AALBORG UNIVERSITY
STUDENT REPORT

Master's Thesis

Deep Learning for Synchronization and Channel Estimation in NB-IoT Random Access Channel

Author:
Mads Helge Jespersen

Supervisor:
Petar Popovski
Industry supervisor:
Milutin Pajovic
(Mitsubishi Electric
Research Laboratories)

This report is compiled in LaTeX. Figures are made using Inkscape.

AALBORG UNIVERSITY
STUDENT REPORT

Department of Electronic Systems
Wireless Communication Systems
Aalborg University
http://www.aau.dk

Title:
Deep Learning for Synchronization and Channel Estimation in NB-IoT Random Access Channel

Project Period:
Spring Semester 2019

Project Group:
Group 1050

Participant:
Mads Helge Jespersen

Supervisor:
Petar Popovski

Industry supervisor:
Milutin Pajovic (Mitsubishi Electric Research Laboratories)

Number of Pages:
51

Date of Completion:
June 6th, 2019

Abstract:

Effective decoding of wireless signals requires various parameter acquisition techniques including user activity detection, synchronization, channel estimation, and channel equalization. In traditional systems, these unknown, underlying parameters of the communication channel are individually estimated. This work proposes a novel joint estimation process applying deep learning. The proposed method shows superior performance to traditional methods and is further able to find the multiplicity of collisions, handle synchronization and channel estimation in the case of colliding non-orthogonal transmissions, and is able to discover superior preamble sequences using an auto-encoder structure. The proposed method is intended for decoding transmissions at the base-station in a massive connectivity scenario with many low-complexity devices operating concurrently. Excellent performance is demonstrated in estimating Time-of-Arrival (ToA), Carrier-Frequency Offset (CFO), channel gain and collision multiplicity from a received mixture of transmissions using the random access preamble structure structure of the NB-IoT standard. The proposed estimation scheme, employing a convolutional neural network (CNN), achieves a ToA Root-Mean-Square Error (RMSE) of 2.88 µs and a CFO RMSE of 3.44 Hz at 10 dB Signal-to-Noise Ratio (SNR), whereas a conventional estimator using two cascaded stages have RMSEs of 16.20 µs and 7.98 Hz, respectively.

PREFACE

This thesis is written as a part of the Master of Science in Engineering - Wireless Communication Systems at Aalborg University, Denmark.

The author would like to thank Milutin Pajovic, Toshiaki Koike-Akino and Ye Wang from Mitsubishi Electric Research Laboratories for their collaboration and supervision during the work of this thesis.

For citation the report employs IEEE referencing method. If citations are not present by figures or tables, these are made by the author of the thesis. All units are indicated according to the SI system.

Aalborg University, June 5, 2019

Mads Helge Jespersen
<mjesp14@student.aau.dk>

TABLE OF CONTENTS

ABBREVIATIONS

3GPP	3rd Generation Partnership Project
AWGN	Additive White Gaussian Noise
BS	Base Station
CDF	Cumulative Distribution Function
CFO	Carrier Frequency Offset
CNN	Convolutional Neural Networks
CP	Cyclic Prefix
CRB	Cramér–Rao lower Bound
DAE	Denoising Auto-Encoder
DL	Deep Learning
eMBB	Enhanced Mobile BroadBand
GPU	Graphics Processing Unit
HARQ	Hybrid Automatic Repeat Request
IoT	Internet of Things
LTE	Long-Term Evolution
MAC	Medium Access Control
MCRB	Modified Cramér–Rao Bound
MMSE	Minimum Mean-Square Error
mMTC	massive Machine-Type Communication
MSE	Mean Squared Error
NB-IoT	Narrowband IoT
NPBCH	Narrowband Physical Broadcast CHannel
NPDSCH	Narrowband Physical Downlink Shared CHannel
NPRACH	Narrowband Physical Random Access CHannel
NPSS	Narrow Band Primary Synchronization Signal
NPUSCH	Narrowband Physical Uplink Shared CHannel
PAPR	Peak-to-Average Power Ratio
PD	Phase-Difference
PDF	Probability Density Function
ReLU	Rectified Linear Unit
RMSE	Root-Mean-Square Error
SDR	Software-Defined Radio
SGD	Stochastic Gradient Descent
SIC	Successive Interference Cancellation
SIR	Signal-to-Interference Ratio
SNR	Signal-to-Noise Ratio
TA	Timing Advance
ToA	Time of Arrival
URLLC	Ultra Reliable Low Latency Communication
VRAM	Video Random Access Memory

1 INTRODUCTION

An emerging communication paradigm is to provide low-power and low-complexity devices with wireless Internet connectivity which is commonly referred to as Internet of Things (IoT). A challenging task in IoT development is to transform the existing human-centered communication structure into an object-centered communication system. This challenge involves redesigning many aspects of the protocol such as data representation and even changing the physical layer.

It is common to divide the communication system into three distinct services: a basic stable connection, low-power massive connectivity and reliable communication. In the 5G standard these are classified as Enhanced Mobile BroadBand (eMBB), massive Machine-Type Communication (mMTC) and Ultra Reliable Low Latency Communication (URLLC) [1]. In the traditional view of IoT communication, devices are expected to generate data packets in the order of bits that are transmitted sporadically which is consistent with the mMTC scenario and this project focuses only on low-power massive connectivity. In order to save power, most devices will not be constantly connected and are therefore required to establish a new connection for every data transmission [2]. Further, the low-complexity of devices means that short-range multi-hop type communication systems may not a suitable. For this reason, a cellular communication structure is expected to play a significant role in IoT connectivity [3]. The performance of the entire IoT system relies on which technology is used and most prominent cellular IoT technologies using unlicensed frequency bands are Sigfox and LoRa. Unlicensed spectrum communication suffers from cross-technology interference and very limited available bandwidth. Narrowband IoT (NB-IoT) is standard proposed to operate in the licensed Long-Term Evolution (LTE) spectrum by the 3GPP meaning it is backed by already-established infrastructure owners and is expected to impact the IoT connectivity landscape [2, 4]. The potentially massive number of devices attempting to gain communication access through the Base Station (BS) simultaneously may become a system bottleneck [2]. Much attention must therefore be paid to reduce signaling overhead and improve efficiency of detection algorithms in the random access phase. Random access is used to request uplink allocation from the base station and the random access procedure has a high impact on device battery life and number of devices that can be supported concurrently [5].

A typical random access procedure is initiated by a user that has a packet to transmit, which transmits a random access preamble. The random access preamble is designed such that the base station is able to efficiently detect the transmitting user and estimate any timing offset between the user and base-station from the received signal. The timing offset comprises of propagation time, downlink synchronization errors and channel delay spread [6].

1.1 PROJECT SCOPE

NB-IoT is a recent standard proposed by the 3rd Generation Partnership Project (3GPP) to accommodate the emerging number of wireless devices connected to the Internet and is chosen as the example application in this project. NB-IoT is designed to co-exist with LTE and provide low-cost and low-power devices with low throughput connectivity.

In NB-IoT random access there are only a relatively small number of different (i.e., orthogonal) random access preambles for users to choose from. Users attempting to gain channel access at the same time choose preambles in an uncoordinated manner. It is likely that two or more users choose the same preamble, given a possibly large number of users and relatively small number of orthogonal preambles. Transmitting using the same preamble results in colliding packets which results in either discarding both transmissions or only discarding one. The resulting collision may lead to a user back-off time of up to almost 9 minutes [7, 8]. The main purpose of the random access preamble is to make the base station able to detect each transmitting user and estimate the synchronization parameters, Time of Arrival (ToA) and Carrier Frequency Offset (CFO), of each user [6, 9].

In order to avoid unnecessary backoff periods and consequently improve channel utilization and overall capacity of the NB-IoT system, this project seeks to find a method for multi-user detection and synchronization parameter estimation in the case of packet collision. The non-coherent addition of colliding signals means that the received signal is simply a superposition of the individual transmissions and still contain information from each user. Multi-user detection is a method used to increase capacity by detecting interference and exploiting it to mitigate its effect on the desired signal [10]. Optimal multi-user detection methods use the Viterbi algorithm which has a high complexity which increases exponentially with the number of users. Further, most methods require exact channel information at the receiver [11]. The optimal multi-user detector is not used in practice but instead approximation such as Successive Interference Cancellation (SIC) and turbo receivers are used instead [10]. Methods exist for multi-user detection but traditional methods have several drawbacks for the scope of this project:

- Optimal decoding has exponentially increasing complexity with the number of users
- The receiver require channel information of each user
- Multi-user detection methods in literature are not developed for synchronization and channel estimation
- The receiver requires knowledge of the number of interfering signals

Deep Learning (DL)-based methods are well-suited for tackling algorithm deficit problems [12] and can utilize non-linear relationships present in the signal to extract the desired information. DL shows good results for source separation in speech processing and is well suited for classification problems such as classifying the number of interfering signals. DL algorithms are straightforward to develop and are typically not computationally complex.

In summary this project proposes a DL-based method for separating colliding users, detecting their number and estimating their respective ToAs and CFOs. The proposed method is validated using simulations which demonstrate significantly improved performance compared to the conventional approaches.

1.1.1 RELATED WORK

Several papers have explored methods for activity detection, ToA and CFO estimation using the NB-IoT Narrowband Physical Random Access CHannel (NPRACH) preamble structure. As such, [6] estimates the ToA by searching for highest correlation between the received signal and delayed/frequency-shifted preamble on a grid of possible delays and frequencies. To reduce the complexity of the algorithm in [6], the ToA and CFO are estimated using the residual phase difference between symbol groups and channel hops in a two-stage procedure in [9]. With the goal to improve the ToA estimation, [13] suggests a novel hopping pattern that renders more accurate ToA estimation compared to that achieved with the already defined NB-IoT preamble.

1.2 MACHINE LEARNING IN WIRELESS PHYSICAL LAYER

The application of machine learning has shown promise in the physical layer where optimal algorithms, e.g., in multi-user networks, tend to be computationally complex. Neural networks have been previously employed to perform detection and successive interference cancellation in multi-user CDMA systems [12]. For an overview of deep learning applied to the wireless physical layer see Appendix B.

2 NARROWBAND IOT

NB-IoT is one of the most prominent cellular techologies to accommodate the emerging number of wireless devices connected to the internet.

The system bandwidth for NB-IoT is only 180 kHz for both downlink and uplink. The uplink supports both single tone transmission and multi-tone transmission. Multi-tone transmission uses SC-FDMA with a sub-carrier spacing of 15 kHz and single-tone transmission uses a subcarrier spacing of either 15 kHz or 3.75 kHz [6]. The following description and implementations are limited to the case of single-tone transmission with a 3.75 kHz subcarrier spacing yielding a total of 48 subcarriers and this project will focus only on uplink.

NB-IoT specifies three physical layer channels: Narrowband Physical Uplink Shared CHannel (NPUSCH), Narrowband Physical Downlink Shared CHannel (NPDSCH) and NPRACH. Particularly interesting for low-complexity IoT devices is NPRACH which is used to request uplink allocation from the base station.Establishing a connection using random access is a four step procedure: First the user initiates a connection by transmitting a random access preamble, the base station transmits a response with allocated radio resources, the user transmits its identity and finally the base station transmits a contention resolution to resolve potential colliding users using the same preamble [13, 14].

2.1 PREAMBLE DESIGN

The inital premable sent by a user should provide enough information to the base station such that the start of a frame can be precisely determined (ToA estimation) and any CFO can be accounted for to improve symbol demodulation. The ToA estimation is sent by the base station to the user to achieve uplink synchronization in the OFDMA system [6].

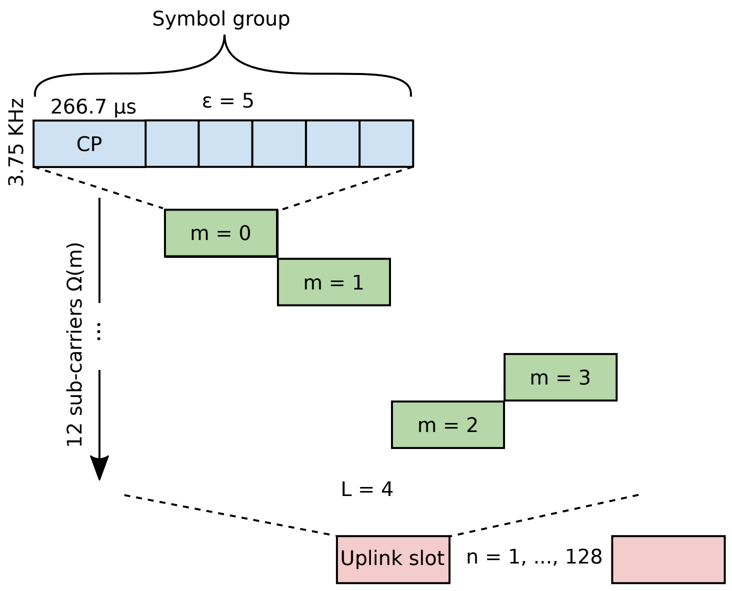


Figure 2.1: Overview of NPRACH preamble and packet structure

The preamble format and packet structure is illustrated in Figure 2.1. The preamble is divided into symbol groups, where each group consists of a Cyclic Prefix (CP) and ε identical symbols. The value of ε depends on preamble format. The preamble format is chosen by the user measuring the downlink power to estimate its coverage area [14]. The most common preamble is preamble format 1 with preamble frame structure 0 or 1 which has $\epsilon = 5$ and a symbol time $T_{\mathrm{SYM}} = 266.7\,\mu\mathrm{s}$. The CP period for frame format 0 is $T_{\mathrm{CP}} = 66.7\,\mu\mathrm{s}$ and $T_{\mathrm{CP}} = 266.7\,\mu\mathrm{s}$ for frame format 1 [15]. The CP is designed such that it is long enough to cover the maximum round trip delay to suppress inter-symbol interference. Therefore one interpretation of allowing adaptive CP selection is for the user to use the short CP in the range 0-8 km and the long CP in the range 8-35 km [6].

The full preamble consists of 4 repetitions of the symbol group which is again repeated $n = 2^J, J = 0, \dots, 7$ times for a full preamble length of $L = 4 \times 2^J$ symbol groups. The repetition of symbol groups occurs within an uplink slot, and the number of repetitions is decided by the upper Medium Access Control (MAC)-layer depending on estimated link quality [15]. For simplicity this projects lets $J = 2$ for all transmissions i.e., four symbol groups are repeated 4 times.

The user chooses a contiguous set of $K = 12, 24, 36$ or 48 subcarriers with 3.75 kHz spacing out of the available 48 subcarriers. This project focuses on preamble frame structure type 1 where $K = 12$.

At the start of the NPRACH preamble transmission, the subcarrier of the first symbol group is chosen at random. After each symbol group the subcarrier will change using a deterministic channel hopping sequence so in the duration of a preamble there will be L subcarrier hops. Since the hopping pattern is deterministic, several users choosing the same initial subcarrier will thus collide for the entirety of the NPRACH preamble sequence. The number of orthogonal preamble sequences is therefore the number of allocated NPRACH subcarriers, K [7].

The channel hopping scheme for frame structure type 1 and preamble format 0 is defined as follows: [15]

$$\tilde{n}_{\mathrm{sc}}^{\mathrm{RA}}(i) = \begin{cases} \left(\tilde{n}_{\mathrm{sc}}^{\mathrm{RA}}(0) + f(i/4)\right) \bmod N_{\mathrm{sc}}^{\mathrm{RA}} & i \bmod 4 = 0 \text{ and } i > 0 \\ \tilde{n}_{\mathrm{sc}}^{\mathrm{RA}}(i-1) + 1 & i \bmod 4 = 1,3 \text{ and } \tilde{n}_{\mathrm{sc}}^{\mathrm{RA}}(i-1) \bmod 2 = 0 \\ \tilde{n}_{\mathrm{sc}}^{\mathrm{RA}}(i-1) - 1 & i \bmod 4 = 1,3 \text{ and } \tilde{n}_{\mathrm{sc}}^{\mathrm{RA}}(i-1) \bmod 2 = 1 \\ \tilde{n}_{\mathrm{sc}}^{\mathrm{RA}}(i-1) + 6 & i \bmod 4 = 2 \text{ and } \tilde{n}_{\mathrm{sc}}^{\mathrm{RA}}(i-1) < 6 \\ \tilde{n}_{\mathrm{sc}}^{\mathrm{RA}}(i-1) - 6 & i \bmod 4 = 2 \text{ and } \tilde{n}_{\mathrm{sc}}^{\mathrm{RA}}(i-1) \geq 6 \end{cases}$$

$$f(t) = \left(f(t-1) + \left(\sum_{n=10t+1}^{10t+9} c(n) 2^{n-(10t+1)} \right) \bmod \left(N_{\mathrm{sc}}^{\mathrm{RA}} - 1\right) + 1 \right) \bmod N_{\mathrm{sc}}^{\mathrm{RA}}$$

$$f(-1) = 0$$

where $\tilde{n}_{\mathrm{sc}}^{\mathrm{RA}}(i)$ is the frequency location of the ith symbol group. In the following the symbol group index will be denoted m and the function which maps symbol groups to a frequency channel is denoted $\Omega(m)$. The function $c(n)$ is a pseudo random sequence generator that is initialised with the base station ID. This means that the hopping pattern is deterministic within a cell, but the subcarrier of every 4th symbol group appears random to neighbouring cells. The above specification means there are two "levels" of hopping as also illustrated in Figures 2.1 and 2.2. The hopping distance is 1 between symbol groups at $m = 0$ and $m = 1$, and between $m = 2$ and $m = 3$. The hopping distance is always 6 between $m = 1$ and $m = 2$.

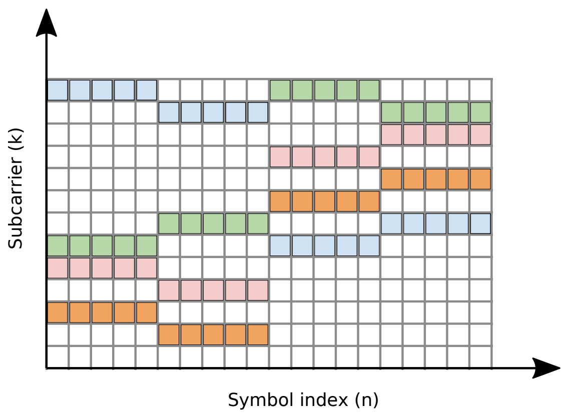


Figure 2.2: Illustration of active channels and users in the scenario where 4 users all choose different initial subcarriers. Color indicates user number where no color indicates an inactive channel slot. Horizontal axis is symbol index n where 5 consecutive symbols correspond to a symbol group. The vertical axis is subcarrier index k.

The channel hopping procedure aids in the estimation of ToA and also reduces inter- and intra-cell interference [6]. The ToA should be estimated by the base station for successful uplink signal decoding and it further enables device positioning. Error in the ToA estimation results in the user not being able to receive the response sent by the base station. ToA estimation therefore has a great impact on performance in NB-IoT [13].

2.2 BASEBAND MODEL OF NARROWBAND PHYSICAL RANDOM ACCESS CHANNEL

At the receiver the phase of each symbol depends on ToA, denoted τ, the CFO, denoted Δf and the frequency of the user's chosen channel with respect to the receiver's uplink carrier which is $\Omega(m)$. The time-domain signal at the receiver can be expressed as:

$$s(t) = e^{-j2\pi \hat{f}(t)(t-\tau(t))} \tag{2.1}$$

Where: $\hat{f}(t) = \Omega(m) + \Delta f(t)$ is received frequency with CFO [Hz]
$\tau(t)$ is time-varying ToA representing a non-stationary transmission [s]

In the baseband formulation each symbol is represented as a single complex number and therefore Equation 2.1 is discretized. The total number of symbols in the preamble (Counting the CP as a symbol) is: $N_{\text{sym}} = 4(1+\epsilon)\cdot 2^J = 24\cdot 4 = 96$. For the sake of simplicity the CFO and ToA are assumed to be constant for each user. The typical FFT length in LTE is 512 [9], but for simplicity this model lets each sample, n, corresponds to a symbol. In this model, the contents of the CP are interpreted as a symbol and therefore no distinction is made between the CP and the $\varepsilon = 5$ repeated symbols in a symbol group.

The nth time-domain sample represents the nth symbol and the signal from the kth user is given by:

$$s_k[n] = h_k\, \mathrm{e}^{-j2\pi(f_n+\Delta f_k)(nT_{\text{sym}}-\tau_k)} \tag{2.2}$$

Where:	h is the complex channel coefficient which is assumed to constant throughout a transmission	$[\cdot]$
	f_n is the subcarrier frequency which is a function of the symbol group number m: $f = \Omega(m)$	[Hz]
	Δf is CFO which is asumed to be constant throughout a transmission	[Hz]
	T_{sym} is symbol time	[s]
	τ is a constant ToA representing a stationary user	[s]

The received signal at the base station is a superposition of signals from multiple users, given by

$$y[n] = \sum_{\text{k=0}}^{K-1} a_k s_k[n] + w[n], \tag{2.3}$$

where K is the maximum number of concurrent users, $a_k \in \{0, 1\}$ indicates whether the kth user is active or not, and $w[n] \sim \mathcal{CN}(0, 1/\rho_n)$ denotes the additive noise with a per symbol Signal-to-Noise Ratio (SNR) of ρ_n.

2.2.1 TIME OF ARRIVAL

It is chosen to only model the signal for the long CP which corresponds to to distances between the user and base station within a minimum of $r = 8\,\text{km}$ and a maximum of $R = 35\,\text{km}$ [6]. The users are assumed to be uniformly distributed in the coverage area of the base station as illustrated in Figure 2.3.

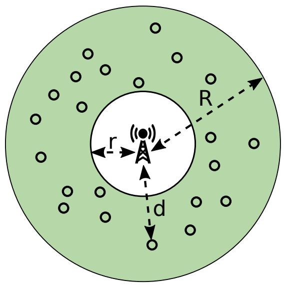


Figure 2.3: Geometry of users distribution.

The distance from the base station to the users d has the following Probability Density Function (PDF) [16]:

$$f_D(d) = \frac{2d}{R^2 - r^2}, \quad r \leq d \leq R, \tag{2.4}$$

which is used to model the ToA $\tau = \frac{d}{c}$, where c is the propagation speed.
The mean and the variance of the distance can then be found as:

$$\mathbb{E}[D] = \int_r^R d \cdot f_D(d) dd = \frac{2(r^2 + r \cdot R + R^2)}{3(r + R)} \approx 24.326\,\text{km} \tag{2.5}$$

$$\mathsf{Var}(D) = \int_r^R d^2 \cdot f_D(d)\, dd - E[D]^2 = \frac{(r - R)^2(r^2 + 4r \cdot R + R^2)}{18(r + R)^2} \approx 52.767\,\text{km}^2 \tag{2.6}$$

According to Equation 2.5 the mean value of τ becomes

$$\mathbb{E}[\tau] = \frac{24.326\,\text{km}}{3 \times 10^8\,\text{m/s}} = 81.1\,\mu\text{s} \tag{2.7}$$

The variance of the ToA is:

$$\mathsf{Var}(\tau) = \mathsf{Var}(D) \cdot \left(\frac{1}{c}\right)^2 = 0.59\,\text{ns}^2 \tag{2.8}$$

The distribution of the ToA represents a realistic cellular system but is only valid for stationary transmitters. A more complete model will include the dynamics of ToA originating from channel sampling time offset and delay spread [6] as well.

2.2.2 CARRIER FREQUENCY OFFSET

The CFO in 2.2 is chosen uniformly at random between -20 and $+20$ Hz and does not account for frequency drift [9]. The mean and variance of Δf are easily derived. The means and the variances of τ and Δf are used for feature scaling of the variables for the neural network as described in section 3.5.1.

The CFO should be estimated already in the downlink cell search procedure and this model should reflect the residual CFO that is due to imperfect estimation or oscillator drift. Again this model does not include the dynamic aspect of frequency drift and motion (Doppler) but rather considers it as a constant offset.

2.2.3 CHANNEL GAIN

The channel coefficient h of the signal model in 2.2 is a complex-valued constant which accounts for small scale fading: $h \sim \mathcal{CN}(0, 1)$. This means that the average received signal power is normalized to one. The narrowband channel is modeled as a slowly varying single-tap Rayleigh fading channel and for this reason, modeled as a single coefficient [9]. Large scale fading is not included in the model since users already have knowledge of the downlink SNR and adjust their transmit power accordingly using power control.

2.2.4 NUMBER OF ACTIVE USERS

The activity indicator a_k is modeled as Bernoulli random variable with the probability of transmitting p and $a_1, \ldots, a_K$ are iid. The number of concurrent active users is

$$N_a = \sum_{\text{k}=1}^{K} a_k \sim \mathcal{B}(K, p), \tag{2.9}$$

where $\mathcal{B}$ is the binomial distribution. The case with $K = 4$ and $p = 0.5$ is considered throughout this project. The probability of exactly k users colliding is then:

$$\Pr(k) = \binom{K}{k} p^k (1-p)^{K-k} = \frac{4!}{k!(4-k)!} p^k (1-p)^{4-k}. \tag{2.10}$$

2.3 CONNECTION ESTABLISHMENT IN NB-IOT

When a user is activated it has to select a base station to access the network through and is considered to be in idle mode. In idle mode the user acquires time and frequency synchronization using Narrow Band Primary Synchronization Signal (NPSS) which is a known sequence transmitted in every 6th out of 10 radio subframes. The user can achieve slot-time synchronization and estimate CFO using correlation between the known sequence and received signal [17]. Joint time and frequency estimation is costly for low complexity IoT devices and is therefore often implemented as a two step procedure: First the timing offset is estimated from the first received NPSS in the presence of CFO, second the CFO is estimated from subsequent received NPSSs using the estimated timing offset [17].

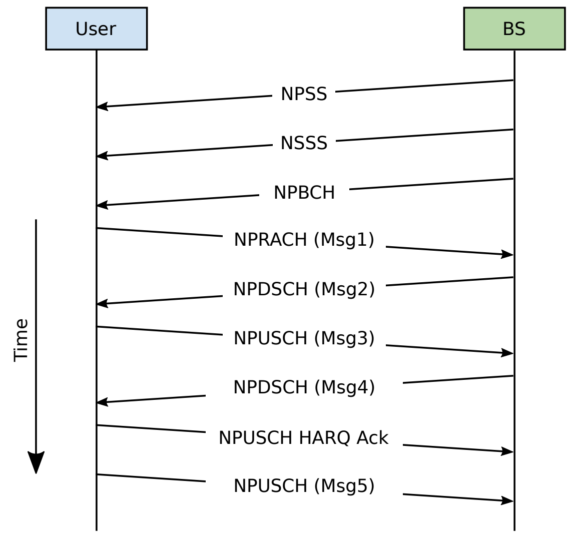


Figure 2.4: Random access procedure of NB-IoT [17]. This work focuses on NPRACH (Msg1).

Before a user can send the NPRACH it needs to be aware of the system configuration acquired from the base station through the Narrowband Physical Broadcast CHannel (NPBCH). The time-domain allocation of the NPRACH (Msg1) transmission is defined by the number of repetitions, n, chosen by the user, and a specific period starting time. The frequency allocation of the NPRACH transmission is the subset of 12 subcarriers chosen by the user [17].

The random access procedure in NB-IoT is a 5 message procedure as illustrated in Figure 2.4. The first message is the NPRACH (Described in section 2.1).

When the base station successfully detects an NPRACH it will respond on NPDSCH with Msg2 (Random Access Response). The RA Response window starts on the subframe containing the end of the preamble repetition plus 4 subframes (for preamble format 0 or 1 and $n < 64$ repetitions) [15]. The response contains a Random Access Preamble identifier, a Timing Advance (TA) parameter, and allocated radio resource for transmitting Msg3. The Random Access Preamble identifier is a number calculated from the subframe index and initial carrier frequency ($\Omega(0)$) of the received NPRACH [8]. The TA is used to time-align users' signals at the base station to account for propagation delay. The TA is an integer number between 0 and 1282 where each integer step corresponds to a time correction of $16T_\text{s}$ [18]. T_s is a basic unit of time in LTE and is defined as $T_\text{s} = 32.55$ ns, resulting in timing corrections in increments of 0.52 µs.

Msg3 is the Connection Resume Request in which a user will transmit using the allocated radio resource in the NPUSCH. Msg3 contains the user identity, a scheduling request and the user's power and buffer status [17].

It can happen that two users transmit using the same NPRACH in Msg1 and will therefore receive the same allocation for Msg3 without the user or base station being aware that a packet collision occurred.

To resolve potential contentions, the user starts a "Contention Resolution Timer" in which it expects a Msg4 response for the user identity. The user transmits incremental parity bits using Hybrid Automatic Repeat Request (HARQ) until the timer reaches zero or it receives the expected response [7].

In Msg4, the base station has resolved potential contentions and transmits a connection setup with allocated radio resources or a connection resume message. Both Msg4 and Msg3 are transmitted using HARQ [7].

In Msg5, the user responds with a connection setup complete or resume complete message. The resume procedure is used to reduce the number of message exchanges between user and base station by resuming configurations from a previously established connection [17].

2.4 TRADITIONAL SYNCHRONIZATION PARAMETER ESTIMATION

From the received NPRACH (MSG1) the BS must detect active users and perform synchronization parameter estimation.

The Phase-Difference (PD)-based method proposed in [9] utilizes the relationship between the phase trace of the received signal and the ToA and CFO. Phase differences between symbols in the received signal are averaged to estimate CFO. The ToA is found by subtracting the phase contribution due to the estimated CFO from the phase of the received signal and averaging the phase difference between symbol groups on different frequencies.

As a benchmark for the detection of the number of users, an amplitude-based estimator is considered. The mean amplitude of the received signal for different number of colliding users is compared to the amplitude of the received signal. The closest match then yields an estimate of the number of colliding users present in the received signal.

2.4.1 ESTIMATING THE NUMBER OF ACTIVE USERS

The number of active users N_a in the received signal can be estimated by comparing the amplitude of y from Equation 2.3 to the expected amplitude with varying number of users. The amplitude of the received signal is a sum of N_a complex random variables:

$$A_{N_a} = \left| \sum_{\mathrm{k}=1}^{N_a} h_k \right| \tag{2.11}$$

and will therefore vary a lot.

The mean amplitude of the received signal is $\mathbb{E}\Big[|y|\Big]$ and the mean amplitude of a signal with N_a users is $\mathbb{E}\Big[A_{N_a}\Big]$. The estimated number of active users is the closest matching value of N_a

$$\hat{N_a} = \arg\min_{N_a} \left(\left| \mathbb{E}\Big[A_{N_a}\Big] - \mathbb{E}\Big[|y|\Big] \right| \right). \tag{2.12}$$

This is only used as a benchmark for detecting the number of users N_a.

2.4.2 Estimating Carrier Frequency Offset

The synchronization parameters Δf and τ can be determined by realizing that the phase of the received signal is proportional to both CFO and ToA. The method presented in [9] is a two-step procedure where first the CFO-induced phase is estimated and then subtracted from the phase of the received signal to estimate the ToA-induced phase.

The phase-trace of a noise-free received signal for user k can be expressed [9]:

$$\beta_k[n] = -2\pi\tau_k f_n - 2\pi\Delta f_k n T_{\text{sym}} + C \tag{2.13}$$

where C is a random constant phase offset. In practice the phase-trace of the received signal is not straightforward to obtain due to 2π-ambiguity but the unwrap-function and complex argument function of the received signal provides a good approximation [9]:

$$\beta_k[n] = \text{unwrap}\left(\arg(s_k[n])\right). \tag{2.14}$$

Phase differences between symbols with the same subcarrier frequency f_n can be used to estimate the phase contribution of the CFO. Symbols groups contain five consecutive identical symbols which phase-differences should be averaged to reduce noise variance. Again then the average of all these estimates are used is used to estimate the CFO-induced phase:

$$\beta_{k,\Delta f} = \frac{1}{N}\frac{1}{4}\sum_{\text{n=0}}^{N-1}\sum_{\text{i=1}}^{4} \beta_k[5n+i+1] - \beta_k[5n+i]. \tag{2.15}$$

The CFO estimate of the kth user is then simply

$$\hat{\Delta f}_k = \frac{1}{2\pi}\beta_{k,\Delta f}. \tag{2.16}$$

This estimate is only valid if the phase-trace of the received signal only contains the contribution from a single user.

2.4.3 Estimating Time of Arrival

The estimated CFO-induced phase is subtracted from the phase-trace of the received signal:

$$\overline{\beta_k}[n] = \beta_k[n] - 2\pi\hat{\Delta f}_k n T_{\text{sym}}. \tag{2.17}$$

Differences when varying the frequency f_n of this phase-trace are averaged to filter out noise and are used to estimate ToA. The phase difference of the same symbol between channel hopping is proportional to the channel hopping distance and the ToA can be estimated more accurately using more channel hops.

This two-stage synchronization parameter estimation is computationally efficient and, the approach decouples detection and estimation [9].

Another approach which jointly detects users and estimation is presented in [6]. This method computes the correlation with the received signal and preambles with different time and frequency corrections to find the point of highest correlation. This 2-D grid search approach finds the ToA and CFO simultaneously but not very computationally efficient and does not specify a detection threshold.

3 DEEP LEARNING ESTIMATOR

3.1 DEEP LEARNING BASICS

This section briefly explains the basics of deep learning as used in the estimator described in the following sections.

3.1.1 CONVOLUTIONAL NEURAL NETWORKS

Convolutional Neural Networkss (CNNs) are a specialized type of neural networks which employs one or more convolution operations. The convolution is defined as:

$$s[n] = \sum_{\mathrm{a}=-\infty}^{\infty} x[a]w[n-a] \tag{3.1}$$

where the discrete input signal is $\mathbf{x}$ and $\mathbf{w}$ is called the kernel. A CNN typically has multiple cascaded convolutions with different kernels and the training procedure attempts to find the kernels which minimize the loss function.

CNNs have equivariance to translation meaning that the convolution operation will produce an output shifted in time by the same amount as the input is shifted in time. For this reason, a CNN can only learn to extract features which are a function of local interactions and are equivariant to translation. A typical layer in a CNN consists of three operations: Convolution over the input signal, a non-linear activation function (typically the Rectified Linear Unit (ReLU) function) and a pooling operation. The pooling operation replaces the output with a lower-dimensional statistical summary of the response over a small range such as the average within a neighborhood. This makes the output representation approximately invariant to small time-shifts. The most popular pooling-operation is the max-pooling [19]. CNNs are used extensively in image processing and speech recognition where they are useful for 1-D or 2-D data where local features affect a particular output. CNNs structures alleviates computation complexity and storage requirements compared to a feed-forward neural network [20].

3.1.2 LOSS FUNCTION

The goal of training a network is minimizing the average loss over training pairs. Each training pair consists of an input x and a target t: (x_n, t_n) and all training pairs are assumed i.i.d. The neural network should be an estimator which uses the input to predict the target, $\hat{t}(x)$. The general loss function is written $\ell(t, \hat{t}(x))$ and a typical example is squared error $\ell(t, \hat{t}(x)) = (t - \hat{t}(x))^2$ or cross-entropy for categorical models [12].

3.1.3 LEARNING

When using machine learning to minimize ℓ this is not solved analytically but instead addressed by Stochastic Gradient Descent (SGD). The model is defined as a set of probability distributions parameterized by a vector θ: $p(t|x, \theta)$. The learning task can be defined to obtain a parameter vector θ which can accurately describe this probability distribution.

Using a maximum likelihood formulation this can be written as maximizing the log-likelihood [12]:

$$\max \ln p(\mathcal{D}|\theta) \tag{3.2}$$

where $\mathcal{D}$ is the set of training pairs.

Each iteration of the SGD uses a set of training pairs called batches to update θ. The parameter vector θ is updated in the direction of the gradient of the loss function [12]:

$$\theta_{\text{new}} \leftarrow \theta_{\text{old}} + \gamma \Delta_\theta \ln p(t_x|x_n, \theta)|_{\theta=\theta_{\text{old}}} \tag{3.3}$$

where γ is the learning rate and Δ_θ is the nabla operator used to denote the gradient of $\ln p(t_x|x_n, \theta)$ with respect to θ averaged over the training batch. For a multilayered network the computation of the gradient becomes the backpropagation algorithm [12]. Increasing the number of SGD iterations decreases the loss and iterations are repeated until sufficient performance is achieved or the loss-function has converged.

3.1.4 HYPER-PARAMETERS

The learning rate γ is an example of a hyper-parameter which typically must be defined prior to training the network. Other hyper-parameters include the number of layers to include in the model, the type of layers, the number of weights in each layer and many more. Hyper-parameters must be fine-tuned in order to obtain the desired performance which is a time-consuming task. The hyper-parameters define the architecture and capacity of the network as well as how the network is trained. A network with too little capacity will not generalize well and too much capacity increases the risk of over-fitting. To avoid over-fitting regularization is typically applied which adds a penalty to the loss function which prevents a priori unlikely values of θ e.g. large weights. To test whether the model is overfitting to the training data the performance is often tested using a separate validation set.

In this project the data is generated by a simulation corresponding to an infinitely large set of available training data $\mathcal{D}$. For each iteration of the SGD process a new batch is generated by the simulation which means that no training data is used more than once. This reduces overfitting and eliminates the need for separate training and validation datasets.

3.2 ESTIMATION PROCEDURE

The goal of the estimator is to use the discrete signal $y[n]$ to estimate the activity indicator a, ToA τ, CFO Δf, and channel coefficient h of each user. Since the activity indicator of each user is a random variable, the total number of active users in the received signal is unknown. This boils down to a notoriously challenging problem of source separation with unknown number of users [21]. Deep learning has significantly improved the field of source separation and the general idea of using deep learning is to capture non-linear relationship between inputs and corresponding targets that is often difficult to model with analytically tractable expressions [21]. In this project, estimating the unknown parameters is dealt with by splitting the problem into:

- Classification of the number of active users; and
- Estimation of ToAs, CFOs and channel coefficients given the number of users.

The two separate tasks are combined such that the synchronization parameters are accurately estimated for each detected user.

3.3 ESTIMATION OF THE NUMBER OF USERS

Finding the number of active users, N_a, is formulated as a classification problem where $\mathbf{p} = \mathsf{OneHot}(N_a)$ is a categorical random variable encoded as a one-hot vector specifying N_a. With a one-hot encoding, the true target $\mathbf{p} = [p_0, p_1, \dots, p_K]$ has entry one at index N_a, and zero entries everywhere else. This is different from a typical way of representing active users where users are ordered in a vector and each index indicates the activity of a unique user. The number of users N_a can then be estimated as the l_0 norm of that sparse vector. In this collision scenario users are transmitting using the same spreading sequence and are not uniquely distinguishable. For this reason, only the information on the number of active users is represented in $\mathbf{p}$.

Cross-entropy loss is typically used in classification problems [22], and [23] suggests that the cross-entropy loss in classification problems leads to faster convergence and better generalization compared to the Mean Squared Error (MSE). For nonbinary classification, we typically use softmax cross entropy loss (or negative log-likelihood) expressed as:

$$\ell_{\text{NLL}}(\mathbf{p}, \mathbf{q}) = -\sum_{\text{k}=0}^{K} p_k \ln q_k, \tag{3.4}$$

where $\mathbf{q}$ is a continuous differentiable softmax function:

$$q_k = \frac{\exp(\pi_k)}{\sum_i \exp(\pi_i)}, \tag{3.5}$$

where $[\pi_0, \pi_1, \dots \pi_K]$ are the outputs from the last layer of the neural network and $[q_0, q_1, \dots q_K]$ represent the *a posteriori* class probabilities. A hard class prediction could then be found as $\arg\max_i[\pi_i]$.

The simple $\arg\max$ is not differentiable, and thus the softmax approximation of argmax is used instead [24]. The softmax function is commonly used together with the negative log-likelihood loss function where it is equivalent to maximum likelihood estimation and the ln in the loss-function keeps the exponential in the softmax function from saturating the output [25].

The loss function for estimating the number of users is written:

$$\ell_p(k, \boldsymbol{\pi}) = -\ln\left(\frac{\exp(\pi_k)}{\sum_i \exp(\pi_i)}\right). \tag{3.6}$$

and the objective of the training procedure is to minimize ℓ_p.

3.4 PARAMETER ESTIMATION

The parameters to be estimated are collected in a vector

$$\mathbf{x}_k = \Big[\tau_k, \Delta f_k, \Re[h_k], \Im[h_k]\Big]^T. \tag{3.7}$$

Note that it was found that representing the complex-valued channel coefficient h by Cartesian coordinate (i.e., real and imaginary parts) shows superior performance to phasor representation (i.e, amplitude and phase) as seen in Figure 4.5. For K users, the respective vectors are collected in a matrix

$$\mathbf{X} = [\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_{\text{K}-1}]. \tag{3.8}$$

The neural network seeks to find an estimate $\hat{\mathbf{X}}$ such that $\mathbb{E}\|\mathbf{X}-\hat{\mathbf{X}}\|_2^2$ is minimal which is equivalent to a Minimum Mean-Square Error (MMSE) estimator.

The above formulation is sufficient to derive an estimation procedure. However, $\mathbf{X}$ consists of multiple parameters which have values on different scales. When using a practical optimization algorithm to find an estimate, any scaling difference between the parameters will affect the impact each value has on the gradient descent step.

To circumvent possible issues arising from error variations across parameters, we minimize the reconstruction error instead. The actual received signal without additive noise, $\mathbf{s}$, with the parameters in matrix $\mathbf{X}$ can be reconstructed using Equation 2.2. The reconstruction is conveniently represented using function $f(\cdot)$ such that

$$\mathbf{s} = f(\mathbf{X}). \tag{3.9}$$

For each estimate $\hat{\mathbf{X}}$, the equivalent noise-free signal $\hat{\mathbf{s}}$ is reconstructed and compared to the actual noise-free received signal $\mathbf{s}$. The noise-free signal is known during the training procedure and is used so the output of the neural network does not account for the distribution of the noise. The data fidelity (i.e., reconstruction loss) is quantified using the MSE metric such that

$$\ell_r(\mathbf{X},\hat{\mathbf{X}}) = \mathbb{E}\left\|f(\mathbf{X}) - f(\hat{\mathbf{X}})\right\|_2^2 = \mathbb{E}\left\|\mathbf{s}-\hat{\mathbf{s}}\right\|_2^2. \tag{3.10}$$

The number of concurrent users in each sample is known during training so when reconstructing the signal $\hat{\mathbf{s}}$, the contributions from the correct number of users are taken into account when calculating the reconstruction loss ℓ_r for each sample.

The loss function which the neural network seeks to minimize is simply the sum of Equation 3.4 and Equation 3.10

$$\mathsf{loss} = \ell_p(k,\mathbf{q}) + \ell_r(\mathbf{X},\hat{\mathbf{X}}). \tag{3.11}$$

3.5 NETWORK IMPLEMENTATION

An overview of the neural network that estimates both the number of users and synchronization parameters is illustrated in Figure 3.1.

The output of the network is the flattened matrix $\mathbf{X}$ and the probability vector $\boldsymbol{\pi}$. For 4 users there are $4 \cdot 4 = 16$ parameters in $\mathbf{X}$ and 5 possible classes in the number of users (including the zero users case). The input to the network is processed so as to extract common features that are subsequently used for multi-task learning, that is, to detect the number of users and estimate their parameters. The first layer performs a 1-dimensional convolution over the input signal. Since the number of users, ToA, CFO and channel coefficient all are assumed to be constant throughout a transmission, a convolution layer is chosen so as to extract translationally invariant features of the input time-domain signal.

Following a typical CNN structure, batch normalization, non-linear activation and max-pooling are employed. The convolution layers, activations and pooling layers are repeated to form a deep neural network. The features found by the convolution layers are reshaped to a single vector which is then used as input to two individual feedforward neural networks. One of the networks performs classification and detects the number of users based on the output of the feature extraction layers. The other network performs regression with the goal to yield parameters so that the reconstructed signal is as close

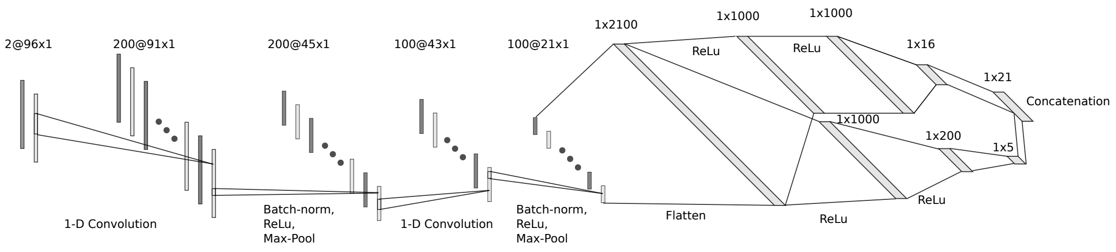


Figure 3.1: Overview of DNN architecture for detecting and estimating synchronization parameters of up to 4 colliding users.

as possible to the received signal in the MSE sense. Each feedforward network has two fully connected layers followed by the ReLU activation and a linear output layer. The network and automatic differentiation are implemented using the PyTorch framework [26] and trained using multiple Graphics Processing Units (GPUs).

Another similar network architecture is in Figure 3.2 where each network can either be trained individually or jointly. By comparing the performance of different network architectures the network in Figure 3.1 was eventually decided as performing adequately.

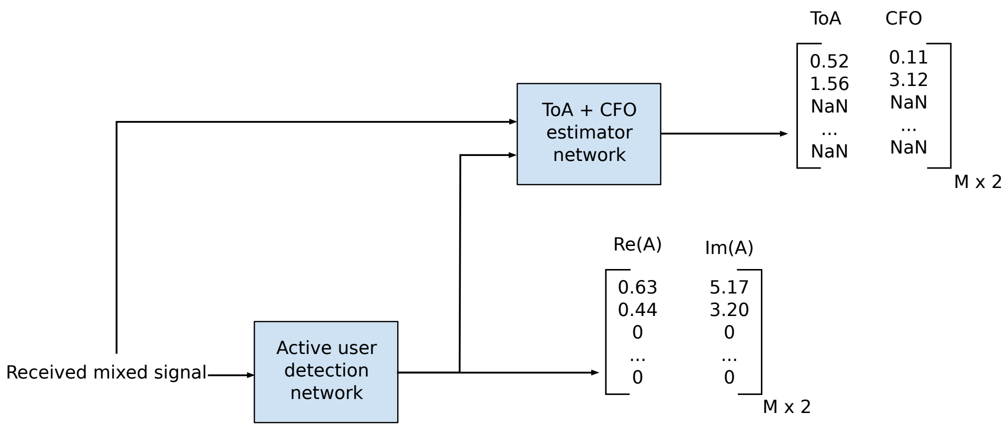


Figure 3.2: Block diagram of network architecture for joint activity detection, ToA and CFO estimation.

3.5.1 SCALING

In general the convergence of a neural network is faster if all inputs to all layers have zero-mean and unit covariance between training examples in the case when all examples are of equal importance [27]. Weights of a network $\mathbf{w}$ are updated according to the error δ by: $\mathbf{w} + \delta\mathbf{x}$. If $\mathbf{x}$ has non-zero mean the updates on all the entries in $\mathbf{w}$ will be biased in a particular direction resulting in inefficient training. The unit covariance ensures that the learning rate stays consistent between each example and that all input examples are made equally important [27].

The input to the network $\mathbf{y}$ and each parameter in the output $\mathbf{X}$ is scaled to have zero mean and unit variance. In the simulation ToA, CFO and channel coefficient are all drawn according to the distributions given in the system model and N_a is drawn according to $\Pr(k)$ for each sample as described in section 2.2. The variance and mean of each parameter (CFO, ToA and h) are known in advance and are used to normalize the parameters to have mean zero and unit variance. The standardized ToA is given by

$$\tau' = \frac{\tau - \mathbb{E}[\tau]}{\mathsf{Var}(\tau)}. \tag{3.12}$$

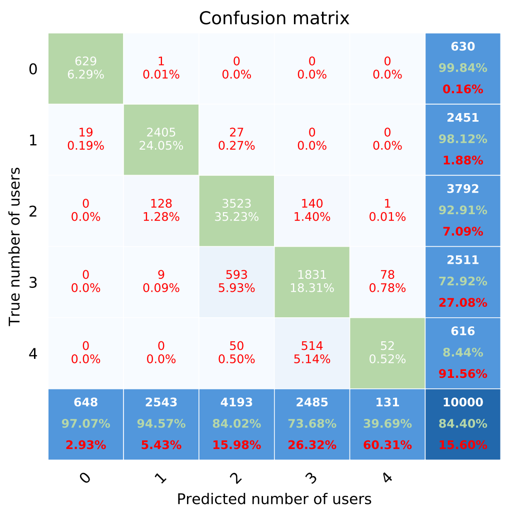
The CFO, Δf, is scaled similarly. No normalization is necessary for the channel coefficients since $h \sim \mathcal{CN}(0, 1)$ and thus no scaling is necessary for the signal $\mathbf{y}$.

4 RESULTS

The neural network is trained using samples generated with up to $K = 4$ concurrent users and at an SNR of 10 dB. New batches are generated for every step in the training procedure, and for this reason the over-fitting is suppressed as no training instance is ever used twice. The learning rate is found using empirical trials to 0.0001 which provides steady convergence of the loss function. Each batch consists of 50,000 realizations of $\mathbf{y}$ from Equation 2.3. The batch size is chosen as large as possible using the available Video Random Access Memory (VRAM) on the GPUs. Intuitively the size of each batch is linked to the accuracy of each gradient descent step and is therefore chosen as large as possible.

The stochastic optimization method based on adaptive momentum (ADAM) [28] is used due its effectiveness and popularity in recent deep learning research. A total of 20,000 different batches are used in training. In general the number of batches used are increased until the loss converges or sufficient results are achieved.

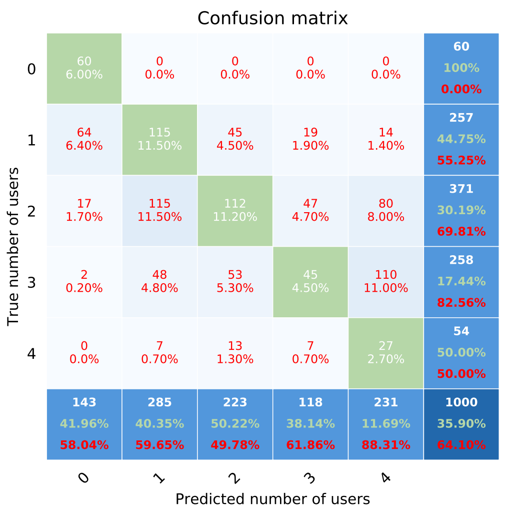
In Figures 4.1 and 4.2, the estimation of collision multiplicity is shown for the proposed classification method compared to the simple amplitude-based method described in section 2.4.1. As colliding signals will add non-coherently, the amplitude of the signal is not a good indicator on collision multiplicity. Using the proposed estimator 1 and 2 users are successfully identified with 98.1 % and 92.9 % at an SNR of 10 dB and the estimation accuracy decreases with the number of concurrent users. The amplitude-based method successfully identified 1 and 2 users with 44.8 % and 30.2 % accuracy but is better at estimating 0 and 4 users. The proposed method often miss-classifies a signal containing 4 colliding users as resulting from transmissions of 3 users. Further, the classification accuracy depends on the SNR used during training. Training at testing using the same SNR provides better results.



Confusion matrix

True number of users \ Predicted number of users	0	1	2	3	4	
0	629 6.29%	1 0.01%	0 0.0%	0 0.0%	0 0.0%	630 99.84% 0.16%
1	19 0.19%	2405 24.05%	27 0.27%	0 0.0%	0 0.0%	2451 98.12% 1.88%
2	0 0.0%	128 1.28%	3523 35.23%	140 1.40%	1 0.01%	3792 92.91% 7.09%
3	0 0.0%	9 0.09%	593 5.93%	1831 18.31%	78 0.78%	2511 72.92% 27.08%
4	0 0.0%	0 0.0%	50 0.50%	514 5.14%	52 0.52%	616 8.44% 91.56%
	648 97.07% 2.93%	2543 94.57% 5.43%	4193 84.02% 15.98%	2485 73.68% 26.32%	131 39.69% 60.31%	10000 84.40% 15.60%

(a) Proposed method: 84.4% total accuracy at 10 dB SNR.



Confusion matrix

True number of users \ Predicted number of users	0	1	2	3	4	
0	60 6.00%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	60 100% 0.00%
1	64 6.40%	115 11.50%	45 4.50%	19 1.90%	14 1.40%	257 44.75% 55.25%
2	17 1.70%	115 11.50%	112 11.20%	47 4.70%	80 8.00%	371 30.19% 69.81%
3	2 0.20%	48 4.80%	53 5.30%	45 4.50%	110 11.00%	258 17.44% 82.56%
4	0 0.0%	7 0.70%	13 1.30%	7 0.70%	27 2.70%	54 50.00% 50.00%
	143 41.96% 58.04%	285 40.35% 59.65%	223 50.22% 49.78%	118 38.14% 61.86%	231 11.69% 88.31%	1000 35.90% 64.10%

(b) Conventional power-detection method: 35.9% total accuracy at 10 dB SNR.

Figure 4.1: Confusion matrices for estimating the multiplicity of collisions comparing the neural network classifier with a simple classifier based on the amplitude of the received signal.

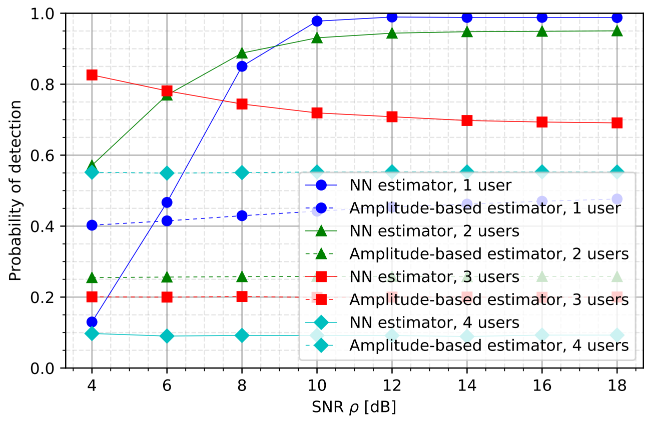


Figure 4.2: Accuracy of estimating the number of colliding users. A signal is deemed correctly detected if the number of users are estimated correctly. The NN estimator is trained for signals with 10 dB SNR.

The accuracy of the parameter estimation $\hat{\mathbf{X}}$ is found by comparing with the target $\mathbf{X}$. Since the loss function defined in Equation 3.10 only depends on the reconstruction error, the estimated parameters in $\hat{\mathbf{X}}$ are arbitrarily ordered across users. To compare the output with the target $\mathbf{X}$ the parameters are chosen to be ordered according to the estimated amplitudes. In cases where the estimated amplitudes are similar, the ordering may be wrong which leads to an artificially high error when evaluating performance for multiple users. Any of the parameters can be used to dictate the ordering but the amplitude is chosen as it is linked to SNR at the receiver in practice.

The Root-Mean-Square Error (RMSE) of each parameter in $\hat{\mathbf{X}}$ is calculated as:

$$\mathsf{RMSE}_k = \sqrt{\mathbb{E}\left[\|e_k\|_2^2\right]}, \tag{4.1}$$

where e.g. the estimation error of τ is: $e_k = \tau_k - \hat{\tau}_k$. The RMSE of the proposed neural network-based estimator is the average of all RMSEs up to user k:

$$\mathsf{RMSE}_{\mathrm{NN,k}} = \frac{1}{k}\sum_{\mathrm{i}=1}^{k} \mathsf{RMSE}_i. \tag{4.2}$$

The performance of the neural network estimator is to be compared with the conventional estimator described in section 2.4. The conventional estimator is only able to estimate a single set of parameters, regardless of the actual number of users k. The error of the conventional estimator is therefore measured as the estimate which has the smallest error over all actual sets of parameters in $\mathbf{X}$, e.g. the estimated ToA error is

$$e_{\tau,\mathrm{PD}} = \min_k(|\tau_k - \hat{\tau}_{\mathrm{PD}}|). \tag{4.3}$$

This gives the conventional estimator an artificial advantage.

The RMSE of ToA and CFO estimation with a varying number of users are shown in Figures 4.3 and 4.4. The neural network-based estimator shows lower estimation error for both ToA and CFO compared to the PD-based estimator even for a single user. For two users the proposed estimator is superior to the conventional estimator when estimating ToA. At 10 dB the proposed estimator has an RMSE of 2.88 µs and

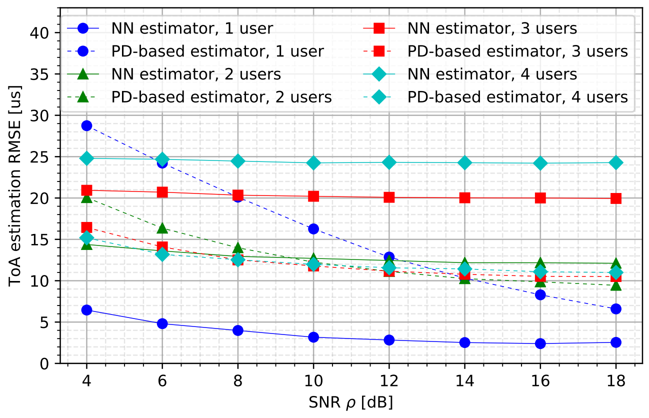


Figure 4.3: RMSE of ToA estimation across SNRs.

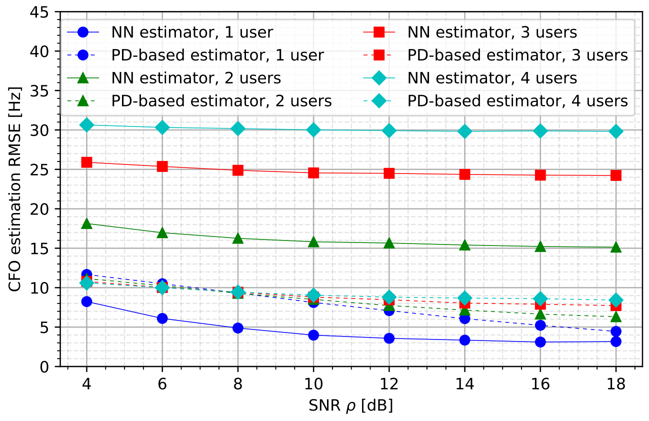


Figure 4.4: RMSE of CFOs estimation across SNRs.

3.44 Hz for a single user compared to 16.20 µs and 7.98 Hz for the conventional estimator. The relatively high RMSE of the conventional estimator is likely due to the noise which causes wrong phase unwrapping at low SNRs [9].

To explore the accuracy of the conventional and the proposed estimator the distribution of the errors are plotted in Figures 4.6a and 4.6b. The model is trained with the number of users varying from 0-4 but for exploring the results Cumulative Distribution Functions (CDFs) are shown for signals with each number of collisions individually. It is seen that there is a clear advantage using the NN estimator for 1 user but the performance advantage is not convincing for multiple users. This is believed attributed to the unfair advantage of the conventional estimator where estimates are chosen as the closest matching value across all different users.

ToA and CFO both show similar distributions and it better performance can be achieved by including more capacity in the network and fine-tuning of the hyper-parameters.

The convergence of the loss function during training of a network trained for a single user transmission is seen in Figure 4.7 along with the CDF of estimation error. This shows the excellent performance that can be achieved even at high SNR.

The accuracy in estimating the channel coefficient h is shown in Figure 4.5. The RMSE is 0.101 for the in-phase part and 0.103 for the quadrature part for a single user.

The RMSE shows a similar trend as in ToA and CFO estimation with deteriorating performance as the number of concurrent users increases.

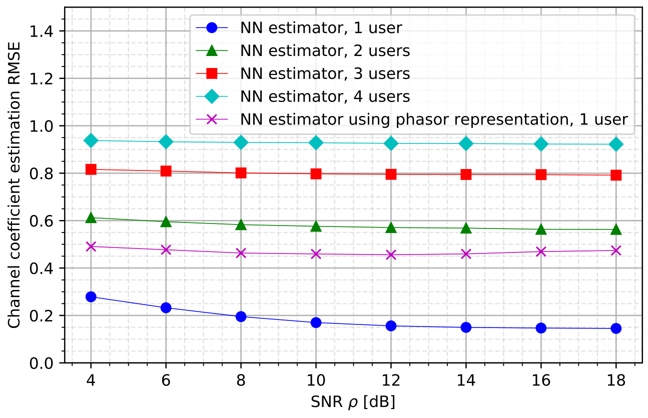
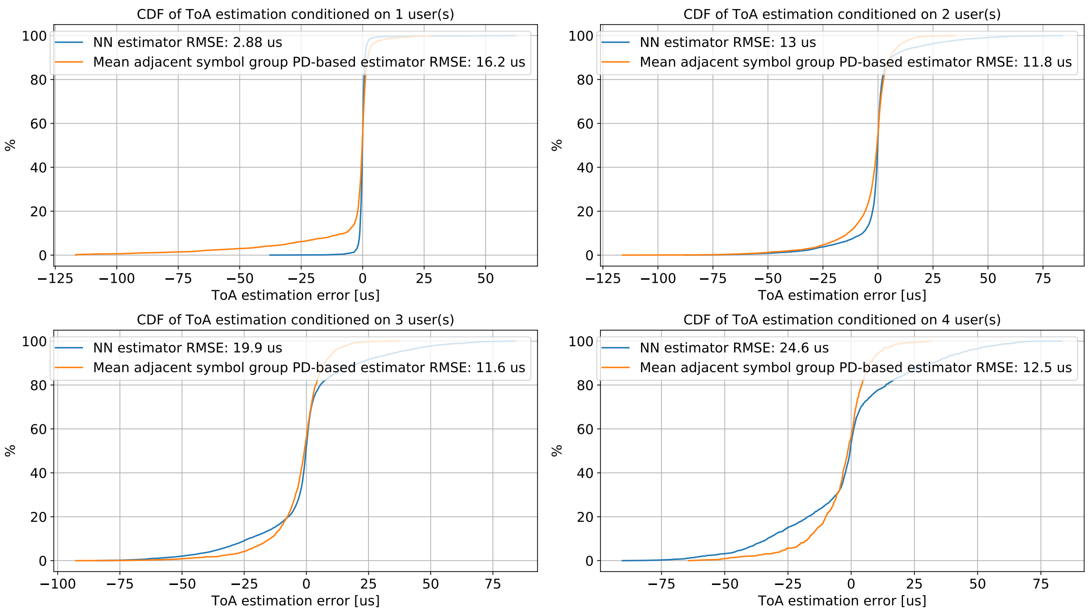
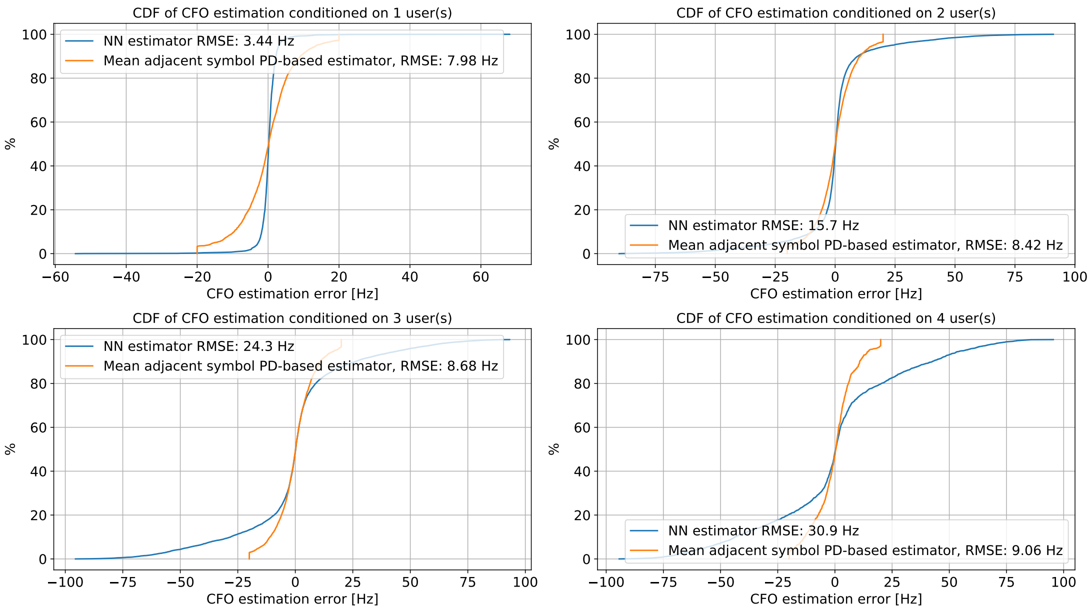


Figure 4.5: RMSE of channel coefficient estimation across SNRs.

Overall the proposed method presents considerably improved performance compared to the traditional estimator in scenarios with a single, as well as multiple users. Despite the success of the numerical results, they are far from the theoretical achievable performance bound as derived in Appendix A implying that there are still opportunities for further enhancements.



(a) CDFs for ToA estimation conditioned on varying number of collisions.



(b) CDFs for CFO estimation conditioned on varying number of collisions.

Figure 4.6: Distribution of estimation errors for the conventional and proposed estimator.

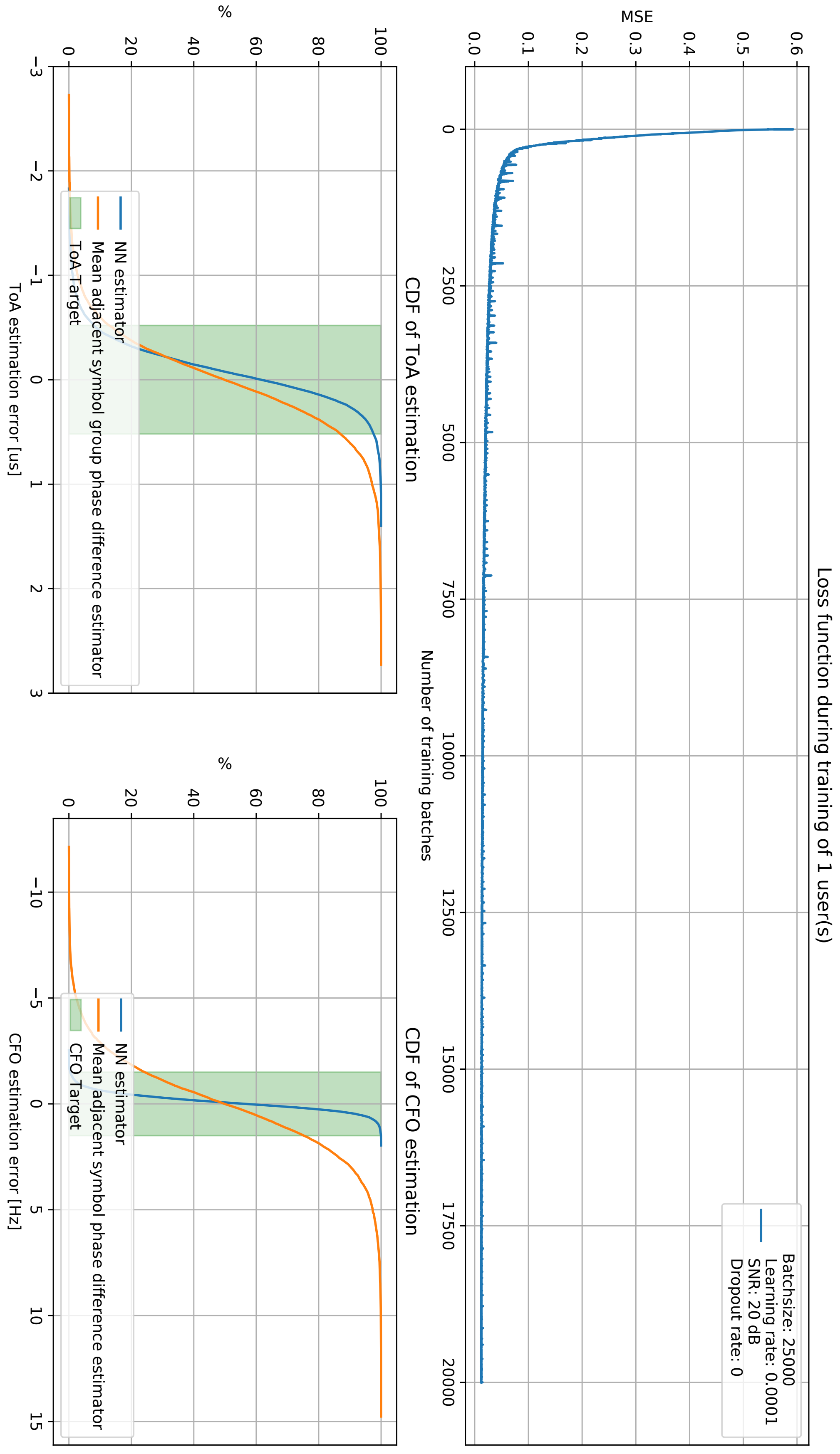


Figure 4.7: Loss function and CDF of ToA and CFO estimations using the trained deep learning estimator.

5 AUTO-ENCODER

Current communication systems are optimized block by block for performance in relation to a model [29]. A novel concept is to consider the communication system in an end-to-end manner and jointly optimize both the blocks within the transmitter and receiver as well as optimizing the transmitter and receiver structure jointly. In [29] an end-to-end auto-encoder is proposed to derive modulation and coding schemes which shows performance comparative to state-of-the art communication methods. The channel is treated as a layer in a the neural network and encoder and decoder are both neural networks. Applications of an auto-encoder applied to the physical layer of a wireless communication systems shows great promise [30] and for this reason, are considered applied to preamble encoding and decoding.

The general auto-encoder attempts to reconstruct the given input based on a latent representation called $\mathbf{h}$. Traditionally it is designed for dimensionality reduction and feature learning where $\mathbf{h}$ is a much lower dimensionality space. In its simplest form the encoder $\mathbf{h} = f(\mathbf{x})$ produces the latent representation and the decoder reconstructs the input from the latent representation $\hat{\mathbf{x}} = g(f(\mathbf{x}))$. In the noise-free scenario $\hat{\mathbf{x}} = \mathbf{x}$ [19].

A modern application for auto-encoders is the Denoising Auto-Encoder (DAE) where the decoder predicts the original input based on a corrupted sample [19]. In this project a modified DAE is used to find a preamble sequence and decoding method jointly. The computational graph for the auto-encoder structure is illustrated in Figure 5.1.

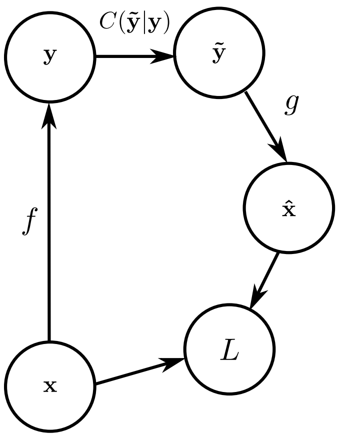

$$\begin{tikzcd}
\mathbf{y} \arrow[r, "C(\tilde{\mathbf{y}}|\mathbf{y})"] & \tilde{\mathbf{y}} \arrow[d, "g"] \\
 & \hat{\mathbf{x}} \arrow[d] \\
\mathbf{x} \arrow[uu, "f"] \arrow[r] & L
\end{tikzcd}$$

Figure 5.1: Structure of auto-encoder mapping the input $\mathbf{x}$ to latent variable $\mathbf{y}$ using the function f (encoder). The corruption process is the simple Additive White Gaussian Noise (AWGN) channel producing $\mathbf{y} + \mathbf{w}$. The function g (decoder) takes the corrupted latent variable to create a reconstruction of the input $\hat{\mathbf{x}}$. The learning process attempts to minimize the scalar reconstruction error described by the loss L.

The DAE maps the input:

$$\mathbf{x}_k = \Big[\tau_k, \Delta f_k, \Re[h_k], \Im[h_k]\Big]^T \tag{5.1}$$

to a transmitted preamble sequence $\mathbf{y} = f(\mathbf{x})$ where $f(\cdot)$ is a parametric function of frequency hopping pattern and symbol sequence. The typical DAE has the corruption process applied to the input space $\mathbf{x}$ but in this context the corruption process applied to the latent representation. Only the simple AWGN channel is considered where the corruption process is $C(\tilde{\mathbf{y}}|\mathbf{y}) = \mathbf{y} + \mathbf{w}$ where $\mathbf{w} \sim \mathcal{CN}(0, 1/\rho)$.

The decoder is a deep neural network which takes the corrupted latent representation $\tilde{\mathbf{y}}$ as input to reconstruct the input $\mathbf{x}$ with minimum error. The reconstruction loss is calculated by some function $L(\cdot)$, such as the l_2 norm.

$$L(\mathbf{x}, g(f(\mathbf{x}) + \mathbf{w})) = L(\mathbf{x}, g(\tilde{\mathbf{y}})) = L(\mathbf{x}, \hat{\mathbf{x}}) \tag{5.2}$$

The training process attempts to minimize this loss by updating the weights in $f(\cdot)$ and $g(\cdot)$.

5.1 ENCODING

The preamble sequence in NB-IoT is a deterministic channel hopping pattern predetermined by the initial sub-carrier and the QPSK-symbol $1 + 0\jmath$ is continuously repeated. The encoding seeks to find a different channel hopping pattern and symbol sequence which enables the decoder to provide a more accurate estimate of ToA, CFO and channel coefficient.

The function $f(\cdot)$ is used similarly as in Equation 3.9 where the transmitted sequence is:

$$\mathbf{y} = f(\mathbf{x}, \mathbf{f}, \mathbf{s}) = h\,\mathrm{e}^{-\jmath 2\pi(\mathbf{f}+\Delta f)(nT_{\mathrm{sym}}-\tau)} \odot \mathbf{s} \tag{5.3}$$

Where:	$\mathbf{f}$ is a vector containing frequency hopping pattern	[Hz]
	$\mathbf{s}$ is a vector containing symbol sequence	[·]
	$\odot$ is component-wise multiplication	[·]

The frequency pattern $\mathbf{f}$ and symbol sequence $\mathbf{s}$ are fixed sequences that are valid for all $\mathbf{x}$ and are only adjusted during the training procedure to "discover" a superior preamble sequence. Once a well-suited preamble sequence is found, the preamble sequence should remain fixed for all future transmissions. $\mathbf{f}$ and $\mathbf{s}$ are implemented as learnable parameters that are updated according to their gradients with a learning rate of $\gamma = 1 \times 10^{-4}$.

5.2 DECODING

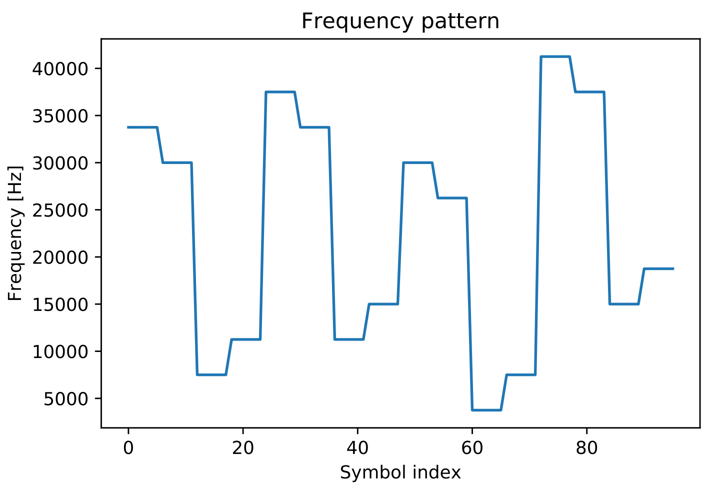
Decoding refers to detection and synchronization parameter estimation. In practice decoding happens at the BS where it is unknown whether a transmission has occurred or not. In this set-up the encoder and decoder are a joint operation and therefore coordinated such that the decoding each transmission is always decoded.

The corrupted received signal $\tilde{\mathbf{y}}$ is the input to the decoder function $g(\cdot)$

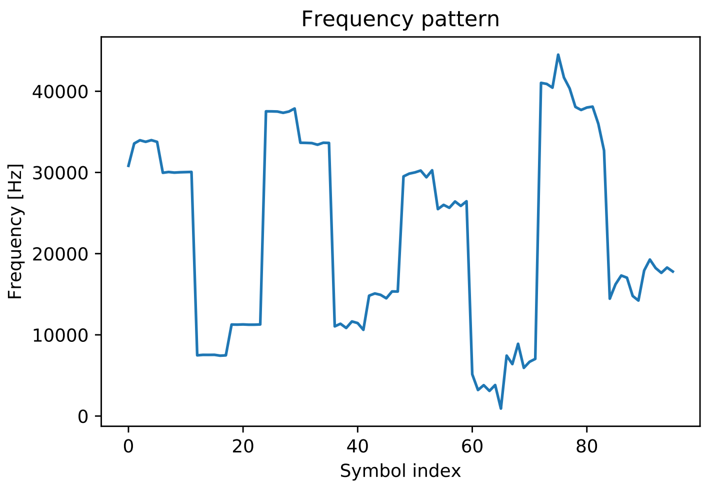
$$g(\tilde{\mathbf{y}}) = \hat{\mathbf{x}}. \tag{5.4}$$

The decoder is a neural network and the same pre-trained network from section 3.5 can be used as initialization. However, for the sake of simplicity to demonstrate the concept a simple feedforward neural network is used and the input signal consists only of 1 user. Figure 5.2 shows the development of the frequency pattern which is found throughout the training process. In Figure 5.2a an example of the NB-IoT NPRACH frequency pattern is illustrated and the encoding function is initialized with this pattern. In Figure 5.2b is the frequency pattern found after 100 iterations. It is seen that the frequency pattern does not converge towards an orderly sequence and is not restricted to the 12 frequency channels defined by the NB-IoT standard. This is a limitation in the implementation which requires learnable parameters to be differentiable. In this implementation the parameters are continuous and further development should address how to make frequency and symbol pattern adhere to the pre-defined discrete subcarriers and symbols.

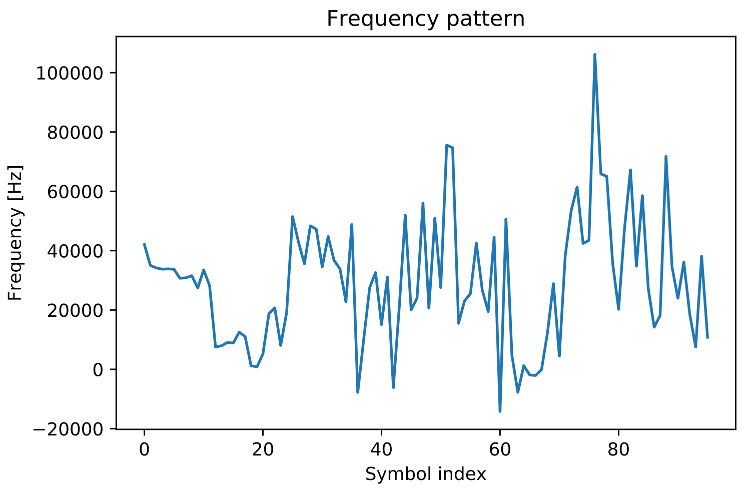
Note that the current NPRACH sequence is designed to minimize the Peak-to-Average Power Ratio (PAPR) compared to Zadoff–Chu sequence used in LTE. When tweaking the preamble sequence the NPRACH PAPR will become worse which is a trade-off between performance and power. When using the auto-encoder the best estimation performance is 4.38 µs and 6.00 Hz at 10 dB after 100 000 iterations. This is not improved compared to the NN estimator using the NPRACH pattern which is the performance goal. Further development is necessary since it is believed the method still has basis for achieving superior performance.



(a) One example of NB-IoT frequency pattern



(b) Learned frequency pattern using auto-encoder initialized with NB-IoT frequency hopping pattern



(c) Learned frequency pattern using auto-encoder after 100 000 training iterations

Figure 5.2: Development of frequency pattern throughout auto-encoder training

6 DISCUSSION AND CONCLUSION

6.1 FUTURE IMPROVEMENTS

In an actual implementation much attention should be made to ensure that the data used to train the model is representative. The performance guarantees that can be provided using machine learning are only numerical using the available data. Either real-world data or realistic models of dynamics should be included in the model.

The deep neural network model should be able to estimate ToA and CFO in real time. Using a Software-Defined Radio (SDR) wideband signals can be captured and processed in real-time. AIR-T is an SDR specifically designed for deep learning deployment that combines an AD9371 transceiver with an FPGA for signal processing and a GPU for deep learning [31].

6.2 THE PERFORMANCE OF DL IN NB-IOT

The goal of this project is to find a method to increase system capacity. The increase in number of supported users is therefore investigated. The number of concurrent users each cell supports will be limited by the number of allocated channels and user traffic pattern. In NB-IoT there are 48 available channels for NPRACH for each cell. This gives an average traffic intensity of 13 erlangs according to the Erlang B loss system at a required probability of blocking less than 1% [32].

It is assumed that each users has an average holding time of 50 ms corresponding to the periodicity of the normal NPRACH transmission. The traffic pattern is assumed to consist of independent users transmitting once every 10th second which gives a channel usage per user:

$$360\,\text{Transmission/h} \cdot 50\,\text{ms} = 5.0 \times 10^{-3}\,\text{erlangs} \tag{6.1}$$

Since the system can support 13 erlangs in total the maximum number of users are:

$$\frac{13\,\text{erlangs}}{5.0 \times 10^{-3}\,\text{erlangs}} = 2600\,\text{users} \tag{6.2}$$

Using a simple model for user activity and only accounting for the normal NPRACH configuration a total of 2600 users can be supported simultaneously if NPRACH transmission are rejected in the case of collisions.

When using the proposed estimation procedure not only will synchronization be more accurate but simultaneous preamble transmissions can be detected which improves the access probability. 2 user transmissions are detected correctly 93 % of the time which effectively increases the available number of random access preambles by 93 %. This increase of successfully transmitted preambles, can be used to allocate fewer NPRACH resources, reduce access delay since collision induced back-offs will be reduced or increase the number of supported users to 5018. However, this project does not consider limitations of NPUSCH resources which may prove to be a bottleneck for the RA process.

The intuition behind being able to detect and decode identical simultaneous preamble transmissions is to exploit diversity in user locations, specific channel conditions and oscillator imperfections to realize multiplexing.

When the base station detects multiple preambles, users are not separable by any ID and the RAR (MSG2) cannot be specifically addressed to each user. A procedure to associate each response to each user will be to have individual users estimate distance from the BS using the reference signal. The estimated distance is used by the user to approximate a scope of TA which it can use to select the RAR with the closest matching TA.

6.3 CONCLUSION

This project considers the problem of separating colliding NB-IoT users that choose the same random access preamble in the NPRACH scheme, and propose a method to detect the number of colliding users and estimate their ToA, CFO and channel gains. Motivated by recent success in leveraging learning-based methods for addressing problems related to physical layer communications [12], the proposed method builds upon deep learning framework. In particular, the method jointly detects the number of active users and estimates their parameters, with the aim to improve the capacity of the critical random access phase by not discarding interfering signals in order to utilize channel resources better, which in turn reduces back-off periods. In addition to handling much richer class of scenarios, the proposed method outperforms [9] in their own scenario where users transmit orthogonal preambles and do not collide.

The method is demonstrated in NB-IoT NPRACH where the number of orthogonal preambles is limited ensuring the proposed method is practical in IoT systems currently being deployed. The estimation error of a conventional approach in NB-IoT is compared to the performance of the proposed scheme. Traditional synchronization methods fail in the case of collisions with high Signal-to-Interference Ratio (SIR) whereas, with the proposed algorithm users can be distinguished and respective synchronization parameters can still be estimated with a reasonable performance.

Deep learning is a promising tool for developing joint estimation procedures, which are notoriously difficult in traditional model-based methods, and enables separation of synchronization parameters even when users transmit using the same preamble. A deep learning building block the denoising auto-encoder, is applied in a novel concept to discover an alternative superior preamble sequence. The found preamble sequence reflects the distribution of the input data but further works is required to to achieve an increase in performance compared to the neural network estimator. Although deep learning-based estimation will lead to sub-optimal estimators compared to an analytically derived joint estimator, it allows for practical, straightforward development and efficient computation.

BIBLIOGRAPHY

[1] P. Popovski, K. F. Trillingsgaard, O. Simeone, and G. Durisi, “5g wireless network slicing for embb, urllc, and mmtc: A communication-theoretic view,” *IEEE Access*, vol. 6, pp. 55 765–55 779, 2018.

[2] M. Centenaro, L. Vangelista, A. Zanella, and M. Zorzi, “Long-range communications in unlicensed bands: the rising stars in the iot and smart city scenarios,” *IEEE Wireless Communications*, vol. 23, no. 5, pp. 60–67, October 2016.

[3] A. Biral, M. Centenaro, A. Zanella, L. Vangelista, and M. Zorzi, “The challenges of m2m massive access in wireless cellular networks,” *Digital Communications and Networks*, vol. 1, no. 1, pp. 1–19, 2015.

[4] I. Catherine, R. Tardy, N. Aakvaag, B. Myhre, R. Bahr, I. Catherine, R. Tardy, N. Aakvaag, B. Myhre, and R. Bahr, “Comparison of wireless techniques applied to environmental sensor monitoring,” SINTEF Digital, Trondheim Norway, Tech. Rep., 2017.

[5] A. Azari, P. Popovski, G. Miao, and C. Stefanovic, “Grant-free radio access for short-packet communications over 5G networks,” in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Dec 2017, pp. 1–7.

[6] X. Lin, A. Adhikary, and Y. P. Eric Wang, “Random access preamble design and detection for 3GPP narrowband IoT systems,” *IEEE Wireless Communications Letters*, vol. 5, no. 6, pp. 640–643, 2016.

[7] L. Feltrin, G. Tsoukaneri, M. Condoluci, C. Buratti, T. Mahmoodi, M. Dohler, and R. Verdone, “Narrowband IoT: A survey on downlink and uplink perspectives,” *IEEE Wireless Communications*, vol. 26, no. 1, pp. 78–86, February 2019.

[8] 3GPP TS, *36.321 - Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol specification, V. 15.4.0*, 2018.

[9] J. Hwang, C. Li, and C. Ma, “Efficient detection and synchronization of superimposed NB-IoT NPRACH preambles,” *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 1173–1182, Feb 2019.

[10] A. Molisch, *Wireless Communications*, ser. Wiley - IEEE. Wiley, 2010.

[11] A. Goldsmith and K. (Firm), *Wireless Communications*. Cambridge University Press, 2005.

[12] O. Simeone, “A Very Brief Introduction to Machine Learning With Applications to Communication Systems,” *arXiv e-prints*, p. arXiv:1808.02342, Aug 2018.

[13] W. S. Jeon, S. B. Seo, and D. G. Jeong, “Effective frequency hopping pattern for ToA Estimation in NB-IoT Random Access,” *IEEE Transactions on Vehicular Technology*, vol. 67, no. 10, pp. 10 150–10 154, 2018.

[14] Y. . E. Wang, X. Lin, A. Adhikary, A. Grovlen, Y. Sui, Y. Blankenship, J. Bergman, and H. S. Razaghi, “A primer on 3GPP narrowband Internet of things,” *IEEE Communications Magazine*, vol. 55, no. 3, pp. 117–123, March 2017.

[15] 3GPP TS, *36.211 - Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulation, V. 15.4.0*, 2018.

[16] T. Hien, Z. Wang, S. Kim, J. Nielsen, and P. Popovski, "Preamble detection in NB-IoT random access with limited-capacity backhaul," in *Proceedings of IEEE ICC'19*, 2 2019.

[17] O. Liberg, M. Sundberg, E. Wang, J. Bergman, and J. Sachs, *Cellular Internet of Things: Technologies, Standards, and Performance.* Elsevier Science, 2017.

[18] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 36.213, 03 2019, version 15.4.0.

[19] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016.

[20] T. Diamandis, "Survey on Deep Learning Techniques for Wireless Communications," *Stanford.Edu*, vol. 521, no. July, pp. 3–4, 2017.

[21] J. Chien, *Source Separation and Machine Learning.* Elsevier Science, 2018.

[22] C. Bishop, *Pattern Recognition and Machine Learning*, ser. Information Science and Statistics. Springer New York, 2016.

[23] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, Aug 2003, pp. 958–963.

[24] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with Gumbel-softmax," *arXiv e-prints*, p. arXiv:1611.01144, Nov 2016.

[25] J. Jagannath, N. Polosky, A. Jagannath, F. Restuccia, and T. Melodia, "Machine Learning for Wireless Communications in the Internet of Things: A Comprehensive Survey," *arXiv e-prints*, p. arXiv:1901.07947, Jan 2019.

[26] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS-W*, 2017.

[27] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural Networks: Tricks of the Trade.* Springer-Verlag, 1998.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv e-prints*, p. arXiv:1412.6980, Dec 2014.

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

[30] V. Raj and S. Kalyani, "Backpropagating through the air: Deep learning at physical layer without channel models," *IEEE Communications Letters*, 2018.

[31] I. Deepwave Digital, “Deepwave digital,” Website, Mar. 2019. [Online]. Available: https://www.deepwavedigital.com/sdr

[32] R. Freeman, *Fundamentals of Telecommunications*, ser. Wiley Series in Telecommunications and Signal Processing. Wiley, 2005.

[33] A. van den Bos, *Parameter Estimation for Scientists and Engineers.* Wiley, 2007.

[34] A. N. D’Andrea, U. Mengali, and R. Reggiannini, “The modified cramer-rao bound and its application to synchronization problems,” *IEEE Transactions on Communications*, vol. 42, no. 234, pp. 1391–1399, February 1994.

Appendices

A) CRAMÉR–RAO LOWER BOUND

The achievable precision of an unbiased estimators may be given by the lower bound on the variance of the estimation, called the Cramér–Rao lower Bound (CRB) [33]. The CRB is used as reference to compare the performance of the traditional estimator and the deep learning estimator in chapter 3.

The observed vector $\mathbf{r}$ depends on the synchronization parameters: Δf the carrier frequency offset, τ the ToA and the channel carrier phase θ which are desired to be estimated. For practical reasons the challenging case of a joint estimation of $\tau, \Delta f, \theta$ is overlooked and the CRB can be derived for an estimator of a single parameter and treating the other parameters as unwanted parameters. In the general case a single element of $\Delta f, \tau, \theta$, is denoted λ and is assumed deterministic while other elements are random variables collected in a vector $\mathbf{u}$. The vector $\mathbf{u}$ is assumed to have a known PDF that does not depend on λ.

The CRB is formulated as:

$$CRB(\lambda) = \frac{1}{E_{\mathbf{r}}\left[\left(\frac{\delta \ln p(\mathbf{r}|\lambda)}{\delta\lambda}\right)^2\right]} \tag{A.1}$$

Equation A.1 is often difficult to evaluate and therefore a different bound, the Modified Cramér–Rao Bound (MCRB) is considered instead [34]:

$$MCRB(\lambda) = \frac{1}{E_{\mathbf{r}}\left[\left(\frac{\delta \ln p(\mathbf{r}|\mathbf{u},\lambda)}{\delta\lambda}\right)^2\right]} \tag{A.2}$$

Generally $MCRB(\lambda) \leq CRB(\lambda)$ meaning it is a more loose bound. However, for most practical applications is still useful [34].

For the Gaussian channel it is much easier to derive the conditional probability $p(\mathbf{r}|\mathbf{u}, \lambda)$ than evaluating $p(\mathbf{r}|\lambda)$.

The received complex signal waveform with additive noise can be written:

$$r(t) = s(t) + w(t) \tag{A.3}$$

where $s(t)$ is the information signal as described in Equation 2.1 and $w(t)$ is additive noise distributed according to a complex normal distribution.

$p(\mathbf{r}|\mathbf{u}, \lambda)$ is replaced by the likelihood function $\Lambda(\lambda, \mathbf{u})$ and after some manipulation Equation A.2 becomes [34]:

$$MCRB(\lambda) = \frac{N_0}{E_{\mathbf{u}}\left[\int_{\mathrm{T}_0} \left|\frac{\delta s(t)}{\delta\lambda}\right|^2 dt\right]} \tag{A.4}$$

where N_0 is the noise variance.

Going from the general case of estimating λ to the specialised case of estimating CFO, Δf, by setting $\lambda = \Delta f$ and $\mathbf{u} = \mathbf{u}_{\Delta\mathrm{f}} = \{\tau, \theta\}$. The integral in the denominator can be evaluated to:

$$\int_{\mathrm{T}_0} \left| \frac{\delta s(t)}{\delta \Delta f} \right|^2 dt = \int_{\mathrm{T}_0} \left| \frac{\delta e^{-j2\pi(f+\Delta f)(t-\tau)}}{\delta v} \right|^2 dt \tag{A.5}$$

$$= \int_{\mathrm{T}_0} 4\pi^2 \left| t-\tau \right|^2 dt \tag{A.6}$$

According to Equation A.4 the expectation should be calculated over $\mathbf{u}_{\Delta \mathrm{f}}$ that is over both θ and τ. However, Equation A.5 does not depend on θ and the limitations is only calculated over τ.

$$E_\tau \left[\int_{\mathrm{T}_0} 4\pi^2 (t-\tau)^2 \, dt \right] = 4\pi^2 E_\tau \left[\int_{\mathrm{T}_0} (t-\tau)^2 \, dt \right] \tag{A.7}$$

$$= 4\pi^2 E_\tau \left[\int_0^{NT} (t-\tau)^2 \, dt \right] \tag{A.8}$$

where NT is the length of T_0. A simple case can be assumed where $\tau \sim \mathcal{U}(a,b)$ and the expectation can be evaluated to:

$$= 4\pi^2 E_\tau \left[\frac{(NT)^3}{3} - \tau (NT)^2 + \tau^2 (NT) \right] \tag{A.9}$$

$$= \frac{2}{3}\pi^2 (2a^2(NT) + 2ab(NT) - 3a(NT)^2 + 2b^2(NT) - 3b(NT)^2 + 2(NT)^3) \tag{A.10}$$

setting limits $a = 0$ and $b = NT$ we get:

$$= \frac{2\pi^2 (NT)^3}{3} \tag{A.11}$$

Finally, to summarize we calculated the denominator of the MCRB in Equation A.4:

$$E_{\mathbf{u}_{\Delta \mathrm{f}}} \left[\int_{\mathrm{T}_0} \left| \frac{\delta s(t)}{\delta \Delta f} \right|^2 dt \right] = \frac{2\pi^2 (NT)^3}{3} \tag{A.12}$$

The SNR, E_s/N_0 is normalized according to symbol period T, and is therefore omitted. From Equation A.4 we have:

$$MCRB(\Delta f) = \frac{3N_0}{2\pi^2 N^3} \tag{A.13}$$

This states that the a lower bound on estimation error is a simple analytical expression which is a function of (normalized) SNR and number of samples. The MCRB can be similarly derived for $MCRB(\tau)$ and $MCRB(\theta)$ but the derivation is not included here. The expression is used to provide perspective on performance in Table A.1.

The derived expression uses a simple model that does not account for the channel coefficient as used in the model in Equation 2.2. The frequency is normalized to the interval $[-1/2, 1/2]$ and the expression is used to compare performance between the PD-based (section 2.4) and the NN estimator (section 3.2). The number of samples is chosen to $N = 96$ samples and the SNR is 10 dB .

Table A.1: Performance comparison with MCRB, traditional estimator and NN estimator.

MCRB	Error variance of traditional estimator	Error variance of NN estimator
1.7×10^{-8}	0.78	2.8×10^{-2}

It is seen that both estimator are far from the achievable performance bound.

Literature Study on Deep Learning for the Wireless Physical Layer

Mads Helge Jespersen

I. Overview

This is not a comprehensive review of state of the art on deep learning for the physical layer but this is the areas that deep learning has been applied to the physical layer, examples of successful implementations and ideas for research areas to focus on.

The overview paper [1] presents the progress in the field well and provides advice on when to use ML with a set of criteria. Most importantly the model at hand should be either model deficit or algorithm deficit.

a) Model deficit: No mathematical model, conventional design methods not applicable. Statistical performance guarantees cannot be provided. The algorithm can only be relied upon so far data used is trusted to be representative for the whole set of possible realizations.

b) Algorithm deficit: A mathematical model is available but algorithms are either too complex to derive or too computationally complex to be implemented. Neural networks has possibility to yield lower-complexity solutions. A computer simulation can be carried out to obtain numerical performance guarantees.

A. Possible directions

1) End-to-end Constellation Design and Detection: Motivation can be decoding in non-coherent scenario (algorithm deficit) or transmission through a non-linear medium or non-linear transceiver chain (algorithm deficit). Research has conducted to derive maximum likelihood decoders from simulation samples in non-coherent MIMO communication (Unknown CSI) and in a two-way relay network. Most simulations assume perfect frequency synchronization between transmitter and receiver but vary the channel coefficient between symbols. Commonly methods are restricted to only training for one strict channel scenario and will probably perform poorly in different channel conditions.

2) Machine Learning for Emerging Communication Technologies: The research area to focus on can be identified by finding an application for ML in one of following emerging technologies: mmWave, Massive Connectivity, Massive MIMO.

Current mmWave front end design suffers from being very expensive, bad efficiency and non-linearity [2]. Non-linearity of amplifiers could be accounted for by pre-distortion but a neural network receiver might also give better performance. Expensive receiver chain can be alleviated by using low-resolution ADCs and phase shifter (1 or 2 bit). Effect of these can be hard to account for in an algorithm and maybe a ML algorithm can perform better.

Only a few papers investigate the applicability of machine learning in massive connectivity [3], for this reason there is a high probability of publication using this is as project motivation. MERL work has focused on a CDMA decoding in presence of a structured jammer with deep learning with initial good results in cases where a matched filter method does not work [4].

There are few ML applications in the emerging LPWAN technologies such as LoRa, SigFox and NB-IoT.

3) Apply the latest development within Machine Learning to a Wireless Problem: Adversarial training is a very popular method training robust neural networks. The same concept can be used to create a robust jamming resistant communication. Adversarial neural networks could also be used for encryption and decryption in relay networks.

Two recent trends in deep learning are attention networks and Generative Adversarial Networks (GANs).

II. Channel Detection and Decoding

Design of a channel decoder based on samples is only possible if the channel is stationary over a long period of time, meaning it does not change too rapidly over time (fast fading). Could be made possible for fast fading channels of channel estimation is part of the learning process [1].

Model deficit: Molecular communication [5] Algorithm deficit: Strong non-linearities: Satellite communication, optical communications, modulation schemes such as continuous phase modulation or in multi-user networks [1].

A. Paper [5]

1) Purpose: Detection algorithm in the unexplored molecular communication channel.

2) Challenges: It is a realistic scenario with strong ISI and the channel is not memoryless which requires sequence detection.

3) Methods: A molecular communication experimental platform is established to generate an adequate dataset by repeatedly transmitting a consecutive sequence of N symbols from M possible types. Chemical signals, acids (representing bit-0), and bases (representing bit-1) are used to encode pH level information.

The detector is implemented using LSTM network, a typical algorithm for sequence processing belonging to RNN trained using the acquired experimental samples.

4) Results: LSTM based detector shows outstanding performance in a communication system with ISI

5) Conclusion: A sequence based estimator should be used in case of ISI. DL is promising when there is no model for the physical channel.

B. Paper: [6]

1) Purpose: Deep neural network for channel decoding (NND).

2) Challenges: Should learn a decoding structure rather than learning to classify 2^k different codewords. Block lengths are normally long but complexity of training increases exponentially with k.

3) Methods: Compares neural network decoding performance on polar block codes and random codes.

4) Results: NND for polar codes show possibility for generalization but not for random codes. The BER for a large NND is worse than MAP decoder for both random codes and polar codes. Neural network is able to generalize from a certain fixed SNR at training to any arbitrary SNR.

5) Conclusion: High decoding complexity and only achieves MAP for very short block lengths. But could possibly be improved with RNN and parallelized computation. Deep learning for channel decoding does not seem too promising.

C. Paper [7]

1) Purpose: Considers a MIMO channel with known channel matrix $\mathbf{H}$. Goal is to apply deep machine learning in the classical MIMO detection. Can be used as an example where deep learning can be used to trade off some of the exactness of an existing algorithm provides with faster computation times.

2) Challenges: Maximum likelihood already has really good performance but high computational complexity. A suboptimal implementation is desired.

3) Methods: Unfolding an iterative projected gradient descent method. Each iteration is represented as a layer in a neural network.

4) Results: Tested against a fixed channel model and a varying channel model with $\mathbf{H}$ drawn from a known distribution. Perforns promising both in fixed channels and a varying channel scenario. Performance is comparable to the advanced detector "semidefinite relaxation (SDR)" but computation is 30x faster.

D. Paper: [8]

1) Purpose: Channel estimation and signal detection use traditional communication solutions as initialization and uses DL networks to refine the coarse inputs.

2) Methods: Calls implementation ComNet. CE subnet first estimates OFDM channel from pilot symbols using LS. SD obtains ZF (zero forcing) estimate of the transmitted symbol. The obtained estimate, along with the estimated channel and received signal to the DL model to further refine the symbol estimates.

3) Results: Performs comparably to traditional decoding however, OFDM implementation allows symbols to be decoded when the CP is removed. Related works: Equalization and synchronization in OFDM.

E. Idea

Inspired by OFDM detection papers as [8], an mmWave model-driven receiver can be developed that uses expert knowledge to replace receiver blocks. NN can learn to compensate for non-linearities of mmWave components.

F. Idea

Most neural network detection methods are only created for a stationary channel. Work can be made to extend some of the already developed ML receiver detection methods to many different channel conditions.

III. End-to-end Communication System design

A. Paper [9]:

Rethinking the communication system which is currently optimized block by block for performance in relation to a model [10] proposes an end-to-end autoencoder to derive modulation and decoding schemes which shows performance comparative to traditional communication systems. The channel is treated as a layer in the neural network and therefore its differentiable functional form is needed. A functional form will always be a simplification which does not factor in all impairments of a real system such as hardware imperfections, varying channel conditions. Since the applications of deep learning in the physical layer shows great promise we need a model which can robustly account for these situations [9]

1) Purpose: To make an end-to-end communication approach which does not need a functional description of the channel.

2) Challenges: Needs channel gradients in order to perform backpropagation (optimization).

3) Related work: Previous work have circumvented a functional model using a two-phase training alternative. One approach has trained on a functional model and fine-tuned neural network parameters using a realistic channel [11]. Realizations of a realistic channel has been approximated using a GAN. Another approach applied supervised learning at the receiver and reinforcement learning at the transmitter.

4) Methods: Use a stochastic approximation technique to approximate gradients for the model called Simultaneous perturbation stochastic approximation that does not require knowledge of exact channel model.

5) Results: Achieves the theoretical BER for an AWGN channel without any assumption about the channel model but takes more epochs to converge compared to the case where channel model is known.

6) Conclusion: Successful end-to-end design is created when channel model is not a available or gradient calculation is too complex.

B. Paper [11]

1) Purpose: Communication system solely composed of NNs using unsynchronized SDRs.

2) Challenges: Does not know functional channel. Training data is obtained from over-the-air transmissions. Needs to deal with a continuous transmission with ISI and synchronization issues.

3) Methods: Two phase procedure. First: train the autoencoder using a stochastic channel model that should approximate as closely as possible the behavior of the expected channel. Second: transmitter sends a large number of messages over the actual channel and the corresponding IQ-samples are recorded at the receiver. These samples, together with the corresponding message indices, are then used as a labeled data set for supervised finetuning of the receiver. Can be seen as an way to speed-up the training.

4) Results: Performance comes close to a well designed conventional system. (What is the purpose if it is worse?)

5) Conclusion: First prototype of its kind. Does not work in varying channel conditions. Could be achieved by sporadic transmission of known messages or a very robust error-correcting code that would allow gathering a fine-tuning dataset on the fly.

C. Idea

1) Purpose: Find actual channel measurements to use as channel impairments: $h(\mathbf{z})$. Set up conventional transmitter-receiver and compare performance to neural network receiver.

2) Challenges: Channel model should represent all effects such as synchronization, fast fading, slow fading, additive noise.

3) Methods: Setup random symbol transmissions. Capture high resolution raw (unsynchronized, unequalized) signal (using SDR or spectrum analyzer) along with actual transmitted symbol (training data)). Learn the representation that maps the received signal back to symbols. Will be difficult because of ISI and varying channel conditions. Need a fixed block length as input.

Should account for many varying channel scenarios but this can turn out to be difficult. Areas of interest could be to create a clustering of channel conditions for which a subset of end-to-end networks are trained.

4) Extension: More channel realizations can be generated using a GAN which should capture the effects of the real channel. Does not need to map to discrete symbols. Could also be used for analogue transmissions since it will just be a regression problem.

D. Idea

Transmission of analogue sequences is not really considered in literature. The continuous nature of a neural network makes it straightforward to treat the communication problem as a high-dimensional regression problem. Continuous phase modulation is notoriously difficult and could be used in combination with analogue sequence transmissions. The application can be audio transmission, video transmission (multi-cast streaming) or continuous sensor data transmission.

E. Implementation Practices

Paper [12] by Toshiki, Toshiaki and Ye (MERL) will be used for inspiration for practical considerations.

Their DNN fails to learn multiplication between received signal and channel coefficients. Therefore the original input and the multiplication is used as input to the DNN. This increase of input dimensions may delay the convergence of DNN.

Mini-batch size: 128. 128x1000 pseudo random bits with the same channel coefficients over a mini-batch while Gaussian noise varies from sample to sample.

BER may not be appropriate performance measure for demodulation performance since practical systems use soft decision error-correction.

Trained DNN at every 5 dB, can be regarded as adaptive selection of mapping depending on the channel SNR.

Almost all papers I have read use Adam for stochastic optimization in DNN.

Sees good performance attributed to the DNN flexibly controlling the transmission rate depending on the channel condition at low SNRs. But is bad performance is expected at high SNRs since amplify and forward is maximum likelihood decision and DNN only approximates the ML decision with a non-linear function.

Assumes perfect synchronization and stable channel conditions. Only goal is to choose the constellation points.

Amplitude and phase information in signal constellation can be useless for unknown CSI.

Complex inputs are almost always represented as a one heat vector which is a concatenation of the real part and imaginary part.

IV. Full-duplex Communication

A. Paper [13]

1) Purpose: Learn to cancel self-interference for a full-duplex link in order to overcome the model deficit in the non-linear transmitter-receiver chain.

2) Challenges: Self interference is present at the receiver after the analog cancellation stage. Typically linear cancellation is used but this is usually not sufficient due to non-linear effects created by various transceiver impairments. Polynomial non-linear canceller comes with high implementation complexity.

3) Related work: The author is not aware of any SI cancellation in full-duplex radios in the literature using neural networks.

4) Methods: Using measured samples from a hardware testbed. IQ imbalance and PA non-linearities are normally the dominant non-linearities. Better performance is found if a linear-cancellation method is first applied to the received signal and then used as an input to the neural network.

5) Results: Matches the performance of the polynomial non-linear canceller with significantly lower computational complexity.

6) Conclusion: Reduces computation by 36% compared to polynomial non-linear canceller

V. CDMA/Grant free random access/Massive connectivity

Neural networks were employed in order to perform detection and intra-user (and mostly linear) successive interference cancellation in multi-user CDMA systems in:

B. Aazhang, B. P. Paris, and G. C. Orsak, Neural networks for multiuser detection in code-division multiple-access communications, IEEE Trans. Commun., vol. 40, no. 7, pp. 1212-1222, Jul 1992

M.-H. Yang, J.-L. Chen, and P.-Y. Cheng, Successive interference cancellation receiver with neural network compensation in the CDMA systems, in Asilomar Conference on Signals, Systems and Computers, vol. 2, Oct 2000, pp. 1417-1420.

B. Geevarghese, J. Thomas, G. Ninan, and A. Francis, CDMA in- terference cancellation techniques using neural networks in rayleigh channels, in International Conference on Information Communication and Embedded Systems (ICICES), Feb 2013, pp. 856-860.

VI. Modulation classification

Algorithm deficit, complex problem, optimal solutions are hard. Has been attempted many times with OK results.

VII. Unsupervised machine learning

Step 1: Model selection. Select a model (family of distributions parameterized by a vector θ.)

Step 2: Learning. Data should be used to choose the value for the parameter vector θ.

Step 3: Model is applied to carry out the task of interest. e.g. Clustering, dimensionality reduction or generation of new samples.

A. Autoencoders

The transmitted input message x has an intermediate representation z which is the received signal and the output should match the input. ML should only be used if a model or an algorithm deficit exists. Algorithm deficit: Non-linear dynamical models (optical links), multiple access channels with sparse transmission codes and joint source channel coding.

End-to-end communication described in Section III-A has examples of auto-encoder use.

Auto-encoders can also be used to compress Channel State Information (CSI) for Frequency Division Duplex (FDD) links.

B. Generative models

1) Channel realizations [14]: Example: Learn to generate samples from a given channel. Reasonable for scenarios that lack straightforward channel models. Can be used to mimic and identify non-linear channels for satellite communications. Can be generally used to augment a dataset used for training.

2) Detecting anomalies by learning the typical distribution of features: Can be used for spectrum sensing, identifying covert transmissions.

References

[1] O. Simeone, “A Very Brief Introduction to Machine Learning With Applications to Communication Systems,” *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 4, pp. 648–664, 2018. [Online]. Available: http://arxiv.org/abs/1808.02342

[2] X. Yang, M. Matthaiou, J. Yang, C.-K. Wen, F. Gao, and S. Jin, “Hardware-Constrained Millimeter-Wave Systems for 5G: Challenges, Opportunities, and Solutions,” *IEEE Communications Magazine*, vol. 57, no. 1, pp. 44–50, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8613274/

[3] G. Gui, H. Huang, Y. Song, and H. Sari, “Deep learning for an effective nonorthogonal multiple access scheme,” *IEEE Transactions on Vehicular Technology*, vol. 67, no. 9, pp. 8440–8450, Sep. 2018.

[4] M. Pajovic, T. Koike-Akino, and P. V. Orlik, “Model-Driven Deep Learning Method for Jammer Suppression in Massive Connectivity Systems,” *arXiv e-prints*, p. arXiv:1903.06266, Mar 2019.

[5] N. Farsad and A. Goldsmith, “Detection Algorithms for Communication Systems Using Deep Learning,” *CoRR*, vol. abs/1705.0, 2017. [Online]. Available: http://arxiv.org/abs/1705.08044

[6] T. Gruber, S. Cammerer, J. Hoydis, and S. T. Brink, “On deep learning-based channel decoding,” *2017 51st Annual Conference on Information Sciences and Systems, CISS 2017*, pp. 1–6, 2017.

[7] N. Samuel, T. Diskin, and A. Wiesel, “Deep MIMO detection,” *IEEE Workshop on Signal Processing Advances in Wireless Communications, SPAWC*, vol. 2017-July, pp. 1–5, 2017.

[8] X. Gao, S. Jin, C. K. Wen, and G. Y. Li, “ComNet: Combination of Deep Learning and Expert Knowledge in OFDM Receivers,” pp. 1–11, 2018.

[9] V. Raj and S. Kalyani, “Backpropagating through the air: Deep learning at physical layer without channel models,” *IEEE Communications Letters*, 2018.

[10] T. O’Shea and J. Hoydis, “An Introduction to Deep Learning for the Physical Layer,” *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563–575, 2017. [Online]. Available: http://ieeexplore.ieee.org/document/8054694/

[11] S. Dörner, S. Cammerer, J. Hoydis, and S. Ten Brink, “Deep Learning Based Communication Over the Air,” *Conference Record of 51st Asilomar Conference on Signals, Systems and Computers, ACSSC 2017*, vol. 2017-Octob, no. 1, pp. 1791–1795, 2018.

[12] T. Matsumine, T. Koike-akino, and Y. Wang, “Deep Learning-Based Constellation Optimization for Physical Network Coding in Two-Way Relay Networks,” -, -.

[13] A. Balatsoukas-Stimming, “Non-Linear Digital Self-Interference Cancellation for In-Band Full-Duplex Radios Using Neural Networks,” in *IEEE Workshop on Signal Processing Advances in Wireless Communications, SPAWC*, vol. 2018-June, 2018.

[14] T. J. O’Shea, T. Roy, and N. West, “Approximating the Void: Learning Stochastic Channel Models from Observation with Variational Generative Adversarial Networks,” *CoRR*, vol. abs/1805.0, 2018. [Online]. Available: http://arxiv.org/abs/1805.06350

Deep Learning for Synchronization and Channel Estimation in NB-IoT Random Access Channel

Mads H. Jespersen*, Milutin Pajovic†, Toshiaki Koike-Akino†, Ye Wang†, Petar Popovski*, and Philip V. Orlik†
*Department of Electronic Systems, Aalborg University, Denmark
†Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA 02139, USA

Abstract—**The central challenge in supporting massive IoT connectivity is the uncoordinated, random access by sporadically active devices. The random access protocol and activity detection have been widely studied, while the auxiliary procedures, such as synchronization, channel estimation and equalization, have received much less attention. However, once the protocol is fixed, the access performance can only be improved by a more effective receiver, through more accurate execution of the auxiliary procedures. This motivates the pursuit of joint synchronization and channel estimation, rather than the traditional approach of handling them separately. The prohibitive complexity of the conventional analytical solutions leads us to employ the tools of deep learning in this paper. Specifically, the proposed method is applied to the random access protocol of Narrowband IoT (NB-IoT), preserving its standard preamble structure. We obtain excellent performance in estimating Time-of-Arrival (ToA), Carrier-Frequency Offset (CFO), channel gain and collision multiplicity from a received mixture of transmissions. The proposed estimator achieves a ToA Root-Mean-Square Error (RMSE) of 0.99 µs and a CFO RMSE of 1.61 Hz at 10 dB Signal-to-Noise Ratio (SNR), whereas a conventional estimator using two cascaded stages have RMSEs of 15.85 µs and 8.05 Hz, respectively.**

Index Terms—**Deep learning, IoT standards, massive random access, joint estimation**

I. Introduction

A massive number of devices are expected to be connected to the Internet and several standards have been proposed to enable connectivity of low-complexity devices operating over a shared wireless channel. Most prominent technologies are Sigfox, LoRa and Narrowband IoT (NB-IoT) [1]. In Internet of Things (IoT) applications, the random access procedure has a high impact on device battery life and number of devices that can be supported concurrently [2]. Random access is used to request uplink allocation from the base station without requiring users to be constantly connected to the base station. Most IoT data packets are on the order of bits and users transmit them sporadically by establishing a new connection for every transmission. Establishing a connection using random access is a four step procedure [3], [4], which is initiated by a user that has packet to transmit by sending a random access preamble. The random access preamble is designed such that the base station is able to efficiently detect the transmitting user and estimate any timing offset between the user and base-station from the received signal. The timing offset comprises of propagation time, downlink synchronization errors and channel delay spread [5].

NB-IoT is a recent standard proposed by the 3rd Generation Partnership Project (3GPP) to accommodate the emerging number of wireless devices connected to the Internet. It is designed to co-exist with Long-Term Evolution (LTE) and provide low-cost and low-power devices with low throughput connectivity. The random access procedure in the NB-IoT is initiated by the Narrowband Physical Random Access CHannel (NPRACH). The NB-IoT has a system bandwidth of 180 kHz that accommodates 48 orthogonal channels from which a user attempting to establish a connection chooses one at random. If NB-IoT users choose different (i.e., orthogonal) preambles, the base station is able to estimate Time of Arrival (ToA) and Carrier Frequency Offset (CFO) of each user [5], [6]. However, given a possibly large number of users and relatively small number of orthogonal preambles, it is likely that two or more users choose the same preamble. The resulting collision may lead to a user back-off time of up to almost 9 minutes [7], [8]. In order to avoid unnecessary backoff periods and consequently improve channel utilization and overall capacity of the NB-IoT system, we propose in this paper a Deep Learning (DL)-based method for separating colliding users, detecting their number and estimating their respective ToAs and CFOs. We validate the proposed method using simulations and demonstrate significantly improved performance compared to the conventional approaches.

A. Related Work

Several papers have explored methods for activity detection, ToA and CFO estimation using the NB-IoT NPRACH preamble structure. As such, [5] estimates the ToA by searching for highest correlation between the received signal and delayed/frequency-shifted preamble on a grid of possible delays and frequencies. To reduce the complexity of the algorithm in [5], the ToA and CFO are estimated using the residual phase difference between symbol groups and channel hops in a two-stage procedure in [6]. With the goal to improve the ToA estimation, [4] suggests a novel hopping pattern that renders more accurate ToA estimation compared to that achieved with the already defined NB-IoT preamble.

We consider in this paper a problem of separating colliding NB-IoT users that choose the same random access preamble in the NPRACH scheme, and propose a method to detect the number of colliding users and estimate their ToA, CFO

The first author performed this work as an intern at MERL.

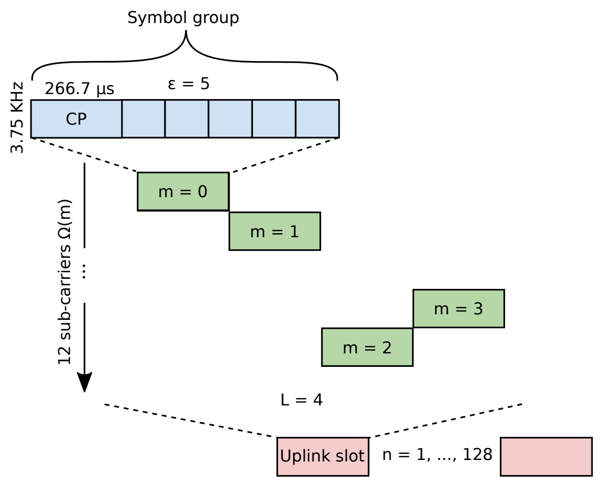


Fig. 1. Overview of NPRACH preamble and packet structure.

and channel gains. Motivated by recent success in leveraging learning-based methods for addressing problems related to physical layer communications [9], our method builds upon deep learning framework. In particular, we jointly detect the number of active users and estimate their parameters, with the aim to improve the capacity of the critical random access phase by not discarding interfering signals in order to utilize channel resources better, which in turn reduces back-off periods. In addition to handling much richer class of scenarios, the proposed method outperforms [6] in their own scenario where users transmit orthogonal preambles and do not collide. In comparison to [4], the random access preamble in this work is as suggested by the NB-IoT standard, ensuring the proposed method is practical in the NB-IoT systems currently being deployed. Finally, looking outside the NB-IoT scope, we believe that this work is the first application of deep learning techniques for user separation in massive connectivity systems.

II. NB-IoT Random Access Preamble Design

The preamble format and packet structure are illustrated in Fig. 1. The preamble is divided into symbol groups, where each group consists of a Cyclic Prefix (CP) and ε identical symbols. The value of ε depends on preamble format. The preamble format is chosen by the user based on the downlink power measurement to estimate its coverage area [3].

The most common preamble format is format 1 with preamble frame structure 0 or 1, which has $\varepsilon = 5$ and a symbol time $T_{\text{SYM}} = 266.7\,\mu\text{s}$. The CP period for frame format 0 is $T_{\text{CP}} = 66.7\,\mu\text{s}$ and $T_{\text{CP}} = 266.7\,\mu\text{s}$ for frame format 1 [10]. The CP is designed such that it is long enough to cover the maximum round trip delay to suppress Inter-Symbol Interference (ISI). Therefore one interpretation of allowing adaptive CP selection is for the user to use the short CP in the range 0–8 km and the long CP in the range 8–35 km [5].

The full preamble consists of 4 repetitions of the symbol group which is again repeated $n = 2^J, J = 0, \ldots, 7$ times for a full preamble length of $L = 4 \times 2^J$ symbol groups. The repetition of the symbol groups occurs within an uplink slot, and the number of repetitions is decided by the upper Medium Access Control (MAC)-layer depending on estimated link quality [10]. For simplicity we consider $J = 2$, i.e., four symbol groups are repeated 4 times.

Before transmission, the user chooses a contiguous set of $N = 12, 24, 36$ or 48 subcarriers with $3.75\,\text{kHz}$ spacing out of the available 48 subcarriers. This paper focuses on the preamble frame structure type 1 where $N = 12$. At the start of the NPRACH preamble transmission, the subcarrier of the first symbol group is chosen at random. After each symbol group the subcarrier will change using a deterministic channel hopping sequence so in the duration of a preamble there will be L subcarrier hops. Since the hopping pattern is deterministic, several users choosing the same initial subcarrier will thus collide for the entirety of the NPRACH preamble sequence. The number of orthogonal preamble sequences is therefore the number of allocated NPRACH subcarriers, K [7].

For frame structure type 1 and preamble format 0, two “levels” of hopping are employed as shown in Fig. 1. The hopping pattern is deterministic within a cell, but the subcarrier of every 4th symbol group appears random to neighbouring cells. The hopping procedure aids in the estimation of ToA and also reduces inter- and intra-cell interference [5]. The ToA should be estimated by the base station for successful uplink signal decoding and it further enables device positioning. Error in the ToA estimation results in the user being unable to receive the response sent by the base station. ToA estimation therefore has a great impact on performance in NB-IoT [4].

III. System Model

The received signal at the base station is a superposition of signals from multiple users, given by

$$y[n] = \sum_{k=0}^{K-1} a_k s_k[n] + w[n], \tag{1}$$

where K is the maximum number of concurrent users, $a_k \in \{0, 1\}$ indicates whether the kth user is active or not, and $w[n] \sim \mathcal{CN}(0, 1/\rho_n)$ denotes the additive noise with a per symbol Signal-to-Noise Ratio (SNR) of ρ_n.

At the receiver, the phase of each symbol depends on the ToA τ, the CFO Δf (which gives the frequency of the user's chosen channel with respect to the receiver's uplink carrier frequency f), and the channel rotation given by $\arg(h)$, where h is the complex-valued channel coefficient. These parameters are assumed to be independent across users and denoted by τ_k, Δf_k and h_k for each user k.

The signal from the kth user is given by

$$s_k[n] = h_k \, \mathrm{e}^{-\jmath 2\pi (f_n + \Delta f_k)(n T_{\text{sym}} - \tau_k)}, \tag{2}$$

where T_{sym} is the symbol duration. The signal model is limited to only considering a single preamble sequence for the sake of simplicity. This means that the sub-carrier frequency pattern f_n is predetermined and identical for all instances of s.

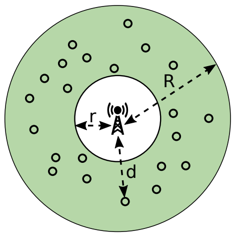


Fig. 2. Geometry of users distribution.

The typical FFT length in LTE is 512 [6], but for simplicity we describe that each sample, n, corresponds to a symbol. In this model, the contents of the CP are interpreted as a symbol and therefore no distinction is made between the CP and the $\varepsilon = 5$ repeated symbols in a symbol group. This signal model may be valid only for the long CP which corresponds to distances between the user and base station within a minimum of $r = 8\,\mathrm{km}$ and a maximum of $R = 35\,\mathrm{km}$ [5]. The users are assumed to be uniformly distributed in the coverage area of the base station as illustrated in Fig. 2. The distance from the base station to the users d has the following Probability Density Function (PDF) [11]:

$$f_D(d) = \frac{2d}{R^2 - r^2}, \quad r \le d \le R, \tag{3}$$

which is used to model the ToA $\tau = \frac{d}{c}$, where c is the propagation speed.

The channel coefficient h of the signal model in (2) is a complex-valued constant which accounts for small scale fading: $h \sim \mathcal{CN}(0, 1)$. This means that the average received signal power is normalized to one. The narrowband channel is modeled as a slowly varying single-tap Rayleigh fading channel and for this reason, modeled as a single coefficient [6]. Large scale fading is not included in the model since users already have knowledge of the downlink SNR and adjust their transmit power accordingly using power control.

The CFO in (2) is chosen uniformly at random between -20 and $+20$ Hz [6]. For the sake of simplicity, the CFO and ToA are assumed to be constant throughout an entire NPRACH transmission for each user.

The activity indicator a_k is modeled as Bernoulli random variable with the probability of transmitting p and $a_1, \ldots, a_K$ are iid. The number of concurrent active users is

$$N_a = \sum_{k=1}^{K} a_k \sim \mathcal{B}(K, p), \tag{4}$$

where $\mathcal{B}$ is the binomial distribution. We consider the case with $K = 4$ and $p = 0.5$ throughout the paper. The probability of exactly k users colliding is then:

$$\Pr(k) = \binom{K}{k} p^k (1-p)^{K-k} = \frac{4!}{k!(4-k)!} p^k (1-p)^{4-k}. \tag{5}$$

IV. Deep Learning Estimator

The goal of the estimator is to use the discrete signal $y[n]$ to estimate the activity indicator a, ToA τ, CFO Δf, and channel coefficient h of each user. Since the activity indicator of each user is a random variable, the total number of active users in the received signal is unknown. This boils down to a notoriously challenging problem of source separation with unknown number of users [12]. Deep learning has significantly improved the field of source separation and the general idea of using deep learning is to capture non-linear relationship between inputs and corresponding targets that is often difficult to model with analytically tractable expressions [12]. In this paper, estimating the unknown parameters is dealt with by splitting the problem into:

- Classification of the number of active users; and
- Estimation of ToAs, CFOs and channel coefficients given the number of users.

The two separate tasks are combined such that the synchronization parameters are accurately estimated for each detected user.

A. Estimation of the Number of Users

Finding the number of active users, N_a, is formulated as a classification problem where $\mathbf{p} = \mathsf{OneHot}(N_a)$ is a categorical random variable encoded as a one-hot vector specifying N_a. With a one-hot encoding, the true target $\mathbf{p} = [p_0, p_1, \ldots, p_K]$ has entry one at index N_a, and zero entries everywhere else. This is different from a typical way of representing active users where users are ordered in a vector and each index indicates the activity of a unique user. The number of users N_a can then be estimated as the l_0 norm of that sparse vector. In this collision scenario users are transmitting using the same spreading sequence and are not uniquely distinguishable. For this reason, only the information on the number of active users is represented in $\mathbf{p}$.

Cross-entropy loss is typically used in classification problems [13], and [14] suggests that the cross-entropy loss in classification problems leads to faster convergence and better generalization compared to the Mean Squared Error (MSE). For nonbinary classification, we typically use softmax cross entropy loss (or negative log-likelihood) expressed as:

$$\ell_{\mathrm{NLL}}(\mathbf{p}, \mathbf{q}) = -\sum_{k=0}^{K} p_k \log q_k, \tag{6}$$

where $\mathbf{q}$ is a continuous differentiable softmax function:

$$q_k = \frac{\exp(\pi_k)}{\sum_i \exp(\pi_i)}, \tag{7}$$

where $[\pi_0, \pi_1, \ldots \pi_K]$ are the outputs from the last layer of the neural network and $[q_0, q_1, \ldots q_K]$ represent the *a posteriori* class probabilities. A hard class prediction could then be found as $\arg\max_i[\pi_i]$.

B. Parameter Estimation

The parameters to be estimated are collected in a vector

$$\mathbf{x}_k = \left[\tau_k, \Delta f_k, \Re[h_k], \Im[h_k]\right]^T. \tag{8}$$

Note that it was found that representing the complex-valued channel coefficient h by Cartesian coordinate (i.e., real and

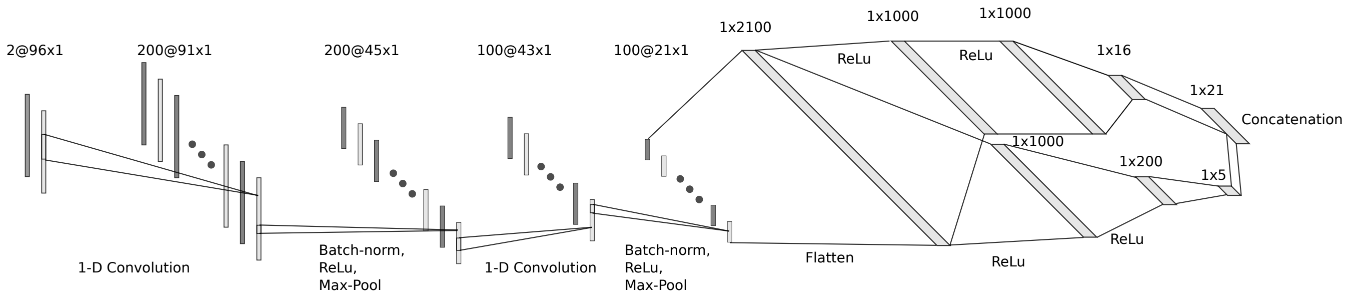


Fig. 3. Overview of DNN architecture for estimating synchronization parameters of up to 4 colliding users.

imaginary parts) shows superior performance to phasor representation (i.e, amplitude and phase) as seen in Fig. 7. For K users, the respective vectors are collected in a matrix

$$\mathbf{X} = [\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_{K-1}]. \tag{9}$$

The neural network seeks to find an estimate $\hat{\mathbf{X}}$ such that $\mathbb{E}\|\mathbf{X} - \hat{\mathbf{X}}\|_2^2$ is minimal which is equivalent to a Minimum Mean-Square Error (MMSE) estimator.

The above formulation is sufficient to derive an estimation procedure. However, $\mathbf{X}$ consists of multiple parameters which have values on different scales. When using a practical optimization algorithm to find an estimate, any scaling difference between the parameters will affect the impact each value has on the gradient descent step.

To circumvent possible issues arising from error variations across parameters, we minimize the reconstruction error instead. The actual received signal without additive noise, $\mathbf{s}$, with the parameters in matrix $\mathbf{X}$ can be reconstructed using (2). The reconstruction is conveniently represented using function $f(\cdot)$ such that $\mathbf{s} = f(\mathbf{X})$.

For each estimate $\hat{\mathbf{X}}$, the equivalent noise-free signal $\hat{\mathbf{s}}$ is reconstructed and compared to the actual noise-free received signal $\mathbf{s}$. The noise-free signal is known during the training procedure and is used so the output of the neural network does not account for the distribution of the noise. The data fidelity (i.e., reconstruction loss) is quantified using the MSE metric such that

$$\ell_r(\mathbf{X}, \hat{\mathbf{X}}) = \mathbb{E}\big\|f(\mathbf{X}) - f(\hat{\mathbf{X}})\big\|_2^2 = \mathbb{E}\big\|\mathbf{s} - \hat{\mathbf{s}}\big\|_2^2. \tag{10}$$

The number of concurrent users in each sample is known during training so when reconstructing the signal $\hat{\mathbf{s}}$, the contributions from the correct number of users are taken into account when calculating the reconstruction loss ℓ_r for each sample.

The loss function which the neural network seeks to minimize is simply the sum of (6) and (10)

$$\mathsf{loss} = \ell_p(k, \mathbf{q}) + \ell_r(\mathbf{X}, \hat{\mathbf{X}}). \tag{11}$$

C. Network Implementation

An overview of the neural network that estimates both the number of users and synchronization parameters is illustrated in Fig. 3. The input to the network is the received signal which consists of 4 NPRACH repetitions each with $L(\epsilon+1)$ symbol periods. The total number of samples in the received signal is: $N_{\text{rep}}L(\epsilon + 1) = 4 \cdot 4 \cdot (5 + 1) = 96$, where the real and imaginary parts are represented in 2 individual channels.

The output of the network is the flattened matrix $\mathbf{X}$ and the probability vector $\boldsymbol{\pi}$. For 4 users there are $4 \cdot 4 = 16$ parameters in $\mathbf{X}$ and 5 possible classes in the number of users (including the zero users case). The input to the network is processed so as to extract common features that are subsequently used for multi-task learning, that is, to detect the number of users and estimate their parameters. The first layer performs a 1-dimensional convolution over the input signal. Since the number of users, ToA, CFO and channel coefficient all are assumed to be constant throughout a transmission, a convolution layer is chosen so as to extract translationally invariant features of the input time-domain signal.

Following a typical CNN structure, batch normalization, non-linear activation and max-pooling are employed. The convolution layers, activations and pooling layers are repeated to form a deep neural network. The features found by the convolution layers are reshaped to a single vector which is then used as input to two individual feedforward neural networks. One of the networks performs classification and detects the number of users based on the output of the feature extraction layers. The other network performs regression with the goal to yield parameters so that the reconstructed signal is as close as possible to the received signal in the MSE sense. Each feedforward network has two fully connected layers followed by the Rectified Linear Unit (ReLU) activation and a linear output layer. The network and automatic differentiation are implemented using the PyTorch framework [15] and trained using multiple Graphics Processing Units (GPUs).

In the simulation ToA, CFO and channel coefficient are all drawn according to the distributions given in the system model and N_a is drawn according to $\Pr(k)$ for each sample. The input to the network $\mathbf{y}$ and each parameter in the output $\mathbf{X}$ is scaled to have zero mean and unit variance. In general the convergence of a neural network is faster if all inputs to all layers have zero-mean and unit covariance between training examples in the case when all examples are of equal importance [16]. From the system model the variance and mean of each parameter (CFO, ToA and h) are known and used to normalize the parameters to have mean zero and unit variance. The mean and variance of τ can be derived from (3)

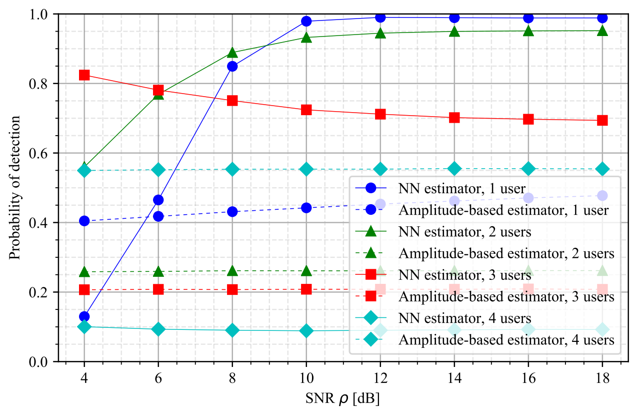


Fig. 4. Accuracy of estimating the number of colliding users. A signal is deemed correctly detected if the number of users are estimated correctly. The NN estimator is trained for signals with 10 dB SNR.

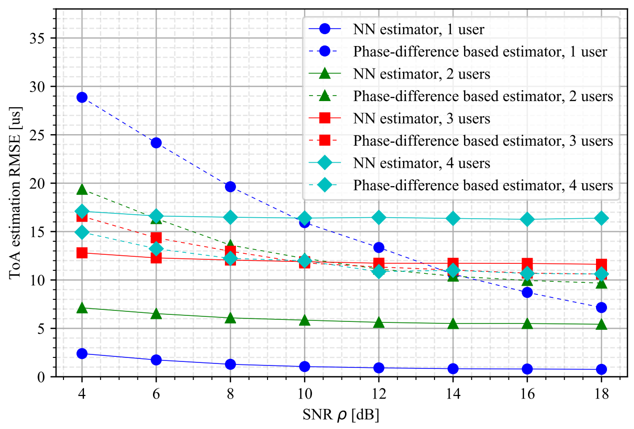


Fig. 5. Root-Mean-Square Error (RMSE) of ToA estimation across SNRs.

and the standardized ToA is given by

$$\tau' = \frac{\tau - \mathbb{E}[\tau]}{\mathsf{Var}(\tau)}. \tag{12}$$

The CFO, Δf, is scaled similarly. No normalization is necessary for the channel coefficients since $h \sim \mathcal{CN}(0,1)$ and thus no scaling is necessary for the signal $\mathbf{y}$.

V. Estimation Results

A. Traditional Methods

The phase-difference based method proposed in [6] utilizes the relationship between the phase trace of the received signal and the ToA and CFO. Phase differences between symbols in the received signal are averaged to estimate CFO. The ToA is found by subtracting the phase due to the estimated CFO from the phase of the received signal and averaging the phase difference between symbol groups on different frequencies.

As a benchmark for the detection of the number of users, an amplitude-based estimator is considered. The mean amplitude of the received signal for different number of colliding users is compared to the amplitude of the received signal. The closest match then yields an estimate of the number of colliding users present in the received signal.

B. Simulation

The neural network is trained using samples generated with up to $K = 4$ concurrent users and at an SNR of 10 dB. New batches are generated for every step in the training procedure. The learning rate is 0.0001 and each batch consists of 50,000 realizations of $\mathbf{y}$ from (1). The stochastic optimization method based on adaptive momentum (ADAM) [17] is used and a total of 20,000 different batches are used in training.

In Fig. 4, the estimation of collision multiplicity is shown for the proposed classification method compared to a simple amplitude-based method. As colliding signals will add non-coherently, the amplitude of the signal is not a good indicator on collision multiplicity. 1 and 2 users are successfully identified with 98.0 % and 93.2 % at an SNR of 10 dB and the estimation accuracy decreases with the number of concurrent users. The proposed method often miss-classifies a signal containing 4 colliding users as resulting from transmissions of 3 users.

Since the loss function only depends on the reconstruction error, the estimated parameters in $\hat{\mathbf{X}}$ are arbitrarily ordered across users. To compare the output with the target $\mathbf{X}$ the parameters are ordered according to the estimated amplitudes. In cases where the estimated amplitudes are similar, the ordering may be wrong which leads to an artificially high error when evaluating performance for multiple users.

The RMSE of each parameter in $\mathbf{X}$ is calculated as:

$$\mathsf{RMSE}_k = \sqrt{\mathbb{E}\left[\|e_k\|_2^2\right]}, \tag{13}$$

where e.g. the estimation error of τ is: $e_k = \tau_k - \hat{\tau}_k$. The RMSE of the proposed neural network-based estimator is the average of all RMSEs up to user k:

$$\mathsf{RMSE}_{\mathrm{NN},k} = \frac{1}{k}\sum_{i=1}^{k} \mathsf{RMSE}_i. \tag{14}$$

The conventional estimator is only able to estimate a single set of parameters, regardless of the actual number of users k. The error of the conventional estimator is therefore measured as the estimate which has the smallest error over all actual sets of parameters in $\mathbf{X}$, e.g. the estimated ToA error is

$$e_{\tau,\mathrm{PD}} = \min_k(|\tau_k - \hat{\tau}_{PD}|). \tag{15}$$

This gives the conventional estimator an artificial advantage.

The RMSE of ToA and CFO estimation with a varying number of users are shown in Figures 5 and 6. The neural network-based estimator shows lower estimation error for both ToA and CFO compared to the phase-difference-based estimator even for a single user. For two users the proposed estimator is superior to the conventional estimator when estimating ToA. At 10 dB the proposed estimator has an RMSE of 0.99 µs and 1.61 Hz for a single user compared to 15.85 µs and 8.05 Hz

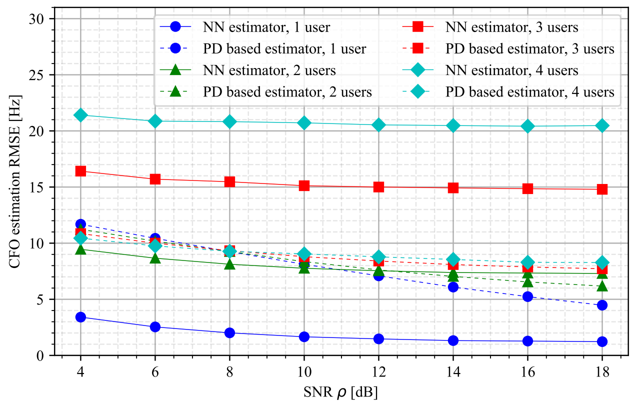


Fig. 6. RMSE of CFOs estimation across SNRs.

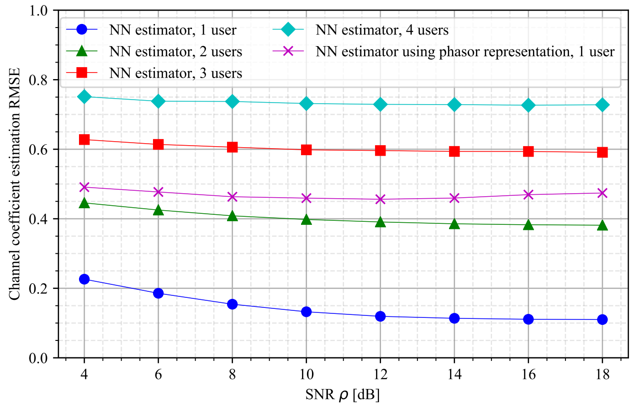


Fig. 7. RMSE of channel coefficient estimation across SNRs.

for the conventional estimator. The relatively high RMSE of the conventional estimator is likely due to the noise which causes wrong phase unwrapping at low SNRs [6].

The accuracy in estimating the channel coefficient h is shown in Fig. 7. The RMSE is 0.101 for the in-phase part and 0.103 for the quadrature part for a single user. The RMSE shows a similar trend as in ToA and CFO estimation with deteriorating performance as the number of concurrent users increases.

Overall the proposed method presents considerably improved performance compared to the traditional estimator in scenarios with a single, as well as multiple users.

VI. Discussion and Conclusion

We proposed a novel approach to synchronization and channel estimation. The system model consists of a superposition of an unknown number of users transmitting with the same preamble sequence. Deep learning is used to classify the multiplicity of collisions and estimate ToA, CFO and the channel coefficients for all user simultaneously.

The method is demonstrated in NB-IoT NPRACH where the number of orthogonal preambles is limited. The estimation error of a conventional approach in NB-IoT is compared to the performance of the proposed scheme. Traditional synchronization methods fail in the case of collisions with high Signal-to-Interference Ratio (SIR) whereas, with the proposed algorithm users can be distinguished and respective synchronization parameters can still be estimated with a reasonable performance.

Deep learning is a promising tool for developing joint estimation procedures, which are notoriously difficult in traditional model-based methods, and enables separation of synchronization parameters even when users transmit using the same preamble. Although deep learning-based estimation will lead to sub-optimal estimators compared to an analytically derived joint estimator, it allows for practical, straightforward development and efficient computation.

References

[1] I. Catherine, R. Tardy, N. Aakvaag, B. Myhre, R. Bahr, I. Catherine, R. Tardy, N. Aakvaag, B. Myhre, and R. Bahr, “Comparison of wireless techniques applied to environmental sensor monitoring,” SINTEF Digital, Trondheim Norway, Tech. Rep., 2017.

[2] A. Azari, P. Popovski, G. Miao, and C. Stefanovic, “Grant-free radio access for short-packet communications over 5G networks,” in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Dec 2017, pp. 1–7.

[3] Y. . E. Wang, X. Lin, A. Adhikary, A. Grovlen, Y. Sui, Y. Blankenship, J. Bergman, and H. S. Razaghi, “A primer on 3GPP narrowband Internet of things,” *IEEE Communications Magazine*, vol. 55, no. 3, pp. 117–123, March 2017.

[4] W. S. Jeon, S. B. Seo, and D. G. Jeong, “Effective frequency hopping pattern for ToA Estimation in NB-IoT Random Access,” *IEEE Transactions on Vehicular Technology*, vol. 67, no. 10, pp. 10 150–10 154, 2018.

[5] X. Lin, A. Adhikary, and Y. P. Eric Wang, “Random access preamble design and detection for 3GPP narrowband IoT systems,” *IEEE Wireless Communications Letters*, vol. 5, no. 6, pp. 640–643, 2016.

[6] J. Hwang, C. Li, and C. Ma, “Efficient detection and synchronization of superimposed NB-IoT NPRACH preambles,” *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 1173–1182, Feb 2019.

[7] L. Feltrin, G. Tsoukaneri, M. Condoluci, C. Buratti, T. Mahmoodi, M. Dohler, and R. Verdone, “Narrowband IoT: A survey on downlink and uplink perspectives,” *IEEE Wireless Communications*, vol. 26, no. 1, pp. 78–86, February 2019.

[8] 3GPP TS, *36.321 - Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol specification, V. 15.4.0*, 2018.

[9] O. Simeone, “A Very Brief Introduction to Machine Learning With Applications to Communication Systems,” *arXiv e-prints*, p. arXiv:1808.02342, Aug 2018.

[10] 3GPP TS, *36.211 - Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulation, V. 15.4.0*, 2018.

[11] T. Hien, Z. Wang, S. Kim, J. Nielsen, and P. Popovski, “Preamble detection in NB-IoT random access with limited-capacity backhaul,” in *Proceedings of IEEE ICC’19*, 2 2019.

[12] J. Chien, *Source Separation and Machine Learning*. Elsevier Science, 2018.

[13] C. Bishop, *Pattern Recognition and Machine Learning*, ser. Information Science and Statistics. Springer New York, 2016.

[14] P. Y. Simard, D. Steinkraus, and J. C. Platt, “Best practices for convolutional neural networks applied to visual document analysis,” in *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, Aug 2003, pp. 958–963.

[15] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in PyTorch,” in *NIPS-W*, 2017.

[16] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, “Efficient backprop,” in *Neural Networks: Tricks of the Trade*. Springer-Verlag, 1998.

[17] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv e-prints*, p. arXiv:1412.6980, Dec 2014.