critic NN parameters	on T
$\mathbf{s}, \mathbf{a}$	RL precoder input state, output action	Yes
$\mathcal{B}, B, b$	Batch set, size, index	on T
L	Loss function	on T

A. System Setup

We perform downlink transmissions from a set $\mathcal{M} = \{1, \dots, M\}$ of LEO satellites at an altitude d_0 working cooperatively to a set $\mathcal{K} = \{1, \dots, K\}$ users on Earth, as depicted in Fig. 1. For simplicity of demonstration, we restrict ourselves to a two dimensional geometric setup, though the principles shown in subsequent sections are not restricted to the two dimensional case. Each satellite $m \in \mathcal{M}$ is equipped with a Uniform Linear Array (ULA) of N antennas with combined transmit gain G_{Sat}, and we assume $MN \geq K$ unless stated otherwise. Each user $k \in \mathcal{K}$ has a handheld receiver with only a single receive antenna with low receive gain G_{Usr}, e.g., a mobile phone. Since the antenna elements of each satellite's ULA are spatially separated, the output of each antenna element $n \in \{1, \dots, N\}$ experiences a different phase shift on its way to the receiver. Hence, a steering vector $\mathbf{v}_{k,m} \in \mathbb{C}^{1\times N}$ describes the phase shift introduced in each antenna element relative to the geometric center of the ULA. Its entries $v_{k,m,n}$ depend on 1) the Angle of Departure (AoD) $\nu_{k,m}$ between satellite m and user k as well as 2) the inter-antenna distance $d_{n,n'}$ of two antennas n, n'. Recall that a ULA has constant inter-antenna distance, thus, $d_{n,n'} := d_n \, \forall \, n, n'$. To ensure uniform antenna distribution around the center of the array, the steering vector entries for any antenna n of satellite m calculate as [3]

$$v_{k,m,n}(\cos(\nu_{k,m})) = \exp\left(-j2\pi \frac{d_n}{\lambda} \frac{N+1-2n}{2} \cos(\nu_{k,m})\right), \quad (1)$$

with wavelength λ.

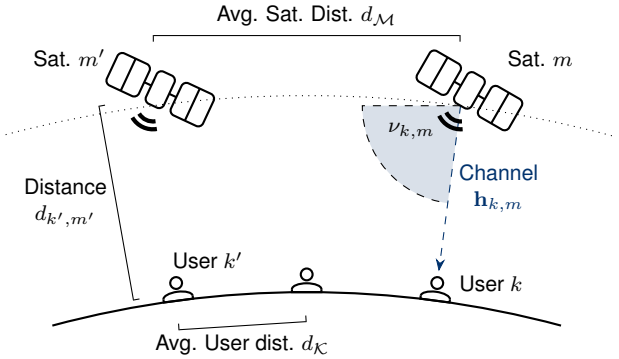

FIGURE 1. The multi satellite downlink scenario. One or more satellites $m \in \mathcal{M}$ are performing cooperative downlink transmissions to multiple users $k \in \mathcal{K}$. The CSIT $\mathbf{h}_{k,m} \in \mathbb{C}^{1\times N}$ between satellite m and user k will influence the quality of transmission. The satellites make use of estimated, potentially erroneous CSIT to cooperatively form transmission beams.

We do not consider a continuous flyover. Instead, upon a new simulation step $t \in \mathbb{N}$, all random variables are reevaluated and dependent parameters, including the positions of all satellites $\mathcal{M}$ and users $\mathcal{K}$, are updated. This applies to both the learning phases and the subsequent inference, where the mean performance of a scenario is evaluated using the Monte Carlo method. For the duration of a simulation step, all parameters are considered as constant, hence, we omit the time index t for brevity. In Table 2, we provide an overview of the parameters that change between different simulation steps t, where each step corresponds to a new channel realization and precoding instance. Satellite and user positioning, used both during learning and inference, are defined by an average inter-satellite distance $d_{m,m'} \in \mathbb{R}^+$, average inter-user distance $d_{k,k'} \in \mathbb{R}^+$ and movement ranges $\tilde{d}_{\mathcal{M}}, \tilde{d}_{\mathcal{K}} \in \mathbb{R}$ around their initial positions. The average distances are assumed as constant for all $m, m' \in \mathcal{M}, m \neq m'$ and $k, k' \in \mathcal{K}, k \neq k'$, so we denote $d_{k,k'} = d_{\mathcal{K}}$, $d_{m,m'} = d_{\mathcal{M}}$ for brevity. Satellites and users are moved from their initial positions by a value drawn from a continuous Uniform distribution $\mathcal{U}_{[-\tilde{d}_{\mathcal{M}}, \tilde{d}_{\mathcal{M}}]}$ and $\mathcal{U}_{[-\tilde{d}_{\mathcal{K}}, \tilde{d}_{\mathcal{K}}]}$ for satellites and users respectively. Thus, for a given scenario, satellites and users have constant average relative positions for $t \to \infty$, but specific realizations can vary drastically depending on the choices of movement ranges $\tilde{d}_{\mathcal{M}}, \tilde{d}_{\mathcal{K}}$.

B. Wireless Communications Model

One or multiple LEO satellites receive complex data symbols $u_k \in \mathbb{C}$ for each user $k \in \mathcal{K}$ from terrestrial data servers. Since we evaluate our results in terms of achievable rate,

the data symbols are assumed to be Gaussian distributed. The LEO satellites then further transmit these data symbols to the users $\mathcal{K}$ in downlink using their ULAs, assuming perfect synchronization among the satellites. We assume a full buffer, i.e., data symbols u_k for all users k are available in every simulation step t. The satellites' ULAs and the receiving users single antennas span a multi-user Multiple Input Multiple Output (MIMO) broadcast channel [3]. As such, to transmit the data symbols u_k, each symbol is spread across the transmit antennas using a digital precoding vector $\mathbf{w}_k \in \mathbb{C}^{MN\times 1}$, which will be discussed subsequently. All spread symbols are then superimposed to form the transmit signal

$$\mathbf{x} = \sum_{k\in\mathcal{K}} \mathbf{w}_k u_k, \quad \mathbf{x} \in \mathbb{C}^{MN\times 1}. \tag{2}$$

During transmission, we assume a linear channel where the weighted symbols $\mathbf{x}$ are distorted by the channel conditions between the satellites' antennas and the user antenna, modeled as a channel vector $\mathbf{h}_k \in \mathbb{C}^{1\times MN}$, as well as affected by additive complex Gaussian noise $n_k \sim \mathcal{CN}(0, \sigma^2_{\text{noise}})$. Thus, the receive signal y_k at user k is modeled as

$$y_k = \mathbf{h}_k \mathbf{x} + n_k \tag{3}$$

$$= \underbrace{\mathbf{h}_k \mathbf{w}_k u_k}_{\text{User Rx Symbol}} + \underbrace{\mathbf{h}_k \sum_{\substack{k'\in\mathcal{K}\\ k'\neq k}} \mathbf{w}_{k'} u_{k'}}_{\text{Inter-User Interference (IUI)}} + n_k, \tag{4}$$

with $\mathbf{h}_k = [\mathbf{h}_{k,1}, \ldots, \mathbf{h}_{k,M}]$. We model the channel $\mathbf{h}_{k,m} \in \mathbb{C}^{1\times N}$ between user k and a satellite m as a LoS channel [20] with 1) amplitude fading caused by free space path loss and random large scale fading $\varsigma \sim \text{Lognormal}(0, \sigma_\varsigma)$, 2) phase rotation due to satellite-user distance with phase shift $\varphi_{k,m} \in [0, 2\pi)$ and 3) relative phase offsets due to the spatial geometry of the ULA, collected in the previously introduced steering vector $\mathbf{v}_{k,m}$. Accordingly, we define each channel vector

$$\mathbf{h}_{k,m} = \underbrace{\frac{\lambda\sqrt{G_{\text{Sat}}G_{\text{Usr}}}}{4\pi d_{k,m}} \frac{1}{\sqrt{\varsigma}}}_{\text{Amplitude Fading}} \underbrace{\exp(-j\varphi_{k,m})}_{\text{Phase Rotation}} \underbrace{\mathbf{v}_{k,m}(\cos(\nu_{k,m}))}_{\text{Steering Vector}}. \tag{5}$$

For the amplitude fading term, no steering correction is applied as $d_n \ll d_{k,m}$. We assume the channel vector to be constant for the duration of a simulation instance t.

We see in (3) that the transmitting satellites can use knowledge of CSIT $\mathbf{H} = [\mathbf{h}_1^{\text{T}}, \ldots, \mathbf{h}_K^{\text{T}}]^{\text{T}}, \mathbf{H} \in \mathbb{C}^{K\times MN}$ (or estimates thereof) to identify a matching digital precoding $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_K], \mathbf{W} \in \mathbb{C}^{MN\times K}$ that ideally mitigates channel influence $\mathbf{h}_k\mathbf{w}_k$ and suppresses Inter-User Interference (IUI) $\mathbf{h}_k\mathbf{w}_{k'}$. The design of suitable precoding methods for perfect CSIT is well studied from terrestrial multi-user MIMO, e.g., [3], [21], though the findings do not transfer perfectly to NTN [8].

However, in practice, perfect CSIT cannot be guaranteed, which is why we consider imperfect CSIT according to Section C arising from inaccuracies in user position or mismatches in synchronization between satellites. Imperfect CSIT can significantly degrade the precoder performance [5]. We present two baseline analytical precoder designs later in Section IV. While one of these baselines is specifically designed to handle CSIT errors, its analytical derivation constrains it to predefined scenarios and limits its adaptability. In contrast, our proposed approach employs RL, which directly takes the available CSIT, perfect or imperfect, as input and can flexibly adapt the precoding strategy to the given scenario.

C. Uncertainty Models

In traditional precoding approaches, the acquisition of accurate CSIT is a prerequisite. While uplink-based CSI estimation can suffer from rapid channel aging, NTN benefit from predominantly LoS propagation, allowing CSIT to be inferred from user positions and the known satellite antenna geometry. Such position-based estimation is inherently less susceptible to channel aging, as position and geometry vary on much slower time scales than the small-scale fading effects impacting uplink-based estimation. Nevertheless, various factors can still impair the accuracy of the estimated CSIT. In LEO systems, the most relevant are the fast relative motion of satellites and users, and, particularly in cooperative multi-satellite beamforming, the latency in distributing CSIT and synchronization among satellites.

In this paper, we apply two different error sources to the CSIT estimate provided to the precoders.

Firstly, as seen in (5), the AoD $\nu_{k,m}$ from satellite m toward a user k is a key contributor to the channel phase. Thus, we can model a positioning mismatch by introducing an error on the steering vector [5]. We define a steering error vector

$$\boldsymbol{\epsilon}_{\text{aod},k,m} = \mathbf{v}_{k,m}(\varepsilon_{\text{aod},k,m}) \in \mathbb{C}^{1\times N}, \tag{6}$$

with entries analogous to (1), where $\varepsilon_{\text{aod},k,m} \sim \mathcal{U}(-\Delta\varepsilon_{\text{aod},k,m}, \Delta\varepsilon_{\text{aod},k,m}) \in \mathbb{R}$ is an error realization from a Uniform distribution. Given the relative distances, we assume that an error realization $\varepsilon_{\text{aod},k,m}$ is constant for all antennas of a satellite and assume the same error bound $\Delta\varepsilon_{\text{aod},k,m} = \Delta\varepsilon_{\text{aod}}\, \forall\, k \in \mathcal{K}, m \in \mathcal{M}$ for all users and satellites. Note that this error model still leads to different scaling of an error realization on different antennas of the same array. Effectively, element-wise multiplication of this error vector to a CSIT $\mathbf{h}_{k,m}$ translates to introducing an additive error on $\cos(\nu_{k,m})$.

As a second error source, we model a total phase error

$$\epsilon_{\text{ph},k,m} = \exp(-j\varepsilon_{\text{ph},k,m}) \in \mathbb{C} \tag{7}$$

with $\varepsilon_{\text{ph},k,m} \sim \mathcal{N}(0, \sigma^2_{\text{ph}})$ drawn from a Normal distribution for each satellite m and user k. Such a total phase error per satellite may be caused by a synchronization mismatch.

Given both errors, the CSIT estimate that a satellite m may receive for the channel to a user k is as follows

$$\tilde{\mathbf{h}}_{k,m} = \epsilon_{\text{ph},k,m}\mathbf{h}_{k,m} \circ \boldsymbol{\epsilon}_{\text{aod},k,m} \tag{8}$$

with $\tilde{\mathbf{h}}_k, \tilde{\mathbf{H}}$ defined analogously to the true CSIT. Given no error, the estimate is perfect, i.e., $\tilde{\mathbf{H}}|_{\Delta\varepsilon_{\text{aod}}=\sigma^2_{\text{ph}}=0} = \mathbf{H}$.

So far, we have considered global CSIT for precoding, i.e., all satellites have access to the same complete CSIT matrix, whether perfect $\mathbf{H}$ or imperfect $\tilde{\mathbf{H}}$. Since this assumption may not hold in practical systems due to factors such as communication overhead and delays, we subsequently introduce and investigate distributed precoding based on the local CSIT available at each satellite.

D. Local Information Models

In addition to precoding based on global information, we also consider distributed precoding using *local* CSIT at each satellite. In a cooperative precoding scenario without global CSIT, other satellites' information at each satellite may either not be available or have a higher level of uncertainty due to, e.g., being outdated by transmission delay. We consider three variants of locally available CSIT:

1) *Local Precoding*, where, at each inference step t, each satellite m has access only to its own CSIT estimate and possesses no knowledge of the CSIT of any other satellite m'

$$\tilde{\mathbf{H}}_m = [\tilde{\mathbf{h}}^{\mathrm{T}}_{1,m}, \ldots, \tilde{\mathbf{h}}^{\mathrm{T}}_{K,m}]^{\mathrm{T}}, \quad \tilde{\mathbf{H}}_m \in \mathbb{C}^{K\times N} \ . \tag{9}$$

The overall estimated channel can be written as $\tilde{\mathbf{H}} = [\tilde{\mathbf{H}}_1 \ \ldots \ \tilde{\mathbf{H}}_M]$. Each satellite then computes its own slice of the full precoding matrix $\mathbf{W}_m \in \mathbb{C}^{N\times K}$ based on its local estimate $\tilde{\mathbf{H}}_m$, such that the global precoding matrix can be assembled as $\mathbf{W} = [\mathbf{W}^{\mathrm{T}}_1, \ldots, \mathbf{W}^{\mathrm{T}}_M]^{\mathrm{T}}$.

2) *Limited 1*, where each satellite m has perfect knowledge of their own CSIT $\mathbf{H}_m \in \mathbb{C}^{K\times N}$ but erroneous CSIT $\tilde{\mathbf{H}}_{m'} \in \mathbb{C}^{K\times N}$ from other satellites m', i.e., satellite m performs the precoding based on a channel matrix

$$\begin{aligned}\tilde{\mathbf{H}}_{\text{L1},m} &= [\tilde{\mathbf{H}}_{m'<m}, \mathbf{H}_m, \tilde{\mathbf{H}}_{m'>m}], \\ \tilde{\mathbf{H}}_{\text{L1},m} &\in \mathbb{C}^{K\times MN}\end{aligned} \tag{10}$$

3) *Limited 2*, where each satellite has an erroneous estimate of their own CSIT and a further degraded estimate of all other satellites' CSIT. Specifically, we define an erroneous CSIT $\tilde{\tilde{\mathbf{H}}}_m \in \mathbb{C}^{K\times N}$ exactly like $\tilde{\mathbf{H}}_m$, except that the error realizations $\varepsilon_{\text{aod},k,m}$ are multiplied by, for example, two. Consequently, for precoding, each satellite has access to a channel matrix

$$\begin{aligned}\tilde{\mathbf{H}}_{\text{L2},m} &= [\tilde{\tilde{\mathbf{H}}}_{m'<m}, \tilde{\mathbf{H}}_m, \tilde{\tilde{\mathbf{H}}}_{m'>m}], \\ \tilde{\mathbf{H}}_{\text{L2},m} &\in \mathbb{C}^{K\times MN}.\end{aligned} \tag{11}$$

E. Problem Statement

In this paper, the measure of a precoding $\mathbf{W}$ is the achieved sum rate

$$r = \sum_{k\in\mathcal{K}} \log\left(1 + \frac{|\mathbf{h}_k\mathbf{w}_k|}{\sigma^2_{\text{noise}} + \sum_{k'\in\mathcal{K},k'\neq k}|\mathbf{h}_k\mathbf{w}_{k'}|}\right). \tag{12}$$

Given the stochastic fading channel and uncertain CSIT as defined in the prior sections, we will seek to maximize the mean sum rate

$$\bar{r} = \mathrm{E}_{\mathbf{H}}\left[\sum_{k\in\mathcal{K}} \log\left(1 + \frac{|\mathbf{h}_k\mathbf{w}_k|}{\sigma^2_{\text{noise}} + \sum_{k'\in\mathcal{K},k'\neq k}|\mathbf{h}_k\mathbf{w}_{k'}|}\right)\right] \tag{13}$$

$$= \mathrm{E}_{\mathbf{H}}[r] \tag{14}$$

which we approximate from the average sum rates achieved. The satellites also share a power budget $P \in \mathbb{R}$. Thus, we formulate the general optimization objective as

$$\max_{\mathbf{W}} \quad \bar{r} \tag{15a}$$

$$\text{s.t.} \quad \|\mathbf{W}_m\|^2 \leq P/M \quad \forall\, m \in \mathcal{M}. \tag{15b}$$

This problem is known to be NP-hard [5]. To ensure comparable results, we distribute the power budget equally among all satellites according to (15b), so that constellations with more LEO satellites do not inherently benefit from higher total transmit power. In the case of distributed precoding, instead of optimizing the global precoding matrix $\mathbf{W}$ in (15), each satellite m optimizes their own precoding matrix $\mathbf{W}_m$.

III. REINFORCEMENT LEARNED PRECODERS

Compared to reality, we can reasonably assume any mathematical representation of both the communications model and error model to be imperfect. We might also encounter a problem such as the sum rate maximization that is too complex or resource intensive to solve directly, thus working with relaxed targets or substituting approximations. As an alternative, we may forego deriving a precoding algorithm from the flawed model and instead infer it from given data. RL approaches this with a process of trial-and-error learning, where a learning agent interacts with the environment by selecting a precoding $\mathbf{W}$ for a given channel state estimation $\tilde{\mathbf{H}}$ and receiving the resulting sum rate r. Based on this experienced interaction, the agent adapts its future behavior to promote decisions that lead to high sum rate r and deemphasize actions that result in lower sum rate r. Thus, using RL, we make no model assumptions and require no pre-existing labeled data set. In typical RL parlance, this problem may also be stated as a Markov Decision Process (MDP). Here, the channel state estimations $\tilde{\mathbf{H}}$ are part of the set of possible states as spanned by the system model, the precodings $\mathbf{W}$ come from action set of all possible precodings, and the sum rate r corresponds to the reward emitted when transitioning to a new state. The state transition probability $\text{Prob}(\tilde{\mathbf{H}}_t, \tilde{\mathbf{H}}_{t+1})$ is defined by the system model discussed in the previous section.

In terms of selecting a learning algorithm from the family of RL, we can assume limitations in compute hardware and power onboard the satellites. Further, we can assume that the system, i.e., the downlink channels properties, will not structurally change on a short time scale or unexpectedly. State of the art on-policy RL algorithms such as TRPO [14], PPO [15] quickly discard learning samples once the policy is changed, reducing sample efficiency. Hence, we suggest that data samples are stored and learning is executed off-policy and offline at a terrestrial data center. The trained models can then be distributed to the satellites as required. Should the system change structurally, e.g., through changed user hardware assumptions, pre-trained models can be adapted to new data [22], though we do not consider that case in this paper. Offline training also allows for easy bootstrapping or other augmentation of the data set by simulated data, e.g., [23], which we do use in this paper. Finally, we wish to be sample efficient in selecting training trials, as generating data samples incurs a cost in terms of transmission and, potentially, system performance. We therefore opt to use the state-of-the-art SAC [17] learning algorithm, which we describe in detail in the following. It learns offline from a buffer of data samples and imposes maximum entropy in the Bayesian sense to generate high value data samples. Subsequent to that, in Section C, we will describe some additions to the vanilla SAC that were necessary to achieve high performance.

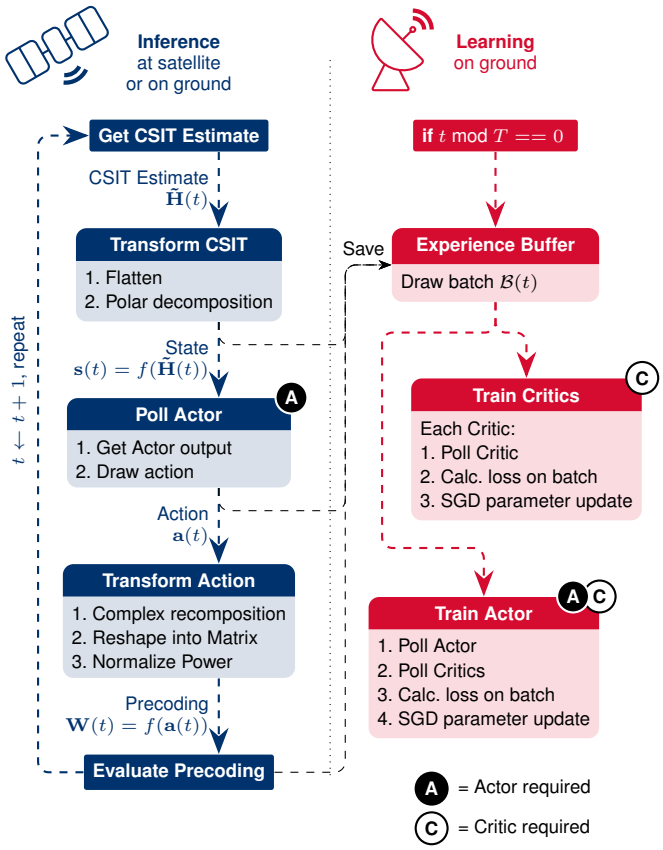

FIGURE 2. Flowchart of the inference loop at a satellite or satellite controller on the left half and a learning step at a ground-based server on the right half.

A. Soft Actor Critic

In RL, broadly, an *agent* interacts with the *environment*, in this case by selecting a precoding $\mathbf{W}$ according to Section II. This will result in a certain measure of success, e.g., a sum rate r. The agent will try to learn from this interaction to take more successful actions in the future. The core of our agent is the SAC RL algorithm. The SAC, as we use it in this paper, makes use of three NN: the Actor Neural Network (AcNN) estimates precodings $\mathbf{W}$ for a given system state. Two Critic Neural Network (CrNN) estimate the goodness of a precoding $\mathbf{W}$ for a given state and guide the AcNN to make better decisions. As an offline learning algorithm, SAC also uses an Experience Buffer to store past interaction samples. Two modules exist to transform NN inputs and outputs to embed with the system. Fig. 2 shows the SAC process flow, where we see the decoupled inference loop and learning step. In the following, we describe the design of the depicted modules in more detail, starting with the input transform. We then return to Fig. 2 to describe a full iteration of inference and learning each.

Input Transform: The estimated CSIT $\tilde{\mathbf{H}}$, as Section II describes, is the system information that our learned precoder, the AcNN, will use to decide on a precoding $\mathbf{W}$. We are using real-valued NN while the CSIT $\tilde{\mathbf{H}} \in \mathbb{C}^{K \times MN}$ is complex valued. Hence, we transform $\tilde{\mathbf{H}}$ into a flat, real valued vector $\mathbf{s} \in \mathbb{R}^{1 \times 2KMN}$ by decomposing each complex channel entry in $\tilde{\mathbf{H}}$ into two real-valued components. Specifically, for the input transform we select the decomposition into phase and amplitude information, which proves to be inductive to learning convergence. The order of values does not matter as we will not be using, e.g., convolutional NN.

Actor Neural Network: The AcNN is a fully connected NN [24] represented by a function $\boldsymbol{\mu}_{\boldsymbol{\theta}_\mu}(\mathbf{s})$ parametrized by a set of weights $\boldsymbol{\theta}_\mu$. For readability in notation we omit the parametrization index in the following. The NN takes a transformed channel state $\mathbf{s}$ as input, performs sequential nonlinear transforms on it, and outputs a vector $\boldsymbol{\mu}(\mathbf{s}) = \mathbf{a}' \in \mathbb{R}^{1 \times 4KMN}$ with four values per entry of a precoding $\mathbf{W}$. The output vector $\mathbf{a}'$ effectively decomposes into two parts of equal lengths, $\mathbf{a}' = [\mathbf{a}'_{\mathrm{m}}, \mathbf{a}'_{\sigma}]$, where $\mathbf{a}'_{\mathrm{m}}$ represents the estimated real-valued components of a precoding matrix $\mathbf{W}$ and $\mathbf{a}'_{\sigma}$ is a measure of uncertainty for each entry. During training, the uncertainty is used to sample a real-valued vector $\mathbf{a} \sim \mathcal{N}(\mathbf{a}'_{\mathrm{m}}, \exp(\mathbf{a}'_{\sigma})), \mathbf{a} \in \mathbb{R}^{1 \times 2KMN}$ from a Normal distribution. We later discuss how this is used as a built-in sample selection mechanism. During inference, we instead simply select $\mathbf{a} = \mathbf{a}'_{\mathrm{m}}$. The output transform then converts the sampled network output $\mathbf{a}$ into a real precoding matrix $\mathbf{W}$.

Critic Neural Network: Like the AcNN, a CrNN is a fully connected NN. It is represented by a function $\hat{Q}_{\boldsymbol{\theta}_{\hat{Q}}}(\mathbf{s}, \mathbf{a})$ parametrized by a set of weights $\boldsymbol{\theta}_{\hat{Q}}$. As before, we will omit the parameter index from here on for readability. A CrNN estimates the expected mean sum rate (13) as $\hat{Q}(\mathbf{s}, \mathbf{a}) = \hat{r} \in \mathbb{R}$ for a given transformed system state $\mathbf{s}$ and sampled AcNN output $\mathbf{a}$, i.e., it has only one output $\hat{r}$. CrNNs are only used in the learning step, guiding the AcNN's learning. This implementation makes use of two independently initialized, but functionally identical CrNNs $\hat{Q}_i, i \in \{1, 2\}$ for learning stability, as explained in the subsequent learning step section.

Output Transform: To transform the sampled AcNN output $\mathbf{a}$ into a complex-valued precoding matrix $\mathbf{W}$, we separate the sampled output $\mathbf{a} = [\mathbf{a}_{\text{Re}}, \mathbf{a}_{\text{Im}}]$ into two vectors $\mathbf{a}_{\text{Re}}, \mathbf{a}_{\text{Im}} \in \mathbb{R}^{1 \times KMN}$ of equal length. We pair two values from each vector such that $w_i = a_{\text{Re},i} + j a_{\text{Im},i} \, \forall \, i \in \{1, \dots, KMN\}$. We then reshape the w_i to a matrix $\mathbf{W} \in \mathbb{C}^{K \times MN}$. The specific choice of reshaping algorithm does not matter as the order is learned implicitly. Power normalization according to the power budget from (15b) is applied to the precoding matrices of each satellite, ensuring that all satellites are identical in design.

Experience Buffer: Any tuple $\text{XP} = (\mathbf{s}, \mathbf{a}, r)$ of transformed CSIT $\mathbf{s}$, sampled AcNN outputs $\mathbf{a}$ and achieved sum rate r is saved in a ring buffer with Z elements. They are indexed as XP_z with $z \in \{1, \dots, Z'\}$ where Z' is the current number of sample tuples in the buffer at simulation step t, i.e., $Z' = Z$ for a full buffer. As a ring buffer, the indexation z and number of elements Z' may change whenever a new sample is added on each t.

B. Inference Loop and Learning Step

With all elements of the vanilla SAC learning agent known, we return to Fig. 2 where the left, blue half depicts the *inference loop*. At a time t all users and satellites update their positions and the satellites obtain estimated CSIT $\tilde{\mathbf{H}}$ as described in Section II. The estimated CSIT is transformed into the AcNN input $\mathbf{s}$, and forwarded through the AcNN to obtain $\mathbf{a}$. The output transform reshapes the sampled AcNN output $\mathbf{a}$ into a matrix $\mathbf{W}$. Finally, the precoding $\mathbf{W}$ is evaluated according to Section II, resulting in a sum rate r. The simulation step t concludes by adding the obtained tuple $\text{XP} = (\mathbf{s}, \mathbf{a}, r)$ to the experience buffer. As such, an inference step is fairly light-weight and may be carried out onboard a satellite with dedicated ML hardware.

For the *learning step*, we refer to the right, red half in Fig. 2. Once the experience buffer has filled with a minimum number Z'_{min} of learning samples, we begin updating the AcNN and CrNN every $T \in \mathbb{N}$ simulation steps. Both the CrNN and AcNN networks' parameters are updated using Stochastic Gradient Descent (SGD). Hence, we select a batch $\mathcal{B} = \{\text{XP}_\mathbf{i}\}_{\mathbf{i} \sim \mathcal{U}[1, Z']}$ of experiences sampled uniformly from the current buffer. The size of the batch is denoted as $|\mathcal{B}| = B$. For the CrNNs $\hat{Q}$, the aim is to best match the concealed mapping $f(\mathbf{s}, \mathbf{a}) = r$ of the erroneously estimated transformed channel state $\mathbf{s}$ and a given sampled AcNN output to a resulting sum rate r. Hence, for the sampled batch $\mathcal{B}$ of experienced transitions, the mean square loss is

$$\text{L}_{\hat{Q},i}(\mathcal{B}) = \frac{1}{B} \sum_{(\mathbf{s}_b, \mathbf{a}_b, r_b) \in \mathcal{B}} (\hat{Q}_i(\mathbf{s}_b, \mathbf{a}_b) - r_b)^2 \tag{16}$$

for each CrNN $i \in \{1, 2\}$ and update their parameters $\boldsymbol{\theta}_{\hat{Q},i}$ accordingly via SGD,

$$\boldsymbol{\theta}_{\hat{Q},i}(t) \leftarrow \boldsymbol{\theta}_{\hat{Q},i}(t-1) - \zeta_{\hat{Q}} \nabla_{\boldsymbol{\theta}_{\hat{Q},i}} \text{L}_{\hat{Q},i}, \tag{17}$$

with a Learning Rate (LR) $\zeta_{\hat{Q}}$.

For the AcNN, we wish to find actor outputs $\mathbf{a}$ that maximize the received sum rate r for a given transformed state $\mathbf{s}$ after the output transformation. The mapping $f(\mathbf{s}, \mathbf{a}) = r$ is unknown and potentially undifferentiable. As a substitute, we have learned the known and differentiable CrNN mappings $\hat{Q}(\mathbf{s}, \mathbf{a}) = \hat{r}$. It has been shown in, e.g., [25] that misestimating the value of a state tends to be detrimental to the learning progress. Hence, using multiple, independently initialized CrNNs to guide the AcNN stabilizes the learning process, especially early on. Thus, using both CrNNs, the AcNN's estimated sum rate loss is first defined as

$$\text{L}_{\boldsymbol{\mu},1}(\mathcal{B}) = \frac{1}{B} \sum_{\mathbf{s}_b \in \mathcal{B}} - \min \left(\hat{Q}_1(\mathbf{s}_b, \mathbf{a}(\mathbf{s}_b)), \hat{Q}_2(\mathbf{s}_b, \mathbf{a}(\mathbf{s}_b)) \right). \tag{18}$$

Learning from this loss most closely corresponds to the optimization objective (15), optimizing for a maximum sum rate $\bar{r}$ in the mean sense. However, SAC also aims to find the maximum entropy solution using the AcNN's uncertainty measures for each output. Therefore, an AcNN entropy loss is defined as

$$\text{L}_{\boldsymbol{\mu},2}(\mathcal{B}) = \frac{1}{B} \sum_{\mathbf{s}_b \in \mathcal{B}} \text{logprob}_{\mathbf{a} \sim \mathcal{N}(\mathbf{a}'_\text{m}(\mathbf{s}_b), \exp(\mathbf{a}'_\sigma(\mathbf{s}_b)))} (\mathbf{a}(\mathbf{s}_b)), \tag{19}$$

using the log probabilities of the actions $\mathbf{a}(\mathbf{s}_b)$ that were sampled for the training batch states $\mathbf{s}_b$. Recall that the actions $\mathbf{a}(\mathbf{s}_b)$ are sampled from a distribution $\mathcal{N}(\mathbf{a}'_\text{m}(\mathbf{s}_b), \exp(\mathbf{a}'_\sigma(\mathbf{s}_b)))$ parametrized by the AcNN outputs $\mathbf{a}'_\text{m}(\mathbf{s}_b), \mathbf{a}'_\sigma(\mathbf{s}_b)$. Hence, the entropy loss (19) may be minimized in the mean sense by increasing the output variance, i.e., changing the actor parameters to produce higher $\mathbf{a}'_\sigma$. By increasing the variance of sampled values, the AcNN generates more varied data samples. Vice versa, output values that are known to produce significant returns, i.e., low loss (18), may be sampled with lower variance. Hence, the two losses (18), (19) are combined for the AcNN loss

$$\text{L}_{\boldsymbol{\mu}}(\mathcal{B}) = \text{L}_{\boldsymbol{\mu},1} + \exp(\zeta_\text{e}) \text{L}_{\boldsymbol{\mu},2} \tag{20}$$

with a balancing factor ζ_e. Based on this batch loss $\text{L}_{\boldsymbol{\mu}}$, the AcNN parameters $\boldsymbol{\theta}_\mu$ are also updated with an SGD update step

$$\boldsymbol{\theta}_\mu(t) \leftarrow \boldsymbol{\theta}_\mu(t-1) - \zeta_\mu \nabla_{\boldsymbol{\theta}_\mu} \text{L}_{\boldsymbol{\mu}}, \tag{21}$$

at a LR ζ_μ. The updated AcNN parameters are then distributed to the agents. However, we find that the vanilla SAC [17] as described above is not reliably able to handle the communications problem statement as described in Section II. Hence, we augment the vanilla algorithm using techniques from the domain of ML that we describe in the following. We then provide algorithm boxes that describe the full workflow.

C. Pitfalls & Additions

Applying ML often requires minute adjustments to vanilla learning algorithms that sometimes go unreported. In the following, we aim to list the less prominent challenges and adjustments enabling vanilla SAC learning to precode, as well as briefly discuss their reasoning.

Exploitation of simulation limitations: Broadly speaking, RL may extract any kind of information available from given data, which may yield unexpected performance gains over analytical approaches that do not use that particular information. Unfortunately, for synthetic data, this may include extracting patterns that would not be found in real data. If, e.g., the precoder was learning on a static constellation of user and LEO satellites, it would implicitly infer the user positions regardless of estimation error and achieve approximately perfect performance always. This would be useless in real applications. We have sanitized data generation from exploitable simulation artifacts. However, we see in the evaluation that the learned precoders utilize model assumptions that are not considered in the design of the benchmark analytical precoders, e.g., relatively lower error on the power fading component of the channel. This is explicitly a strength of data-driven algorithm design.

Low frequency network updates: With our proposed method of learning precoders offline at a ground based server and distributing the learned precoders to the satellites, it is already intuitive to not update the networks at every inference step, but only every T steps. Reducing the communication overhead is desirable, and with LEO satellites, updates may not even possible when there is no communication link between the ground server and the satellite. Additionally, a lower update frequency has also shown to be beneficial to offline learning. [26] show that the ratio of learning updates to data samples in buffer is a decisive factor in learning convergence. With less frequent updates, older, potentially outdated data samples are pushed out of the buffer more quickly while maintaining a large, varied sample data collection.

Input standardization: It is well known that, given normal SGD parameter updates, normalizing each individual NN input channel to a mean and covariance that is matched to the first layer's selected nonlinearity speeds up training [27]. Shifting the mean removes an update direction bias, and scaling the inputs balances the parameter update size. Additionally, with two different types of inputs as in our case with phase and magnitude components of $\tilde{\mathbf{H}}$, scaling balances the relative importance of each factor. However, shifting and scaling inputs requires knowledge of the data statistics, which we cannot assume as known. Using a hypothesis test, we confirm that the scaling statistics for magnitude as well as the phase variance may be determined within a relatively low number of samples, i.e., 100 samples achieve within $\pm 10\,\%$ of the true statistics at a significance level of $5\,\%$. Since these statistics are unlikely to change rapidly or at all, we consider it realistic to include a warm-up period to sample these statistics at the very beginning of a learning process. Thereafter, the warm-up does not need to be repeated unless the NN need to be retrained from scratch.

Inter layer standardization via Batch Normalization: Same as the NN inputs, normalizing the connections of the in-between layers of the NN benefits learning speed. However, unlike the first layer, the statistics of these connections shift constantly as the NN parameters are updated. Batch normalization layers [28] maintain a moving average of the statistics sampled from each learning batch $\mathcal{B}$ and normalize by these. While approximated normalization will not yield the optimum statistics, the noise introduced by normalizing with a moving average regularizes the parameter updates comparable to the noise introduced by SGD batches. We find alternatingly stacking fully connected layers and batch normalization layers to have a significant positive effect.

Weight Penalty: L2 weight regularization [24, Chp. 7] is one of the most basic regularization techniques in ML. It introduces losses

$$\mathrm{L}_{\boldsymbol{\theta}_\mu} = \frac{1}{2}\|\boldsymbol{\theta}_\mu\|^2, \tag{22}$$

$$\mathrm{L}_{\boldsymbol{\theta}_{\hat{Q}}} = \frac{1}{2}\|\boldsymbol{\theta}_{\hat{Q}}\|^2 \tag{23}$$

on the magnitude of AcNN and CrNN weights $\boldsymbol{\theta}_\mu, \boldsymbol{\theta}_{\hat{Q}}$. The optimization losses (20), (16) are regularized to

$$\mathrm{L}'_{\boldsymbol{\mu}} = \mathrm{L}_{\boldsymbol{\mu}} + \zeta_{\theta_\mu}\mathrm{L}_{\boldsymbol{\theta}_\mu}, \tag{24}$$

$$\mathrm{L}'_{\hat{Q}} = \mathrm{L}_{\hat{Q}} + \zeta_{\theta_{\hat{Q}}}\mathrm{L}_{\boldsymbol{\theta}_{\hat{Q}}} \tag{25}$$

with weights $\zeta_{\theta_\mu}, \zeta_{\theta_{\hat{Q}}}$. The benefit of this is two-fold: 1) individual parameters with little influence on an inference are pulled to zero, reducing artifacting; 2) the growth of parameters from within a single batch $\mathcal{B}$ of data is limited, preventing overly dominant paths.

With these adjustments in place, the SAC algorithm is able to reliably learn well performing precoders. Algorithms 1, 2 detail a full inference and learning step, respectively.

D. Adjustments for Distributed Precoding

In distributed scenarios, as described in Section II D, each satellite individually determines their own slice $\mathbf{W}_m$ of the full precoding matrix $\mathbf{W}$ given local CSIT. Hence, to adjust for this scenario, multiple SACs (one per satellite) are deployed and trained in parallel at the ground based computation cluster. Once trained, each NN is distributed to its corresponding satellite and precodings can be generated on-board. The NNs input and output dimensions are adjusted

Algorithm 1: Inference

Input: Estim. CSIT $\tilde{\mathbf{H}}(t)$
Data: Input norm. factors, AcNN $\boldsymbol{\mu}$
Result: Precoding $\mathbf{W}(t)$ achieves sum rate $r(t)$

$\mathbf{s}(t) \leftarrow \text{InputTransform}(\tilde{\mathbf{H}}(t))$
$\mathbf{s}'(t) \leftarrow \text{Normalize}(\mathbf{s}(t))$
$\mathbf{a}'(t) \leftarrow \boldsymbol{\mu}(\mathbf{s}'(t))$
if *GenerateTrainingSamples* **then**
  $\mathbf{a}(t) \leftarrow \mathcal{N}(\mathbf{a}(t))$; `Sample with variance`
else
  $\mathbf{a}(t) \leftarrow \mathbf{a}'_{\text{m}}(t)$
$\mathbf{W}(t) \leftarrow \text{OutputTransform}(\mathbf{a}(t))$
$r(t) \leftarrow \text{Evaluate}(\mathbf{W}(t))$; `Acc. to (12)`
Save data tuple$(\mathbf{s}'(t), \mathbf{a}(t), r(t))$
$t \leftarrow t + 1$

Algorithm 2: Learning Step

Data: Experience Buffer
Result: Parameter updates to AcNN, CrNN

if t *mod* $T = 0$ **then**
  Draw Batch $\mathcal{B}$ from Experience Buffer
  ; `Train Critics`
  for $i \in \{1, 2\}$ **do**
    for $b \in \{1, \ldots, B\}$ **do**
      $\hat{r}_b \leftarrow \hat{Q}_i(\mathbf{s}_b, \mathbf{a}_b)$
    $\text{L}'_{\hat{Q},i} \leftarrow \text{BatchLoss}(\hat{\mathbf{r}})$; `Acc. to (25)`
    $\boldsymbol{\theta}_{\hat{Q},i} \leftarrow \text{SGDStep}(\text{L}'_{\hat{Q},i})$; `Acc. to (17)`
  ; `Train Actor`
  for $b \in \{1, \ldots, B\}$ **do**
    $\mathbf{a} \leftarrow \boldsymbol{\mu}(\mathbf{s}_b)$; `Get action according to current AcNN`
    for $i \in \{1, 2\}$ **do**
      $\hat{r}_i \leftarrow \hat{Q}_i(\mathbf{s}_b, \mathbf{a})$
  $\text{L}'_{\boldsymbol{\mu}} \leftarrow \text{BatchLoss}(\hat{\mathbf{r}}_1, \hat{\mathbf{r}}_2)$; `Acc. to (24)`
  $\boldsymbol{\theta}_{\mu} \leftarrow \text{SGDStep}(\text{L}'_{\boldsymbol{\mu}})$; `Acc. to (21)`

according to the given local information. No further adjustment is required.

Next, we introduce some analytical precoding baselines before evaluating the all precoders' performance.

IV. ANALYTICAL BASELINE PRECODERS

As noted by, e.g., [8], existing precoder solutions for satellite downlink are either high-complexity or not sum rate optimal, and are not typically designed for robustness. In this work, we will look at two analytical precoder baselines to benchmark the learned precoders' performance. First, the MMSE precoder [4] is the MIMO precoding workhorse, offering relatively low complexity, while not inherently accounting for inaccurate CSIT estimations. Additionally, for single-satellite scenarios we implement a more robust analytical precoder design that maximizes the Signal to Leakage and Noise Ratio (SLNR) [5] as a proxy to the intractable Signal to Interference and Noise Ratio (SINR) (13) given the first error model as defined above, thereby including robustness into its design. Though the MMSE precoder in particular is frequently used, neither precoder directly maximizes the sum rate (15) in general. Both precoders *do* maximize the sum rate under certain conditions [5], primarily depending on accurate CSIT and orthogonal channels, and significant spatial separation may be assumed to achieve approximately orthogonal channels. However, upon violating these special conditions, both precoders' sum rate performance may degrade quickly, as we see in the evaluation. Further, extending the robust SLNR precoder to both multi-satellite and multi-user is not trivial. Complications such as these, namely the intractability of directly maximizing the sum rate analytically and inflexibility in system assumptions, are part of the motivating factors for exploring the use of ML techniques. In the following, we introduce mathematically the implementation of the MMSE and robust SLNR precoders for the given scenario.

First, the MMSE precoder finds its precoding $\mathbf{W}_{\text{MMSE}}$ such that

$$\mathbf{W}_{\text{MMSE}} = \sqrt{\frac{P}{\text{tr}\{\mathbf{W}'^{\text{H}}\mathbf{W}'\}}} \cdot \mathbf{W}' \tag{26}$$

$$\text{with} \quad \mathbf{W}' = \left[\tilde{\mathbf{H}}^{\text{H}}\tilde{\mathbf{H}} + \sigma^2_{\text{noise}} K/P \cdot \mathbf{I}_{MN}\right]^{-1} \tilde{\mathbf{H}}^{\text{H}}, \tag{27}$$

balancing channel distortion and noise power. For a fair comparison, we also normalize each satellite's precoding slice to have equal maximum power $P_m \leq P/M$, same as the learned precoders. The robust SLNR precoder uses the user k's channel power σ_k and the steering vector's autocorrelation $\mathbf{R}_k$ to determine its precoding slices $\mathbf{w}_{\text{SLNR},k}$ as

$$\mathbf{w}_{\text{SLNR},k} = \frac{P}{K}\boldsymbol{\zeta}_{k,\max}, \tag{28}$$

where $\boldsymbol{\zeta}_{k,\max}$ is the eigenvector that corresponds to the largest eigenvalue of

$$\left(\sum_{k' \in \mathcal{K}, k' \neq k} \sigma^2_{k'}\mathbf{R}_{k'} + \sigma^2_{\text{noise}} K/P \cdot \mathbf{I}_N\right)^{-1} \sigma^2_k \mathbf{R}_k. \tag{29}$$

The robust SLNR precoder design assumes an error that affects the steering vector, such as the first error model (6). Recall that this error model causes the antennas of the satellites' ULAs to have differently scaled errors depending on their positions. Hence, using the steering autocorrelation $\mathbf{R}_k$ effectively distributes the transmit power to antennas less affected by error, thereby creating wider beams. To calculate the autocorrelation $\mathbf{R}_k$, the error statistics, i.e., the error's distribution and its parametrization, must be known in advance. The precoding slices $\mathbf{w}_{\text{SLNR},k}$ are then stacked as $\mathbf{W}_{\text{SLNR}} = [\mathbf{w}_{\text{SLNR},1}, \ldots, \mathbf{w}_{\text{SLNR},K}]$, $\mathbf{W}_{\text{SLNR}} \in \mathbb{C}^{MN \times K}$.

A. Adaptations for Distributed Precoding

We use the MMSE precoder as distributed precoding performance benchmark. For the local case (9), each satellite m calculates their precoder slice $\mathbf{W}_{\text{local},m} \in \mathbb{C}^{N\times K}$ using only local information $\tilde{\mathbf{H}}_m$ as

$$\mathbf{W}_{\text{local},m} = \sqrt{\frac{P/M}{\|\mathbf{W}'_{\text{local},m}\|^2}} \cdot \mathbf{W}'_{\text{local},m} \tag{30}$$

$$\text{with} \quad \mathbf{W}'_{\text{local},m} = \left[\tilde{\mathbf{H}}_m^{\text{H}}\tilde{\mathbf{H}}_m + \sigma^2_{\text{noise}}K/P \cdot \mathbf{I}_{MN}\right]^{-1} \tilde{\mathbf{H}}_m^{\text{H}}. \tag{31}$$

In case limited information is available at satellite m, i.e., $\tilde{\mathbf{H}}_{\text{L1},m}$ (10) or $\tilde{\mathbf{H}}_{\text{L2},m}$ (11), each satellite calculates its precoding slice $\mathbf{W}_{\text{L1},m}, \mathbf{W}_{\text{L2},m}$ as follows,

$$\mathbf{W}_{\text{L1},m} = \tilde{\mathbf{H}}_m \left[\tilde{\mathbf{H}}_{\text{L1},m}\tilde{\mathbf{H}}_{\text{L1},m}^{\text{H}} + \sigma^2_{\text{noise}}K/P \cdot \mathbf{I}_K\right]^{-1}, \tag{32}$$

$$\mathbf{W}_{\text{L2},m} = \tilde{\mathbf{H}}_m \left[\tilde{\mathbf{H}}_{\text{L2},m}\tilde{\mathbf{H}}_{\text{L2},m}^{\text{H}} + \sigma^2_{\text{noise}}K/P \cdot \mathbf{I}_K\right]^{-1}. \tag{33}$$

The precoding slices $\mathbf{W}_{\text{local},m}$ are stacked to form a full precoding matrix $\mathbf{W}_{\text{local, MMSE}} = [\mathbf{W}_{\text{local},1}^{\text{T}}, \dots, \mathbf{W}_{\text{local},M}^{\text{T}}]^{\text{T}}$ and analogously for $\mathbf{W}_{\text{L1, MMSE}}, \mathbf{W}_{\text{L2, MMSE}}$. The full precoding matrix is evaluated according to Section II, the results of which we analyze alongside all other introduced precoders and scenarios in the upcoming section.

V. EVALUATION

With both the data-driven precoder learning process and the benchmark precoders introduced, we now evaluate and compare them on a selection of scenarios. Evaluation of learned precoders is done after a learning section is concluded, i.e., $T \to \infty$ and the parameters are not updated during evaluation. Successful convergence of the learning algorithm is assumed once no improvement in mean performance is observed over an extended period. We investigate single satellite and multi-satellite scenarios, primarily in terms of sum rate under increasingly unreliable CSIT. We aim to present scenarios where the benchmark precoders perform strongly as well as scenarios where the benchmarks struggle to achieve high sum rates, as described in the preceding Section IV. We show that the learning process adapts well to the diverse scenarios and level of availability of information. In particular, we show that decreasing reliability of CSIT during training leads to increasingly robust precodings, at the cost of peak performance.

A. Implementation Details

Table 3 and Table 4 list the selected system and learning parameters. In multi-satellite scenarios, a reduced number of total antennas is sufficient since the wide inter-satellite distance $d_{\mathcal{M}} \gg d_n$ allows narrow beams even with a low number of antennas. We primarily evaluate the achieved sum rate (12) averaged over 5000 Monte Carlo evaluations per evaluation point. The rate that a precoder achieves for an evaluation varies not just due to the stochastic components of the error, but also due to the relative positioning of satellites and users. In general, when user channels are more highly correlated, the achievable rate for a specific evaluation is lowered. While different precoding strategies significantly affect the mean achieved sum rate, their variance around the mean is primarily determined by the realizations of positioning and error rather than by choice of algorithm. Hence, we omit presenting the variance for readability. The full code implementation and trained models are provided online [29]. We also note that the selected learning parameters do not represent the optimum, rather, we set most training parameters to reasonable defaults and use the Optuna library [30] to search for good solutions for the remaining parameters, as highlighted in Table 4. More thorough learning parameter optimization, while not within the scope of this paper, would yield slightly more effective precoding algorithms, though the presented results sufficiently show the potential of the approach. The parameters will also have to be adapted to the specific deployed hardware.

During training, the learned precoding strategies are evaluated over batches of simulation steps t due to the non-stationary nature of the satellite downlink model. We denote these batches as *episodes*. Thus, the total number of sim-

TABLE 3. List of System Parameters

Noise Power σ^2_{noise}	6e-13 W	Tx Power P	100 W
Satellite Altitude d_0	600 km	Antennas N	$\{2, 16, 32\}$
Users K	$\{3, 6, 9\}$	Satellites M	$\{1, 2\}$
Wavelength λ	c/2 GHz	LS Fading Scale σ_ς	0.1
User Gain G_{Usr}	0 dBi	Satellite Gain G_{Sat}	20 dBi
ULA spacing d_n	$3/2\lambda$		
Avg. User Dist $d_{\mathcal{K}}$	$\{1\,\text{km}, 10\,\text{km}, 100\,\text{km}\}$		
User Dist. Roam $\tilde{d}_{\mathcal{K}}$	$\{500\,\text{m}, 5\,\text{km}, 50\,\text{km}\}$		
Avg. Sat. Dist $d_{\mathcal{M}}$	$\{10\,\text{km}, 100\,\text{km}\}$		
Sat. Dist. Roam $\tilde{d}_{\mathcal{M}}$	0 m		

TABLE 4. List of RL Parameters

[1]Entropy Scale ζ_{e}	1.0	[1]Min. Samples Z'_{min}	1000
[1]Batch Size B	1024	[1]Buffer Size Z	100 000
[1]L2 Actor ζ_{θ_μ}	0.01	[1]L2 Critic $\zeta_{\theta_{\hat{Q}}}$	0.01
[1]Training Interval T	10	[1]Optimizer	Adam [31]
[2]Actor LR ζ_μ	4.2×10^{-5}	[2]Critic LR $\zeta_{\hat{Q}}$	8.8×10^{-6}
[1]Training Episodes	13 000	[1]Steps per Episode	1000
[1]Actor Activation	Penalized Tanh [32]		
[1]Critic Activation	Leaky ReLU [32]		
[1]Actor Network Size	$[2KMN, 512, 512, 512, 512, 4KMN]$		
[1]Critic Network Size	$[4KMN, 512, 512, 512, 512, 1]$		

[1]Parameter is set using reasonable defaults. [2]Parameter is subsequently set by TPESampling search using the Optuna library [30].

ulation steps during one learning process is the number of episodes multiplied by the number of steps per episode.

Though we do not enumerate all learned precoders for brevity, one or multiple distinct strategies are learned for each scenario. We indicate precoders that learned from perfect CSIT by blue color and square markers, and precoders that learned from imperfect CSIT likewise by red colors and diamond markers. Further, where multiple strategies are learned for a scenario, we mention the operating point during training in the legend, i.e., "SAC$\Delta\epsilon = 0.25$" is trained with data generated at an error bound of $\Delta\varepsilon_{\text{aod}} = 0.25$. Note that the axis scaling differs between plots of different scenarios to highlight the most relevant cross-section of data.

B. Results

1) Centralized Precoding

We begin with a single satellite scenario with wide mean inter-user spacing of $d_{\mathcal{K}} = 100\,\text{km}$, $K = 3$ users and $N = 16$ satellite antennas. This scenario is selected to enable strong performance in the baseline precoders, which approach the optimal sum rate for without error (MMSE) and including error (robust SLNR). As the system model describes, each precoder is presented with a certain CSIT estimate $\tilde{\mathbf{H}}$ and selects a precoding $\mathbf{W}$ based on it. The precoding is evaluated in terms of sum rate r achieved when a precoding $\mathbf{W}$ is applied to the true CSIT $\mathbf{H}$. We compare the MMSE and robust SLNR precoders to multiple learned precoding strategies SAC, and evaluate the precoders' performance over an increasing error bound $\Delta\varepsilon_{\text{aod}}$ according to the error model presented in Section II.

Fig. 3 shows the mean achieved sum rates. We see strong performance from the baseline precoders. Among the learned precoders, the precoder learned with perfect CSIT, i.e., $\Delta\varepsilon_{\text{aod}} = 0$, only barely approaches the analytical precoders' performance. As the CSIT degrades, we see that this learned precoder maintains strong performance, showing increased robustness over the MMSE baseline precoder, though slightly falling behind the robust SLNR baseline precoder. With the two robust learned algorithms $\Delta\epsilon = 0.025$ and $\Delta\epsilon = 0.05$, we also see that, by using increasingly unreliable training data, we can produce increasingly robust learned algorithms. For the given error model, antennas are afflicted with increasingly severe error realizations the further away they are from the center of the antenna array. The robust SLNR precoder, as described in the preceding section, effectively uses this information to allocate more power to antennas closer to the center. Looking at the robust learned precoder NN's input weights, we infer that it has also learned to focus the center antennas, allocating on average about $23\,\%$ more weight to phase information from the inner third of antennas compared to the outer thirds.

Moving on to scenarios that less favor the baseline algorithms, Fig. 4 shows performance in a single satellite scenario with a relatively closer mean inter-user distance of $d_{\mathcal{K}} = 10\,\text{km}$. In the single satellite scenario, the baseline

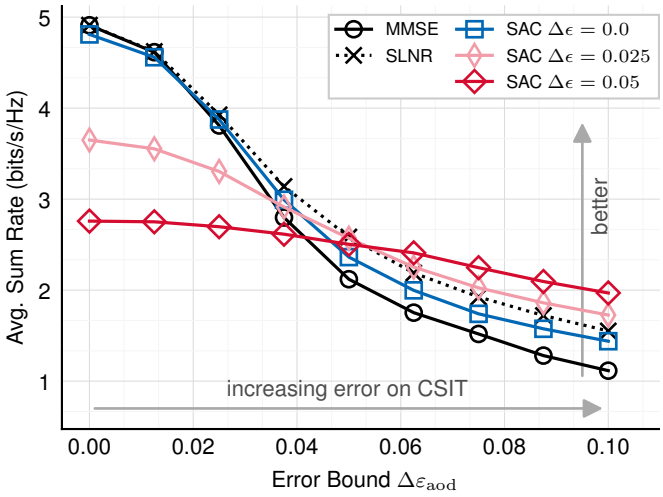

FIGURE 3. Average precoding sum rate performance in single satellite downlink at mean inter-user distance $d_{\mathcal{K}} = 100\,\text{km}, \bar{d}_{\mathcal{K}} = 50\,\text{km}$ over increasingly unreliable CSIT. Three learned algorithms SAC trained with perfect or increasingly unreliable information show elevated robustness over analytical baseline algorithms MMSE, SLNR.

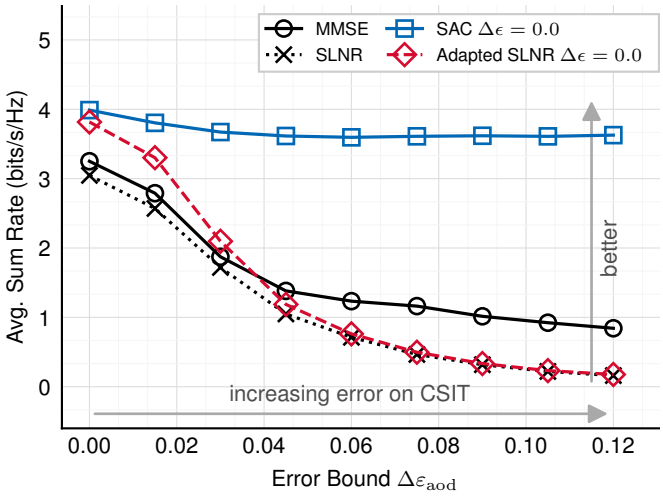

FIGURE 4. Average precoding sum rate performance in single satellite downlink at mean inter-user distance $d_{\mathcal{K}} = 10\,\text{km}, \bar{d}_{\mathcal{K}} = 5\,\text{km}$ over increasingly unreliable CSIT. The learned SAC precoder remains robust, while analytical baselines degrade with higher user correlation. As a hybrid approach, the adapted SLNR closes the SLNR's performance gap at the training point of $\Delta\epsilon = 0.0$.

algorithms struggle with these closer inter-user distances. The SLNR precoder in particular, being used outside its design space, no longer provides robustness benefits even over the MMSE precoder. SAC, even with perfect CSIT during training, learns a strategy that outperforms both baseline algorithms as well as showing strong robustness.

Fig. 5 shows a sample beam pattern for each precoder, providing intuition on why the analytical baselines struggle to achieve high sum rates with close inter-user distances. With the relatively wide beams of the single satellite setup, resolving all three users satisfactorily is only possible at the cost of high interference to the center user. Instead,

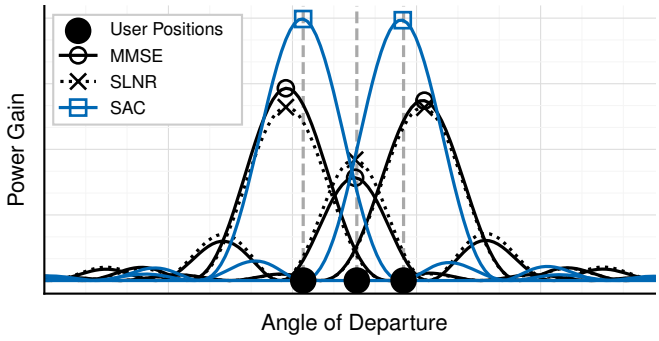

FIGURE 5. **Sample beam patterns for a constellation with a single satellite downlink and relatively low mean inter-user distance $d_{\mathcal{K}} = 10\,\mathrm{km}$. Achieved sum rates: MMSE: 3.7 bps/Hz, SLNR: 3.6 bps/Hz, SAC: 4.5 bps/Hz. At this beam-width, MMSE and SLNR precoders struggle to resolve the three users. The learned algorithm leverages power allocation to serve primarily the outer two users, which is more beneficial to maximizing sum rate in this case.**

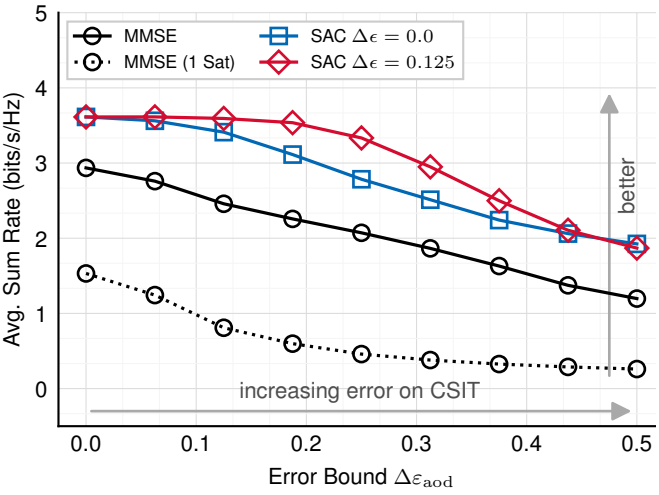

FIGURE 6. **Average precoding sum rate performance in two satellite downlink at inter satellite distance of $100\,\mathrm{km}$ and mean inter-user distance $d_{\mathcal{K}} = 1\,\mathrm{km}, \tilde{d}_{\mathcal{K}} = 500\,\mathrm{m}$ over increasingly unreliable CSIT. Cooperative two-satellite precoding achieves significantly higher sum rates than single satellite MMSE, with the learned SAC algorithms showing superior robustness and overall performance.**

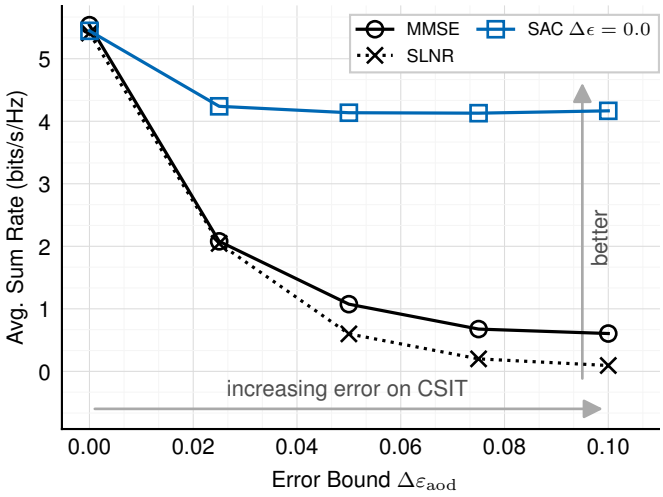

FIGURE 7. **Average precoding sum rate performance in single satellite downlink for more complex scenario with $K = 6$ users, $N = 32$ antennas at mean inter-user distance $d_{\mathcal{K}} = 10\,\mathrm{km}, \tilde{d}_{\mathcal{K}} = 5\,\mathrm{km}$ over increasingly unreliable CSIT. The learned algorithms deal well with the increased complexity of the scenario.**

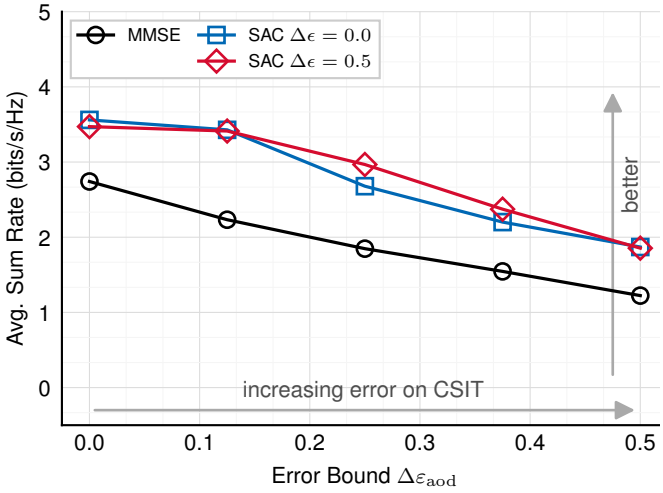

FIGURE 8. **Average precoding sum rate performance in multi satellite downlink for more complex scenario with $K = 9$ users at mean inter-user distance $d_{\mathcal{K}} = 1\,\mathrm{km}, \tilde{d}_{\mathcal{K}} = 500\,\mathrm{m}$ over increasingly unreliable CSIT. The learned algorithms deal well with the increased complexity of the scenario.**

the ML algorithms learn to leverage power allocation more aggressively, serving primarily the outer users, which turns out to be more conducive to achieving high sum rates in this scenario. If fairness is a consideration, the learning algorithm could be adapted to maximize a mixed performance objective, as other work, e.g., [13], have shown.

In the two satellite scenario shown in Fig. 6, all precoders including the analytical baselines achieve a significant sum rate benefit using two cooperative satellites over using a single satellite. However, the learned algorithms still outperform the MMSE baseline, and the robust learned precoder maintains strong sum rate performance even at high levels of unreliability in CSIT.

To confirm that the learning method scales to more complex scenarios, we double the number of antennas and number of users for the single satellite scenario, and triple the number of users for the multi-satellite scenario. The irregular antenna array spanned by multiple satellites allows fairly tight beams with a relatively low number of antennas. The results, shown in Fig. 7 for single satellite and Fig. 8 for multi-satellite, show that the learning method maintains performance for these more complex scenarios.

2) Distributed Precoding

Here, the satellites m are provided with either fully local CSIT estimates $\tilde{\mathbf{H}}_m$ (9) or limited information of the other satellites channels, $\tilde{\mathbf{H}}_{\mathrm{L1},m}$ (10) or $\tilde{\mathbf{H}}_{\mathrm{L2},m}$ (11). From this, each satellite m locally builds a precoding slice $\mathbf{W}_m$. The precoding slices are stacked and the full precoding matrix $\mathbf{W}$

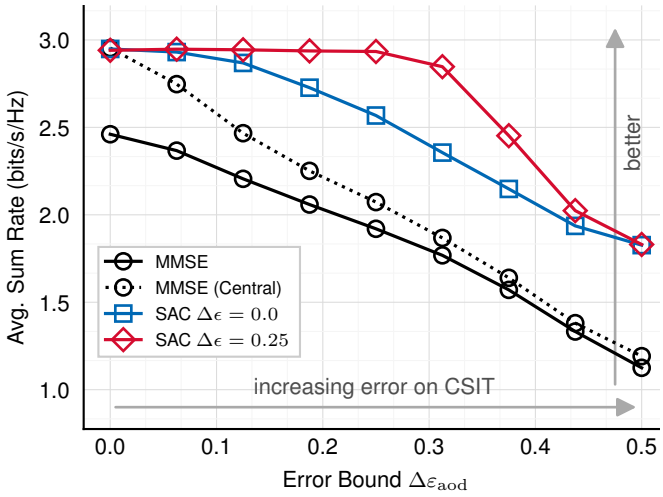

FIGURE 9. Average precoding performance for multi-satellite cooperative downlink with only local information available at each satellite. Inter satellite distance of $100\,\text{km}$ and mean inter-user distance $d_{\mathcal{K}} = 1\,\text{km}, \tilde{d}_{\mathcal{K}} = 500\,\text{m}$ over increasingly unreliable CSIT. Although all precoders degrade under local information, the learned algorithms retain their performance and robustness advantage.

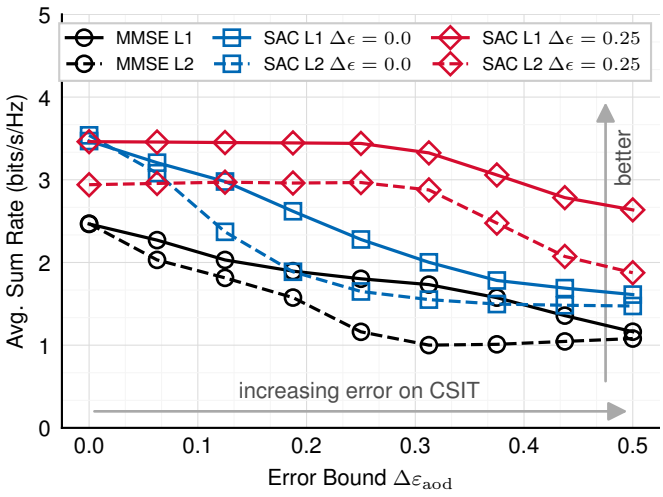

FIGURE 10. Average precoding performance for multi-satellite cooperative downlink with only limited outside information available at each satellite. Inter satellite distance of $100\,\text{km}$ and mean inter-user distance $d_{\mathcal{K}} = 1\,\text{km}, \tilde{d}_{\mathcal{K}} = 500\,\text{m}$ over increasingly unreliable CSIT. L1 (straight lines) and L2 (dashed lines) correspond to the "Limited 1" and "Limited 2" local information models.

is evaluated on the true CSIT in terms of achieved sum rate r. Fig. 9 shows the fully local scenario. While all precoders suffer from a performance dip dealing with only local information compared to the centralized precoders, the learned precoders retain their positive performance gap over the MMSE baseline. In particular, the local learned precoders manage to outperform even the centralized MMSE. Significant robustness gains are achieved by adding unreliable CSIT samples to the learning process.

In Fig. 10 we evaluate performance for local precoding with limited knowledge of other satellites' CSIT. As described above, we simulate outdated CSIT by giving each satellite perfect own CSIT and erroneous CSIT of the other satellite (L1) or erroneous own CSIT and scaled erroneous CSIT of other satellites (L2). Mirroring centralized precoding, significant sum rate and robustness gains are achieved over the baselines. Though each satellite precodes independently, the performance of centralized precoding is matched at perfect CSIT, i.e., $\Delta\varepsilon_{\text{aod}} = 0.0$, compared to Fig. 6. At L1 information, the robust learned precoders manage to even outperform the centralized precoder, which is attributed to having error-free own channel estimates available. We can trace the use of this information in the NN learned parameters, where the own phase information receives around $25\,\%$ higher weight compared to the other satellite's phase information. At L2 information, this advantage disappears and performance drops for all methods. However, our SAC-based algorithms, particularly the one trained with imperfect CSIT, maintain a substantial performance advantage over the MMSE baseline. Overall, the learned algorithms adapt well to the given level of information even in multi-agent learning scenarios.

3) Mixed Error Models

Under real-life conditions, error models may not grasp the full picture. As we have seen earlier, analytically derived algorithms' performance can degrade quickly when used outside their design space, and this holds true for the error modeling as well. We demonstrate this by applying a mixed error model as described in Section II, where we apply both positioning and phase error at the same time. We hold the positioning error bound constant at $\Delta\varepsilon_{\text{aod}} = 0.25$ and evaluate precoders at increasing phase error scale σ^2_{ph}, presented in Fig. 11. As defined earlier, the MMSE precoder has no inherent robustness mechanism, while the SLNR precoder, as described in this work, is designed for positioning errors $\varepsilon_{\text{aod},k,m}$, but not phase errors $\varepsilon_{\text{ph},k,m}$. As such, neither analytical baseline provides much robustness as the influence of the phase error $\varepsilon_{\text{ph},k,m}$ increases. Much like before, learned algorithms can be designed with increasing levels of robustness by presenting them with increasingly unreliable training data.

4) Adapting Model-Based Precoders

As we see, though the learned precoding strategies outperform the model based precoder baselines in aggregate, the analytical precoders work well at their design point as seen in, e.g., Fig. 3. For the last evaluation, we want to take a short peek at the possibility of using existing precoders as a starting point for the learned precoding strategies. To highlight this possibility, we switch the learned precoder to not directly output a precoding, but rather a matrix that scales 1) the robust SLNR precoder's power allocation; or 2) the entries of the robust SLNR precoding. We evaluate this composite precoder in Fig. 12 and Fig. 4

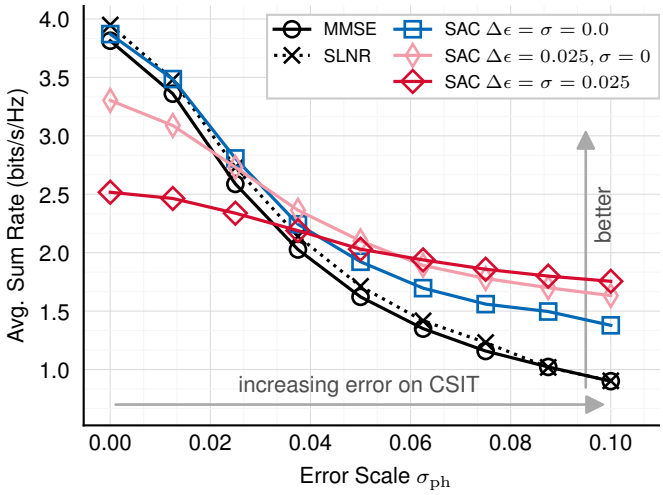

FIGURE 11. **Average precoding sum rate performance in single satellite downlink at mean inter-user distance $d_{\mathcal{K}} = 100\,\text{km}, \tilde{d}_{\mathcal{K}} = 50\,\text{km}$ over increasingly unreliable CSIT. Mixed error model, where the error scale σ^2_{ph} is sweeped while the other error source is active at a constant bound $\Delta\varepsilon_{\text{aod}} = 0.025$. The learned models show increased robustness compared to the baselines MMSE, SLNR, which provide little to no robustness on this mixed error model that lies outside their design space.**

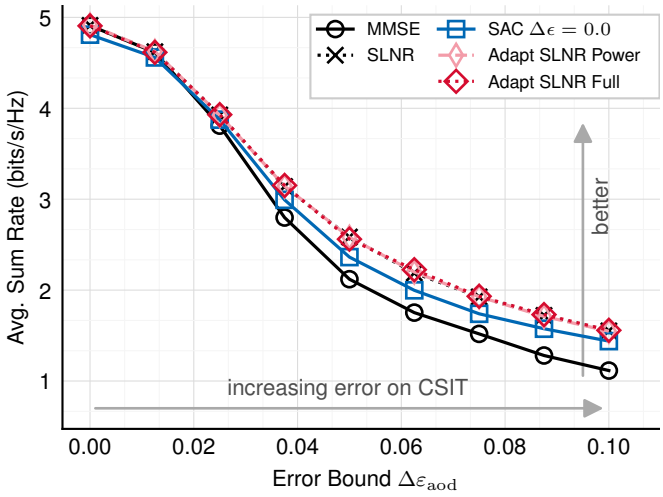

FIGURE 12. **Average precoding sum rate performance in single satellite downlink at mean inter-user distance $d_{\mathcal{K}} = 100\,\text{km}, \tilde{d}_{\mathcal{K}} = 50\,\text{km}$ over increasingly unreliable CSIT. Both learned precoders that adapt the SLNR precoder are flush in performance with it, equalizing the performance gap observed from pure learning (blue square).**

for the scenario where the robust SLNR precoder performs well and badly, respectively. We see that in the scenario where the robust SLNR precoder already performed well, using it as a starting point allows the learning approach to better match the analytical baseline. At the same time, in Fig. 4 the hybrid learning approach significantly closes the performance penalty that the analytical precoder suffers at the training point of $\Delta\varepsilon_{\text{aod}} = 0.0$. Though it does not generalize well outside that training point, it has extended the scope of the analytical precoder outside of its design space. Thus, hybrid approaches may combine the benefits of analytical and data-driven design.

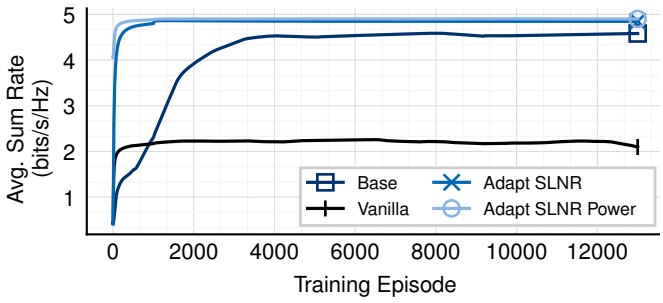

FIGURE 13. **Mean sum rate per episode achieved during training, averaged over a sliding window of length 1000. Adapting the analytical SLNR benchmark by ML methods leads to significantly faster and more stable training. Learning with a vanilla SAC algorithm without the adaptations presented in this paper converges to a local minimum solution significantly worse than other learned precodings.**

Fig. 13 displays a further big advantage of such combined approaches: convergence during training is significantly sped up by providing the SAC learning with a strong starting point. While this particular implementation is not especially suited for real applications due to the need to evaluate both the robust SLNR precoder as well as the adapting NN during inference, it highlights the potential of combining model-based precoders with learning methods for, e.g., continually learning systems.

5) Ablation Study

We train a vanilla SAC algorithm without the additions described in Section III, i.e., no low frequency network updates, no input standardization and inter layer normalization, and no weight penalty, on a scenario with 1 satellite and $100\,\text{km}$ mean inter user distance $d_{\mathcal{K}}$. As shown in Fig. 13, the vanilla SAC trained precoder reaches a low performance local minimum. Though we cannot rule out that the vanilla SAC could perform adequately given the right set of hyperparameters, we have not found such a set, and it is our experience that the additions made convergence to a high performance minimum far less sensitive to hyperparameter selection.

C. Computational Complexity

Estimating the computational complexity for analytical algorithms in advance is fairly straight forward, i.e., a MMSE precoder is dominated by the matrix inversion, which is known to scale at about $O((NM)^3)$, depending on the implementation. With NN, the computational complexity depends on the NN model size, i.e., layers and nodes per layer. Selecting the minimum model size that has the capacity to approximate the desired algorithm with sufficiently high fidelity is non-trivial and an unsolved problem in research. Practical approaches are almost exclusively heuristic, e.g., [33] or [34]. Thus, the exact scaling with problem complexity is not evaluated in this work. However, computational complexity is critical to judging the practical viability of

TABLE 5. Median Precoding Run Time on generic CPU

M	N	K	MMSE	SLNR	SAC	SAC compressed
2	2	3	10 µs	-	93 µs	63 µs
2	2	9	11 µs	-	119 µs	91 µs
1	16	3	66 µs	472 µs	94 µs	65 µs
1	32	6	123 µs	2746 µs	135 µs	94 µs

a real time algorithm, so we give an indication on the expected complexity by comparing run times for the learned precoders that we use in this work, evaluated on a generic CPU (Intel i5-1235U). With the inference as depicted in the flowchart in Fig. 2, the only steps that the learned precoder must perform in real time on constrained satellite hardware are the evaluation of the AcNN and related pre- and postprocessing. Table 5 shows the median run time of 100 000 evaluations of a learned precoder compared to the two analytical baselines for a selection of scenarios with increasing complexity. The learned precoder is overall comparable to the MMSE precoder and much faster than the robust SLNR precoder, though the MMSE precoder shows an advantage on evaluations with less parameters. In more general settings, we expect the computational complexity of the learned precoder to scale favorably for the following reasons: 1) In this paper, we use a single AcNN model size that works for all evaluations. Since some of the evaluations are strictly more complex than others, e.g., when the number of users and antennas increases, we can assume that the model is overdimensioned for at least some of the evaluations. 2) Evaluation of fully connected NN uses simple, highly parallelizable operations, such that availability of dedicated hardware on the satellite, i.e., GPU or NPU, could have a positive effect. 3) As the learned algorithms are only approximately optimal, if computational resources are constrained, it is common practice to start with an overdimensioned NN and compress the resulting learned algorithm by methods such as quantization or pruning. Though we do not investigate compression in depth in this work, we note faster run times with no loss in performance when applying generic dynamic range quantization via [35]. In terms of storage complexity, the AcNN model used in all simulations uses at most 5.5 MiB of memory, or 1.4 MiB when compressed.

D. Limitations

While a perfect learning process would result in the beamforming algorithm that is optimal at its training point, convergence and steerability remain an ongoing issue in Deep Learning (DL) research [36]. We see in, e.g., Fig. 3 that neither learned algorithm achieves perfect performance at their training point, though generalization proceeds generally as expected, i.e., the learned algorithm trained under the most unreliable CSIT exhibits the strongest robustness as error increases further. While the learned algorithms perform strongly overall, performance gaps can be further lessened through more careful selection of training parameters. Minor artifacts such as these are to be expected given current approximate, stochastic, and regularized DL methods. In situations where a straightforward and highly effective analytical algorithm is available, it is more appropriate to select traditional methods rather than learning-based approaches. For all other scenarios, selecting approximate learned precoding strategies offers a significant degree of flexibility at high performance.

VI. CONCLUSION

In this work, we show that data-driven learning can be applied to satellite communications as an extremely flexible method of deriving approximately optimal precoding algorithms. We use Reinforcement Learning (RL), a process of automated scientific trial-and-error, to iteratively improve upon a precoding strategy until is approximately maximizes a given performance objective. Primarily, this is useful as a one size fits all approach to deriving algorithms regardless of the existence of an accurate system model and mathematically tractable optimization problem. Further, the resulting algorithms all share the same Neural Network (NN) structural setup, making a change of algorithm as simple as swapping a set of parameters. We discuss what adjustments need to be made to off the shelf RL methods for the satellite downlink and show on a variety of scenarios, including single and cooperative multi-satellite with local or global information, that RL can be applied to learn precoding algorithms that achieve high sum rates. Compared to analytical baselines, the learned algorithms match performance where the baselines are near optimal and significantly outperform them otherwise. In particular, we show that the approximate nature of current Deep Learning (DL) methods lends itself well to optimizing for robustness to uncertain information, which is of notable use for real-world scenarios. With power efficient NN hardware implementations (e.g., [37]) and dedicated NN hardware onboard satellites (e.g., [38]) coming into focus, learned precoding algorithms may present an attractive option for future Non-Terrestrial Networks (NTN).

REFERENCES

[1] "The New Space Economy," https://www.morganstanley.com/Themes/global-space-economy, Accessed: 2024-03-13.
[2] S. Gracla, A. Schröder, M. Röper, C. Bockelmann, D. Wübben, and A. Dekorsy, "Learning Model-Free Robust Precoding for Cooperative Multibeam Satellite Communications," in *Proc. 2023 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)*, Rhodes Island, Greece, Jun 2023, pp. 1–5.
[3] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.
[4] S. Chatzinotas, G. Zheng, and B. Ottersten, "Energy-Efficient MMSE Beamforming and Power Allocation in Multibeam Satellite Systems," in *Proc. 2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, Pacific Grove, CA, USA, Apr 2011, pp. 1081–1085.
[5] M. Röper, B. Matthiesen, D. Wübben, P. Popovski, and A. Dekorsy, "Robust Precoding via Characteristic Functions for VSAT to Multi-Satellite Uplink Transmission," in *Proc. ICC 2023 - IEEE Interna-*

tional Conference on Communications, Rome, Italy, Jun 2023, pp. 6281–6286.

[6] S. Bazzi and W. Xu, "Robust Bayesian Precoding for Mitigation of TDD Hardware Calibration Errors," *IEEE Signal Processing Letters*, vol. 23, no. 7, pp. 929–933, May 2016.

[7] Z. Lin, M. Lin, B. Champagne, W.-P. Zhu, and N. Al-Dhahir, "Robust Hybrid Beamforming for Satellite-Terrestrial Integrated Networks," in *Proc. ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Barcelona, Spain, May 2020, pp. 8792–8796.

[8] A. I. Perez-Neira, M. A. Vazquez, M. B. Shankar, S. Maleki, and S. Chatzinotas, "Signal Processing for High-Throughput Satellites: Challenges in New Interference-Limited Scenarios," *IEEE Signal Processing Magazine*, vol. 36, no. 4, pp. 112–131, Jun 2019.

[9] C. Zhang, P. Patras, and H. Haddadi, "Deep Learning in Mobile and Wireless Networking: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2224–2287, Mar 2019.

[10] T. Naous, M. Itani, M. Awad, and S. Sharafeddine, "Reinforcement Learning in the Sky: A Survey on Enabling Intelligence in NTN-Based Communications," *IEEE Access*, vol. 11, pp. 19 941–19 968, Jan 2023.

[11] Y. Liu, Y. Wang, J. Wang, L. You, W. Wang, and X. Gao, "Robust Downlink Precoding for LEO Satellite Systems With Per-Antenna Power Constraints," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 10, pp. 10 694–10 711, Jul 2022.

[12] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[13] M. Alsenwi, E. Lagunas, and S. Chatzinotas, "Robust Beamforming for Massive MIMO LEO Satellite Communications: A Risk-Aware Learning Framework," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 5, pp. 6560–6571, Nov 2023.

[14] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. of the 32nd International Conference on Machine Learning*, vol. 37. Lille, France: PMLR, Jul 2015, pp. 1889–1897.

[15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv*, Aug 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[16] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv*, Jul 2019. [Online]. Available: https://arxiv.org/abs/1509.02971

[17] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," in *Proc. of the 35th International Conference on Machine Learning*, vol. 80. PMLR, Jul 2018, pp. 1861–1870.

[18] S. Gracla, C. Bockelmann, and A. Dekorsy, "On the Importance of Exploration for Real Life Learned Algorithms," in *Proc. of the IEEE 23rd International Workshop on Signal Processing Advances in Wireless Communication (SPAWC)*, Oulu, Finland, Jul 2022, pp. 1–5.

[19] A. Schröder, S. Gracla, M. Röper, D. Wübben, C. Bockelmann, and A. Dekorsy, "Flexible Robust Beamforming for Multibeam Satellite Downlink using Reinforcement Learning," in *Proc. ICC 2024 - IEEE International Conference on Communications*, Denver, CO, USA, Jun 2024, pp. 3809–3814.

[20] L. You, K.-X. Li, J. Wang, X. Gao, X.-G. Xia, and B. Ottersten, "Massive MIMO Transmission for LEO Satellite Communications," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 8, pp. 1851–1865, Jun 2020.

[21] T. L. Marzetta, E. G. Larsson, and H. Yang, *Fundamentals of Massive MIMO*. Cambridge University Press, 2016.

[22] M. Huisman, J. N. Van Rijn, and A. Plaat, "A survey of deep meta-learning," *Artificial Intelligence Review*, vol. 54, no. 6, p. 4483–4541, Apr 2021.

[23] "NVIDIA Replicator," https://docs.omniverse.nvidia.com/extensions/latest/ext_replicator.html, Accessed: 2024-05-31.

[24] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[25] S. Fujimoto, H. Hoof, and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods," in *Proc. of the 35th International Conference on Machine Learning*, vol. 80. PMLR, Jul 2018, pp. 1587–1596.

[26] W. Fedus, P. Ramachandran, R. Agarwal, Y. Bengio, H. Larochelle, M. Rowland, and W. Dabney, "Revisiting Fundamentals of Experience Replay," in *Proc. of the 37th International Conference on Machine Learning*, vol. 119. PMLR, Jul 2020, pp. 3061–3071.

[27] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient BackProp," in *Neural Networks: Tricks of the Trade*. Springer, 2002.

[28] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proc. of the 32nd International Conference on Machine Learning*, vol. 37. Lille, France: PMLR, Jul 2015, pp. 448–456.

[29] S. Gracla and A. Schröder, "Code Implementation for Model-Free Robust Beamforming in Satellite Downlink using Reinforcement Learning," https://github.com/Steffengra/2307_learning_beamforming_journal_code/releases/tag/v1.1.0, 2025.

[30] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," in *Proc. of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. New York, NY, USA: Association for Computing Machinery, 2019, p. 2623–2631.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv*, Dec 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

[32] S. Eger, P. Youssef, and I. Gurevych, "Is it Time to Swish? Comparing Deep Learning Activation Functions Across NLP Tasks," *arXiv*, Jan 2019. [Online]. Available: https://arxiv.org/abs/1901.02671

[33] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware," *arXiv*, Dec 2018. [Online]. Available: https://arxiv.org/abs/1812.00332

[34] M. Kaveh and M. S. Mesgari, "Application of Meta-Heuristic Algorithms for Training Neural Networks and Deep Learning Architectures: A Comprehensive Review," *Neural Processing Letters*, p. 4519–4622, Aug 2023.

[35] "tflite Dynamic Range Quantization," https://ai.google.dev/edge/litert/models/post_training_quantization#dynamic_range_quantization, Accessed: 2025-03-18.

[36] D. Hendrycks, N. Carlini, J. Schulman, and J. Steinhardt, "Unsolved Problems in ML Safety," *arXiv*, Sep 2021. [Online]. Available: https://arxiv.org/abs/2109.13916

[37] "Intel Core Ultra Ushers in the Age of the AI PC," https://www.intel.com/content/www/us/en/newsroom/news/core-ultra-client-computing-news-1.html, Accessed: 2024-09-01.

[38] M. Ghiglione and V. Serra, "Opportunities and challenges of AI on satellite processing units," in *Proc. of the 19th ACM International Conference on Computing Frontiers*. New York, NY, USA: Association for Computing Machinery, May 2022, p. 221–224.