Deep Learning For Noisy Communication System

Reem E. Mohamed*, Robert Hunjet†, Saber Elsayed*, Hussein Abbass*
* School of Engineering and Information Technology, University of New South Wales, Canberra ACT, Australia
Emails: r.sherif@student.adfa.edu.au, r.emk.sherif@gmail.com
Emails: {s.elsayed, h.abbass}@unsw.edu.au
†Defence Science and Technology Group, Edinburgh SA, Australia
Email: Robert.Hunjet@dst.defence.gov.au

***Abstract*—In modern communication systems, channels vary with time due to the mobility of the transmitter (Tx) and the receiver (Rx). To achieve error-free communication in changing environments, Tx and Rx must effectively adapt to different noise levels. However, exiting frameworks face challenges during adaptation, leading to poor communication. In this paper, an independent pre-training collaborative learning framework is designed for tuning Tx and the Rx. The proposed framework incorporates more realistic challenges encountered in communication systems such as lack of feedback channels and the lack of updates at the Rx side. The experimental results show that the proposed approach can reduce noise up to 50% more than existing approaches and within a short training time.**

***Index Terms*—noisy channel, internal noise, deep neural network, autoencoder, noise reduction**

I. Introduction

In communication systems, both transmitters (Tx) and receivers (Rx) are susceptible to internal noise. Noise in a communication channel can corrupt the transmitted data resulting in Rx being unable to correctly interpret the data sent [1]. In modern communication systems, noise is also introduced via mobility of the end-systems and changes in the environment. However, recovering data after a high level of corruption has been an overarching problem in the last decade. Deep learning (DL) methods have been employed by modelling a neural network (NN) at the communication end-systems to remove noise from the transmitted signal at the Rx in the presence of a noisy communication channel [2; 3; 4; 5]. In [2], an end-to-end learning was introduced, namely a whole communication system that can be trained in a supervised manner using stochastic gradient descent (SGD) [6].

Removing noise from data has gained interest from researchers in different domains in the last century. De-noising Autoencoder (AE) was introduced earlier [7] for image noise reduction as an alternative to Hopfield models [8], and in the last decade, the concept of pre-training NNs [9] was introduced for AEs and tested on image datasets. Consequently, more attempts to use AE for improving the performance of NNs were carried out [10] and improvements achieved by the de-noising AE introduced in [11]. De-noising AE was shown to perform better than the basic AE on noise reduction tasks applied to different image datasets. An autoencoder (AE) is an NN that is composed of three different stages that are performed sequentially by layers of neurons where (a) the data encoding takes place in the encoding layers; (b) the encoding function is performed in the state approximation layers; and (c) the data decoding is performed at the decoding layers. Recently, the concept of noise reduction in a noisy communication system has been developed by modelling the Tx as the encoder, the Rx as the decoder and the channel as the encoding function [12]. The feasibility of using deep learning on mobile devices has been demonstrated in face recognition applications [13; 14]. It is important to say, though, that the size of the network for most communication problems is significantly smaller than those used in facial recognition. Despite the ability of the Tx and Rx to learn quickly, the approach assumed that the transfer function of the wireless communication channel can be static and modelled by a specific decoding function. This assumption has limited validity given the time-varying nature of wireless communication systems; consider the effect of both the internal noise at each end-system as a result of the change in its environment, and mobility of nodes causing constant changes in the spectrum. This unpredictability of the channel was recently studied in [15].

Furthermore, modelling the stochastic communication channel (CH) as a part of an NN hinders the practical implementation of the model because, in that case, the gradient of the instantaneous channel transfer function must be known, which is not applicable to a wireless channel in a dynamic environment [1]. As a result, the authors in [16] suggested fine-tuning the *receiver* based on the measured data through an initial learning process on the channel model. However, fine-tuning the *transmitter* was not considered, which resulted in sub-optimal performance. As a result, further attempts were conducted where the channel was modelled as a black box for which only inputs and outputs can be observed. Another recent approach applied Deep Reinforcement Learning (RL) concepts to not only overcome the noise on a communication channel but also the perturbation at the Tx during an alternative training process that is independent of the channel model [5]. Although the approach was closer to physical reality, the Rx was still assumed to be noiseless and the performance of the Rx in removing the noise from the transmitted data was improving at a noticeably slower rate.

In this paper, we study the effect of different levels of noise

978-1-7281-8393-0/21/$31.00

in the end-devices, namely internal noise, and the effect of channel noise on the performance of our proposed learning framework applied on common communication system model. We proposed a framework that reduces the effect of the internal noise in each end-system individually and then reduces the effect of the channel noise on the received data through collaborative learning. We compare its performance with an RL-based model. Also, we employ different variations in the existing RL-based model and the communication system for our model to study the effect of each component in the communication system on the performance of data de-noising at the Rx side. The results demonstrate that the proposed framework makes the communication system more resilient to high levels of internal noise and Rx noticeably efficient in channel noise reduction compared to its peers. The contributions of this paper can be summarised as follows:

- Design of an AE at the Tx and the Rx to overcome internal device noise problem
- Development of an efficient two-phase training method for the communication end-systems that allows the Tx and Rx to reach low level of error in a very short learning time even in challenging transmission cases in the presence of either noiseless or noisy communication channels.

The rest of this paper is organised as follows. The problem formulation is discussed in Section II. In Section III, the proposed two-phase learning method for end-systems in a communication system (IPCL) is explained in detail. In Section IV, the results of the proposed method are investigated in different Signal to Noise Ratio (SNR) scenario with different variations in IPCL elements. Finally, conclusions are drawn in Section V along with a discussion of future research directions.

II. Problem Formulation

In this paper, we consider a communication system with one Tx and one Rx which both experience internal noise distributed as an additive white gaussian noise (AWGN), with zero mean ($\mu = 0$) and standard deviations σ_{Tx} and σ_{Rx} for the Tx and Rx, respectively. According to our framework, each end-system has a Neural Network denoted by NNT and NNR inside the Tx and Rx, respectively. The two end-systems (the Tx and Rx) communicate through stochastic forward channel (CH) and a feedback-channel (F-CH) that can be described as a random function with probability $y = P(.|x)$ for each input x. A data exchange scenario through this system is depicted in Figure 1 and can be described in the following sequence:

1) The Tx transmits a message x, but due to its internal noise, it becomes $x' = x + N(0, \sigma_{Tx})$, then, the NNT transmits its estimated value of $\hat{x}$ through CH.
2) The CH transforms the $\hat{x}$ into y; thus, $y = P(.|\hat{x})$.
3) The Rx receives y, but due to its internal noise, it becomes $y' = y + N(0, \sigma_{Rx})$, then, the NNR interprets it as $\hat{y}$.
4) The F-CH transforms the $\hat{y}$ into z; thus, $z = P(.|\hat{y})$, which means that the Tx becomes aware that the transmitted data x was interpreted at the Rx as z.

To minimise the error between the message (x) and its value at the Rx output ($\hat{y}$), the MSE should be minimised. The losses in terms of MSE at the Tx and Rx sides are minimised by finding the optimised NN parameters (θ_{Tx} and θ_{Rx}) as formulated in Eqs. (1) and (2), respectively. The MSE minimisation takes place during the individual as well as collaborative training phases to reduce the internal and channel noises, respectively. Since the actual speed of training will depend on the local hardware available to each agent, we use the number of epochs as an approximation of this speed. An epoch is the time taken to pass one message between both end-systems and receive its feedback when available. During training, learning occurs and therefore both θ_{Tx} and θ_{Rx} get updated. During testing, these updates do not take place.

$$\min_{\theta_{Tx}} \quad L(\theta_{Tx}) = \frac{1}{Ep_{indT}} \sum_{t=0}^{t=Ep_{indT}} (\hat{x_t} - x_t)^2 \qquad (1)$$

$$\min_{\theta_{Rx}} \quad L(\theta_{Rx}) = \frac{1}{Ep_{indT}} \sum_{t=0}^{t=Ep_{indT}} (\hat{y_t} - y_t)^2 \qquad (2)$$

where Ep_{indT} is the number of epochs during the independent training phase, all the messages are vectors with size B, x_t and y_t are the actual generated messages at different random seeds at the Tx and the Rx, respectively, while the $\hat{x_t}$ and $\hat{y_t}$ are the estimated messages at the NNT and NNR, respectively.

We show that despite having two different entities in the system (the Tx and the Rx) that have to optimise their internal parameters (θ_{Tx} and θ_{Rx}, respectively), Tx can send a modified version of x that the Rx can interpret as x with minimum error. The problem of de-noising the data at the Rx side is defined as the minimisation of the MSE between the original transmitted data and the final output at the Rx side. The minimisation process of the MSE in the whole communication system, formulated in Eq. (3), needs to be fast because of the stochastic time varying nature of the wireless communication channel with different signal fading and shadowing effects

$$\min_{(\theta_{Tx}, \theta_{Rx})} \quad L(\theta_{Tx}, \theta_{Rx}) = \frac{1}{Ep_{indT}} \sum_{t=0}^{t=Ep_{indT}} (\hat{y_t} - x_t)^2 \qquad (3)$$

III. Cooperative Learning Method

This section describes the independent pre-training collaborative learning (IPCL) framework that includes (a) a design of the NN at the end-systems which is inspired by the success of Autoencoders (AEs) in noise reduction; and (b) a two-phase sequential learning method that is carried out at the NNs of the end-systems which is inspired by the deployment of DL methods in the communication process [5] without adding assumptions about the communication channels.

A. End-system Independent Learning

Based on the observations of the policy-based method in [5], the learning process takes a long time to stabilise, especially when the internal noise is high, regardless of the CH noise

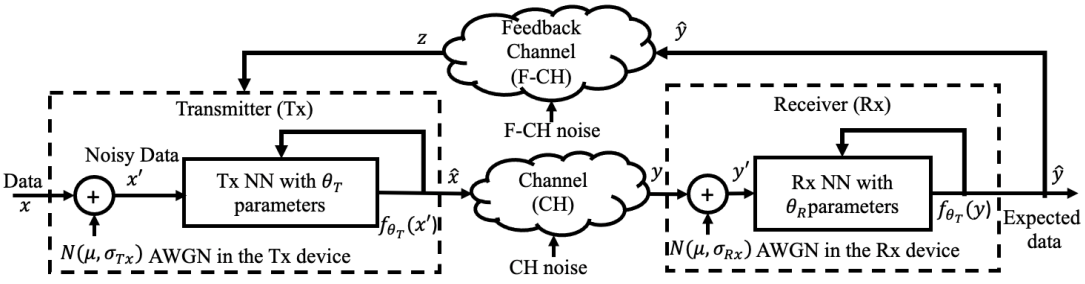


Fig. 1. The Communication Model

level. Combining this observation and the recent improvements in deploying NN for noise reduction [17], we propose a pre-training phase that is individually performed by the Tx and Rx in a completely independent fashion. This is because the Tx and Rx are separate devices located in different places, so allowing each device to get over its internal noise individually facilitates their ability to initialise their communication process and get over the noise introduced by their communication channels in a short time.

In this phase, the input for the Tx is a set of randomly generated independent and identically distributed (i.i.d.) data messages x, each is a vector of size B and the output is the data that will be sent through the channel $\hat{x}$. The aim is to find the NN parameters that minimises the MSE between x and $\hat{x}$, which is defined in Eq. (1). Similarly, the same process is performed at the Rx side independently where the loss function is Eq. (2). The process shown in Algorithm 1 allow the Tx and Rx to independently map the noisy inputs to the actual inputs by finding the optimal NN parameters (θ^*_{Tx} and θ^*_{Rx}) during the training time (Ep_{indT}).

This learning phase facilitates cooperative learning between the Tx and the Rx resulting in fast convergence, which minimises the time required in the initialisation of the communication process. We propose AEs for the Tx and Rx NNs due to their efficiency in noise reduction in different domains [17], as described in the left sub-figure in Fig. 2. We designed a simple AE at both ends of the communication system where Adam optimiser [18; 19] is used to search for the optimal NN parameters that lead to the minimum loss during the limited independent training time (Ep_{indT}). Adam optimiser has an adaptive step size that follows the update rule described in Eq. (4) [18]

Algorithm 1: IndependentTraining($P_{i\in N^t}$)

Input : data
Output : θ
1 $t = 1$
2 **while** $t \in 1, 2, .., Ep_{indT}$ **do**
3 Generate Batch size of i.i.d. data
4 Calculate $L(\theta)$
5 Optimise the θ
6 increment(t)
7 **end**

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{\upsilon}_t} + \epsilon}\hat{m}_t \tag{4}$$

where θ_t is the vector of the NN parameters at the previous time step; such that $\theta \in \mathbb{R}^{\mathrm{d}}$, η is the learning rate, $\epsilon = 10^8$ is a smoothing term that avoids division by zero, and $\hat{m}_t$ and $\hat{\upsilon}_t$ are the bias-corrected first and second moment estimates and are calculated by Eqs. (5) and (6), respectively

$$\hat{m}_t = \beta_1 m_{t-1} + (1-\beta_1)g_t 1 - \beta_1 \tag{5}$$

$$\hat{\upsilon}_t = \beta_2 \upsilon_{t-1} + (1-\beta_2)g_t^2 1 - \beta_2 \tag{6}$$

such that g_t is the gradient of the objective function with respect to θ, while m_t and υ_t are estimates of the first moment (the mean) and the second moment (the un-centred variance) of the gradients, respectively, and $\beta_1 = 0.9$ and $\beta_2 = 0.999$ are the decay rates.

For simplicity, the Tx and Rx AEs are identically structured. The AE used in the design has a relatively low depth and small number of neurons in each layer, where the number of neurons is a function in the number of channel uses as proposed in the NN design proposed in [5]. We set the number of channels used to four, which leads to the number of neurons in the NN model as per the policy-based method in [5]. We describe the difference between the NN design proposed in [5] and our AE in the left and right subfigures of Fig. 2 and we compare the time taken in training in both methods in our performance analysis in Sec. V.

B. End-systems Collaborative Learning

The second learning phase is the collaborative learning phase which enables the Tx and the Rx to overcome the CH noise. Note that initially trained end-systems (Tx and Rx) contribute effectively to the communication process reducing the time needed for the collaborative learning phase to result in a relatively low level of error.

This phase includes the transmission of data along the communication channel where the seeds of the data generators are the same in both ends. This assumption was first introduced in [5] to make sure that the training of Tx and Rx is independent of the CH function, and it was justified by the ability of using the same random number generators at both end-systems (Algorithm 2, step 3). In that case, both the Tx and Rx have the same numbers and start communicating them,

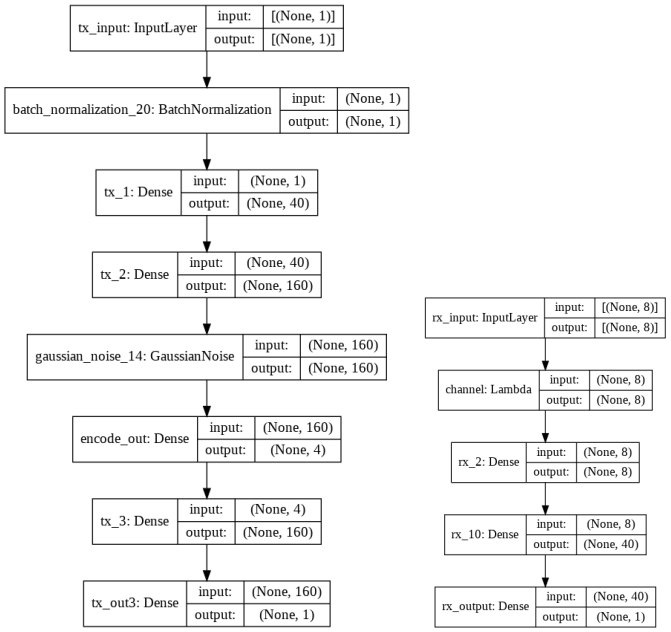


Fig. 2. The proposed Autoencoder model for the Tx which is similar to the one for the Rx (left) Vs the NN design in [5] for the Rx

so the Tx updates its θ_{Tx} as it receives the estimated data at the Rx (z) and the Rx updates its θ_{Rx} as it transmits the data x generated by the Tx (Algorithm 2, steps 4 to 9) as described in Fig. 1. This means that in the presence of F-CH, the loss functions at the Tx and Rx at each time step during the collaborative training time ($t \forall t \in \mathbb{N} = [1, 2, .., Ep_{collT}]$) are modelled as in Eq.s (7) and (8), respectively.

$$\min_{\theta_{Tx}} \quad L(\theta_{Tx}) = \frac{1}{Ep_{collT}} \sum_{t=0}^{t=Ep_{indT}} (z_t - x_t)^2 \qquad (7)$$

$$\min_{\theta_{Rx}} \quad L(\theta_{Rx}) = \frac{1}{Ep_{collT}} \sum_{t=0}^{t=Ep_{indT}} (\hat{y}_t - x_t)^2 \qquad (8)$$

In our collaborative training algorithm, we avoid relying on the assumption that Rx is able to generate the same random numbers during that phase, meaning Rx will not contribute to the training process. Thus, the loss function is only modified at the Tx, as in Eq. (7), while the Rx will have its parameters modified as performed in the previous phase (the independent learning phase), as in Eq. (2). The removal of this assumption makes our algorithm suited also to the difficult situation where Rx is unable to cooperate. This variant of IPCL is called IPCL-no-Rx to highlight that the Rx is not contributing in the collaborative training phase.

Similarly, the presence of feedback channel is not guaranteed; we address this by modifying Tx's loss function to be similar to its independent training process in Eq. (1) while the Rx will keep using its collaborative training function in Eq.(8). This variant of IPCL is called IPCL-no-FCH. Moreover, a highly challenging scenario was also considered in our design, where neither the Rx updates its parameters nor does the Tx receive data from the Rx through F-CH. We refer to this variation of IPCL as IPCL-no-Rx-FCH. This case represents the absence of collaborative training where the time taken in the collaborative training Ep_{collT} represents an extension to the independent training because the training functions for the Tx and Rx as modelled as Eq.s (1) and (2), respectively.

The updates in the NNT and NNR in the first and second training phases of IPCL are shown in Fig.s 3 and 4, respectively, where sub-figure (a) shows the individual training and sub-figure (b) shows the collaborative one. The main differences between these two phases is the ability of each NNT to adapt to what is interpreted at the output of the Rx (see Fig. 3) and the ability of NNR to be trained on the same data generated at the Tx (see Fig. 4).

Algorithm 2: CollaborativeTraining($x, Epochs_{collTrain}$)

Input : x, Ep_{colT}
Output : $\theta^*_{Tx}, \theta^*_{Rx}$

1 $t = 1$
2 **while** $t \in 1, 2, .., Ep_{colT}$ **do**
3 generate x_t then $x' = x_t + \mathcal{N}(\mu, \sigma_{Tx})$ ▷ **At the Tx**
4 $x_t + \mathcal{N}(\mu, \sigma_{Tx}) \xrightarrow{NNT} \hat{x_t}$
5 update θ_{Tx}
6 $\hat{x_t} \xrightarrow{CH} y_t$
7 ▷ **At the Rx**
8 **if** *Rx can use the seed in Tx* **then**
9 generate y_t from the same seed
10 **else**
11 generate y_t from a different seed
12 $y' = y_t + \mathcal{N}(\mu, \sigma_{Rx}) \xrightarrow{NNR} \hat{y_t}$
13 $\hat{y_t} \xrightarrow{F-CH} z_t$
14 update θ_{Rx}
15 ▷ **At the Tx**
16 **if** *F-CH exists* **then**
17 Train the NNT to map the output of the Rx to the actual input (input:z_t output:x_t
18 **else**
19 Train the NNT to map its output to the actual input (input:$\hat{x_t}$ output:x_t)
20 increment(t)
21 **end**
22 Calculate MSE using Eq. 3 where $Ep_{indT} \leftarrow Ep_{colT}$
23 $\theta^*_{Tx} \leftarrow \theta_{Tx}$ and $\theta^*_{Rx} \leftarrow \theta_{Rx}$

IV. Results

This section discusses the analysis of the IPCL framework, the description of the testing scenarios, and the results obtained.

A. Experimental Setup

We compared the work in [5], denoted as "policy" method, to seven methods which include three of its variations, IPCL, and three of IPCL's variations. These variations allow us to test the method in the literature and our proposed framework with different system assumptions to investigate the effect of each component of these methods on its ability to minimise the error in the estimation of the message at the Rx. The methods

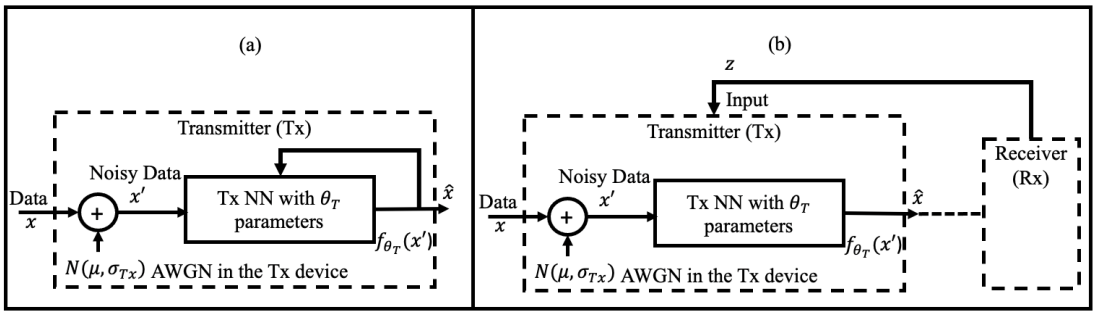


Fig. 3. Tx training process (a) independently, and (b) collaboratively

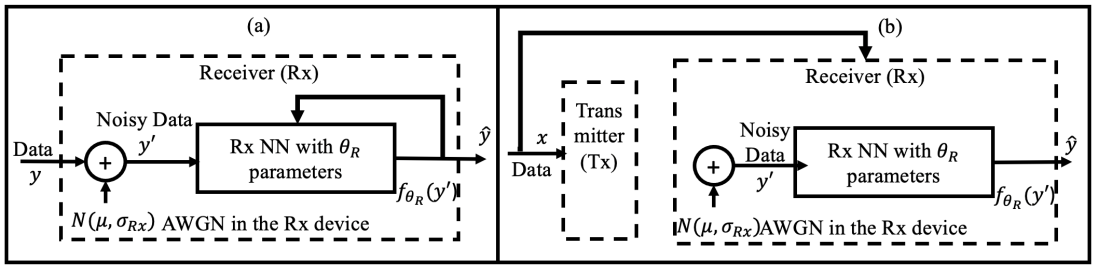


Fig. 4. Rx training process (a) independently, and (b) collaboratively

we used in our comparisons, as well as the components we change in each method accompanied by the justification of the changes are summarised as follows:

- The "policy" method which is an RL-based method proposed in [5] with perturbation variance equals 10^{-4}.
- The "policy-trained" method which is the policy method [5] with independent training phase on both end-systems (the Tx and Rx) in order to investigate the effect of the independent training phase on the existing design.
- The "policy-AE-Tx" method is the "policy-trained" but utilising our proposed AE as the Tx NN in order to investigate the effect of asymmetric NN design at the communication end-systems in the absence of independent training phase.
- The "policy-AE-trained-Tx" method is the "policy-AE-Tx" but the independent training phase is deployed on both the Tx and Rx in order to investigate the effect of independent training phase on asymmetric end-systems .
- The proposed "IPCL" framework with both training phases, and the assumptions in [5] which include the ability of the Tx to receive feedback from the Rx through the F-CH and the Rx to contribute to the collaborative training phase; the Gaussian noise variance in AE is 0.1.
- The "IPCL-no-FCH" method is the IPCL, but without the availability of an F-CH during the collaborative learning phase. This leads to inability of the Tx to receive data from the Rx during the collaborative training phase. This enables investigation of the effect of F-CH during the collaborative training phase.
- The "IPCL-no-FCH-Rx" method is the "IPCL-no-FCH" but removing the assumption that the Rx is able to generate the same data as the Tx during the collaborative learning phase or contribute to the collaborative training process. This helps us investigate the worst possible scenario where the Tx cannot adapt to the Rx while the Rx is not able to contribute to the collaborative training phase, which leads to only further training for the Tx.
- The "IPCL-no-Rx" method is a variation of IPCL including the assumption that an F-CH exists, but without updates in the Rx in order to investigate the effect of Rx training during the collaborative training phase but without the ability of the NN to update its weights according to the received data at the Rx.

All methods were tested on the same testing scenario for 5000 epochs where a message of data x is sent from Tx to Rx in one epoch, and a feedback message may pass from the Rx to the Tx during the same epoch if an F-CH exists. Each message is a set of independent identically distributed (i.i.d.) randomly generated numbers of a Batch size (B) numbers ranging from zero to one. We assume that the Tx and Rx are symmetric as well as the forward and feedback channels for simplicity. This setting allows us to analyse the effect of the NN design and the training framework proposed on minimising the MSE at the receiver end in the presence of noise at the end-systems and channels. The system parameters are presented in table I.

To analyse the IPCL framework in terms of the number of epochs required for collaborative training that is sufficient for minimising MSE (Eq. (3)), we run a testing scenario for different number of epochs in the collaborative training phase. Note, the collaborative training time is the initialisation time for Tx and Rx, which represents the overhead in the communication process; the shorter that time, the more

adaptive the end-systems are to channel variations and the more efficient the communication process is (assuming error is kept to minimum).

TABLE I. SYSTEM ASSUMPTIONS

Parameter	Value
Epochs in independent training	100
Epochs in collaborative training	$[10, 100, 1000]$
Epochs in testing	5000
Batch size (B)	1024
SNR at the end-system	$SNR =$ $[-4, -2, 0, 2, 4, 10^6]$
SNR at the channels	$SNR = [4, 10, 40, 10^6]$

B. Comparative Analysis

We consider three different lengths of the collaborative training phase (10, 100 and 1000 Epochs) in order to investigate the effect of the length of the collaborative training phase on the ability of Rx to reduce the effect of noise on the data transmitted by Tx. The level of noise is defined by the signal to noise ratio (SNR) in decibel (dB) unit as defined in [20] and formulated in Eq.(9). The communication channel as well as the end systems are assumed to add AWGN noise to the transmitted data as described in Sec.II and illustrated in Fig.1.

We consider three different lengths of collaborative training phase (10, 100 and 1000 Epochs) in order to investigate the effect of the length of the collaborative training phase on the ability of the Rx to reduce the effect of noise on the data transmitted by the Tx. The level of noise is defined by the signal to noise ratio (SNR) in decibels (dB) as defined in the literature [20] and formulated in Eq.(9). The communication channel as well as the end systems are assumed to add AWGN noise to the transmitted data as described in Sec.II and illustrated in Fig.1

$$SNR(dB) = 10 \log_{10} \frac{P_{signal}}{P_{noise}} \tag{9}$$

where the signal and noise powers are P_{signal} and P_{noise}, respectively.

We start our analysis by testing all the models when the noises at the end-systems and the channel are negligible $SNR = 10^6$. The results showed that our AE converged to a small level of error at least five times faster than the NN model in [5] during the independent training phase, as shown in the top-right sub-figure in Fig. 5. Moreover, in the first 100 collaborative training epochs, the initially trained Tx variant of the policy method [5] (policy-trained) shows worse progress while its asymmetric variant (policy-AE-trained-Tx) seems to converge faster when the Tx is a pre-trained AE but no progress when independent training was not performed (policy-AE-Tx). However, through the 1000 epochs of the collaborative training time, policy-AE-trained-Tx degrades after 100 epochs, and policy-AE-Tx shows no improvement until the end of the 1000 epochs.

However, through the 1000 epochs of the collaborative training time, policy-trained degrades after 100 epochs, presumably because the policy-based method uses the loss to train the Tx during the collaborative training phase which has very small values (in case of noiseless channel) compared to that of the actual data experienced during the independent training phase. Policy-AE-Tx shows no improvement until the end of the 1000 epochs because the Tx and Rx have different designs so the transmitter and receiver face an instability problem in training due to the use of the very small error values in learning and the asymmetric design of their NNs.

For IPCL-no-FCH, the MSE is almost zero from beginning of the training phase to its end because the Tx is not able to get any information from the Rx while the NNR is already trained to give a small error independently, so the collaborative training leads to neither improvements nor degradation at both ends. The other variations of IPCL show relatively fast convergence compared to the those based on the policy method within 10 as well as 100 epochs. However, by the end of the long training time (1000 epochs), only policy-trained and policy-AE-Tx have high MSE as in Fig. 5(c).

The MSE of the communication system with noiseless channels was tested for 5000 epochs when the number of epochs in the collaborative training phase is 10, 100, and 100. The results in Fig.6 show the averages MSE ± 1 standard deviation. The MSE when the end-systems are noiseless Fig.6(a) decreases and almost vanishes due to the long collaborative training phase (1000 epochs) for all the tested methods except the modified versions of the policy method [5]. Whereas the shortest training time (10 epochs) is enough only for the IPCL, IPCL-no-FCH and policy-AE-trained-Tx to achieve unnoticeable MSE (0.002, 0.002 and 0.007) while the no contribution at the Rx side made IPCL-no-Rx have the worst performance in 10 and 100 epochs compared to all its peers. Moreover, the asymmetry of policy-AE-trained-Tx lead to its instability, which resulted in a higher MSE despite the increase in the training time. All the methods show improvement as the training time increased from 10 to 100, but the policy method shows the best level of improvement.

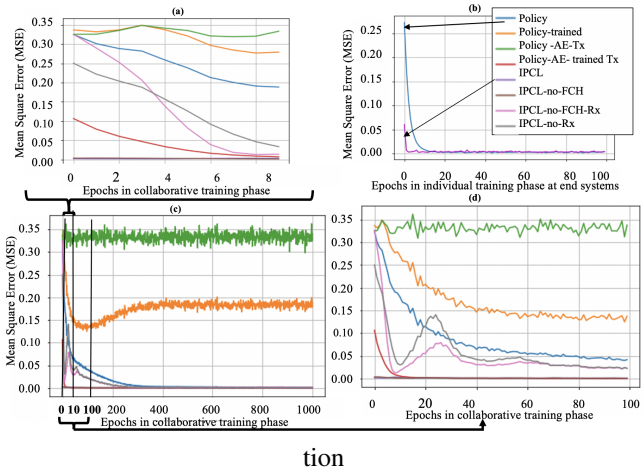
In Fig.6 (b) and (c), the MSE is shown at different noise levels at the end systems $SNR = [-4, 4]$ when the collaborative training time is 10 and 100 epochs, respectively. By comparing sub-figures (a) and (b), a noticeable improvement can be observed for all the variations of the policy method [5] as the training time increased. However, policy-AE-trained-Tx shows unexpected behaviour by degrading as the training time increases and the noise level decreases at $SNR = [0, 2, 4]$. It is worth to note that IPCL and its variants make little improvement as the training time increases despite having IPCL and IPCL-no-FCH with the least MSE in both cases (suggesting convergence is reached quickly). IPCL generally performs better than the policy method [5]. This is somewhat expected as *policy* was predominantly designed to overcome the noise at the communication channel not the end-systems.

Therefore, we performed a testing scenario when only the channels are noisy ($SNR = 40, 10, 4$) while the end-systems are noiseless to fairly compare the policy method [5] to IPCL, as in Fig.7. The policy method [5] outperforms all the other methods only when the collaborative training time is long

(1000 Epochs) in all the cases of the channel noise reaching (0.0005) error. However, after a shorter learning time (100 epochs), it shows better performance than all its variants, except the policy-trained method that performs better in a high SNR case ($SNR = 40$). This could be due to the design objective of the policy method [5] that was to solve low SNR cases. This proved that the performance of the policy method [5] is not guaranteed in all cases of SNR, especially in limited training time. In contrast, IPCL and IPCL-no-FCH show almost the same performance in the three SNR channels, especially when SNR is high, but they both take a shorter time to converge to lower MSE as SNR decreases, despite the better learning rate of IPCL. Nevertheless, IPCL and all its variants couldn't reach the very low MSE achieved by the policy method [5] when SNR was relatively low (10 and 4). Moreover, IPCL-no-Rx seems to take a relatively long time compared to all its variations to converge to a considerably low MSE due to the un-cooperative behaviour at the Rx, which makes the Tx adapt to its own noise, the channel noise and the Rx's noise individually, while IPCL-no-FCH-no-Rx seem to struggle in learning as SNR decreases.

To investigate the performance of all the methods in comparison in noisy channel as well as end-systems scenarios, we test all the models on channel noise $SNR = 4$ and different end-systems noise levels $SNR = [-4, 4]$ when the collaborative phase for 10 and 100 epochs. Fig. 8 shows the training performance through collaborative training time when SNR at the end-systems is -4 sub-figure (a) while the MSE of each method tested on each SNR value at the end-systems when the training took 10 epochs and 100 epochs are shown in subfigure (b) and (c). The training progress of the policy-AE-trained-Tx fluctuates through time while all the other variations of the policy method [5] converge relatively fast, with policy-AE-Tx showing the highest convergence rate. In contrast, IPCL show almost the same small error level throughout all the training time while IPCL-no-FCH-no-RX degrading over time.

The performance in training when $SNR = -4$ at the end systems (Fig. 8(a)) affects the results of all the methods when tested after 10 epochs (Fig. 8(b)) or after 100 epoch (Fig. 8(c)). The performance of IPCL and its variants show more than 50% reduction in the MSE compared to the policy method and all its variants, except for policy-AE-trained-Tx that shows noticeable improvement as SNR at the end-systems increases (see Fig. 8(b)). The improvement of policy-AE-trained-Tx and policy-AE in 10 epochs as the SNR at the end systems increase is because of the ability of the AE to get over internal noise in a short time when SNR is relatively high. IPCL and all its variations converge to a relatively low MSE in a very short time due to the use of AE in the NN design and the individual training phase. The long collaborative training phase (100 epochs) allowed all the methods with symmetric NN design at the end-systems to converge to relatively low MSE (see Fig. 8(c)), while the policy-AE-Tx and the policy-AE-trained-Tx methods show improved performance as the SNR at the end-systems increased.



tion

Fig. 5. The MSE of noiseless end-systems over noiseless channels with collaborative training phase $Ep_{colT} = 1000$ in sub-figure (c), the progress in the first 10 and 100 epochs are shown in sub-figures (a) and (d), respectively, while sub-figure (b) shows the progress of the independent training at the end-systems using the NN model designed in [5] Vs the proposed AE

V. Conclusion and Future Work

In dynamic systems where the communication channel varies with time, adaptive Tx and Rx should be trained in the shortest possible number of transmissions that optimises their NN parameters to be able to achieve low mean square error (MSE) before exchanging messages. This paper addressed the problem of long initialisation time at the Tx and Rx ends in order to overcome noise at both the end-systems and the communication channel. The noise model discussed in this paper is the additive white gaussian noise (AWGN) that is common in electronic devices as well as communication channels. IPCL framework for end-systems' learning was introduced to reduce the effect of noise on the data interpreted at the receiver (Rx) as measured by MSE.

IPCL was tested in different communication system scenarios that include the absence of feedback channel (F-CH), the inability of the Rx to cooperate in the collaborative learning phase and in both cases combined. The IPCL showed better performance without the need for modelling the channel compared to its peer in a very short learning time. The analysis performed in different communication scenarios show that IPCL converges quickly. Further improvements to allow for continual improvement through training time is considered as an extension to this work. Moreover, consideration of different types of channel noise models is recommended for further investigation of the IPCL's performance.

References

[1] D. Tse and P. Viswanath, "The wireless channel," in *Fundamentals of Wireless Communication*, 2012, pp. 10–48.

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

[3] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *arXiv*, vol. 3, no. 4, pp. 563–575, 2017.

[4] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of Deep Reinforcement Learning in Communications and Networking: A Survey," *IEEE Communications Surveys Tutorials*, vol. PP, no. c, pp. 1–1, 2019.

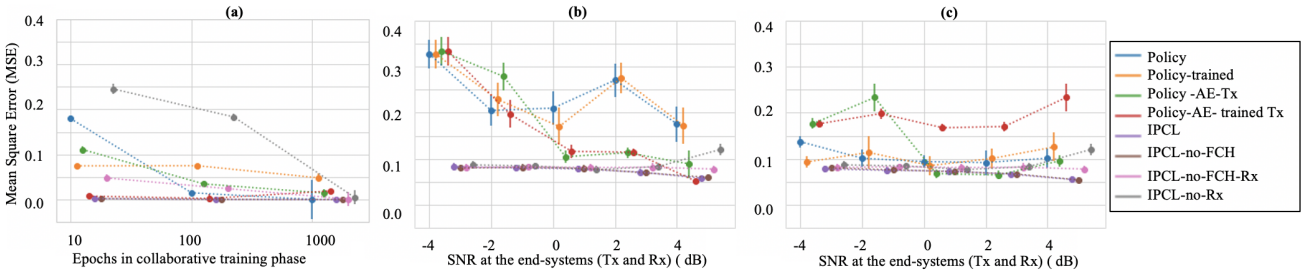


Fig. 6. The MSE for noiseless channels and (a) noiseless end-systems trained at $Ep_{colT} = 10, 100, 1000$ and end-systems with different SNRs for (b)$Ep_{colT} = 10$ and (c)$Ep_{colT} = 100$

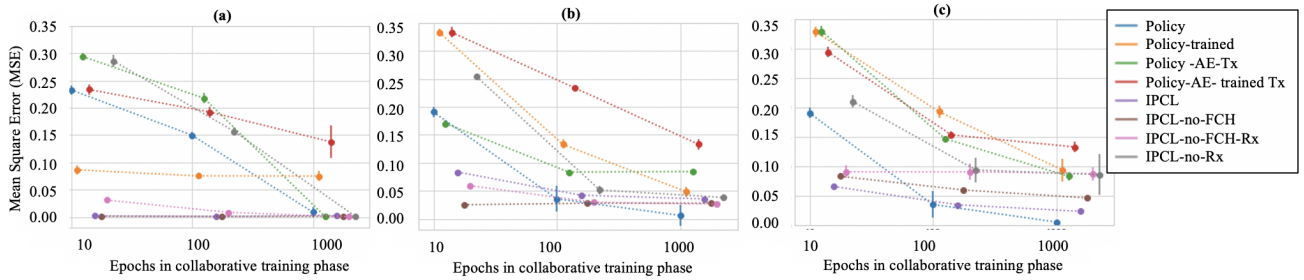


Fig. 7. The MSE for noiseless end-systems trained collaboratively for $Ep_{colT} = [10, 100, 1000]$ and the channel noises are $SNR = [4, 10, 40]$

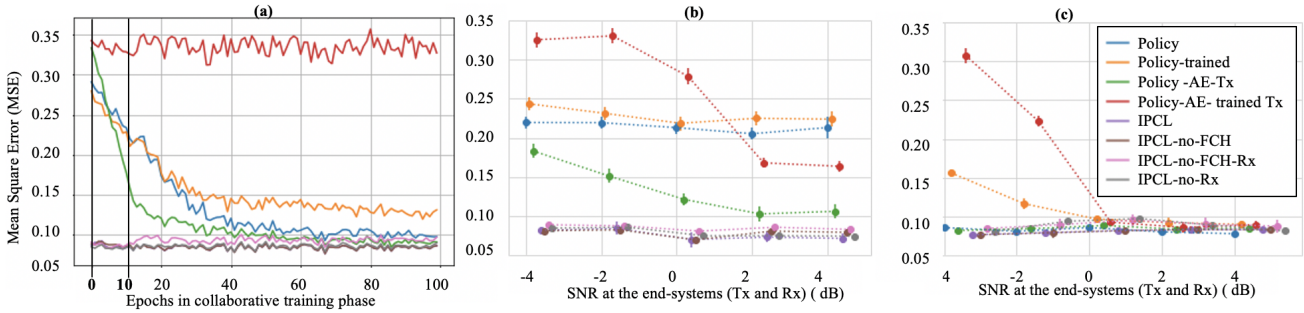


Fig. 8. The MSE for channels with $SNR = 4$ (a) through collaborative training phase of 100 epochs where the horizontal line shows the MSE after the first 10 epochs, and for different SNRs at the end-systems after (b) 10 epochs and (c) 100 epochs of collaborative training phase

[5] M. Goutay, F. A. Aoudia, and J. Hoydis, "Deep Reinforcement Learning Autoencoder with Noisy Feedback," in *International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOPT)*. Avignon, France: IEEE, 2020, pp. 1–6.

[6] L. Bottou, "Stochastic learning," in *Advanced Lectures on Machine Learning*, ser. Lecture Notes in Artificial Intelligence, LNAI 3176, O. Bousquet and U. von Luxburg, Eds. Berlin: Springer Verlag, 2004, pp. 146–168.

[7] P. Gallinari, Y. Lecun, S. Thiria, and F. Fogelman Soulie, "Memoires associatives distribuees: Une comparaison (distributed associative memories: A comparison)," in *Proceedings of COGNITIVA 87, Paris, La Villette, May 1987*. Cesta-Afcet, 1987.

[8] J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proceedings of the National Academy of Science*, vol. 79, no. 8, pp. 2554–2558, Apr. 1982.

[9] G. E. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, pp. 504 – 507, 2006.

[10] A. V. Le, V. Prabakaran, V. Sivanantham, and R. E. Mohan, "Modified a-star algorithm for efficient coverage path planning in tetris inspired self-reconfigurable robot with integrated laser sensor," *Sensors (Switzerland)*, vol. 18, no. 8, 2018.

[11] P. Vincent, H. Larochelle, Y. Bengio, and P. A. Manzagol, "Extracting and composing robust features with denoising autoencoders," *Proceedings of the 25th International Conference on Machine Learning*, no. January, pp. 1096–1103, 2008.

[12] S. Park, O. Simeone, and J. Kang, in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*.

[13] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 6848–6856, 2018.

[14] N. Efremova, M. Patkin, and D. Sokolov, "Face and emotion recognition with neural networks on mobile devices: Practical implementation on different platforms," *Proceedings - 14th IEEE International Conference on Automatic Face and Gesture Recognition, FG 2019*, no. 1, 2019.

[15] M. Soltani, V. Pourahmadi, A. Mirzaei, and H. Sheikhzadeh, "Deep Learning-Based Channel Estimation," *IEEE Communications Letters*, vol. 23, no. 4, pp. 652–655, 2019.

[16] S. Dörner, S. Cammerer, J. Hoydis, and S. t. Brink, "Deep learning based communication over the air," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 132–143, 2018.

[17] L. Yasenko, Y. Klyatchenko, and O. Tarasenko-Klyatchenko, "Image noise reduction by denoising autoencoder," in *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, 2020, pp. 351–355.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

[19] S. Ruder, "An overview of gradient descent optimization algorithms," pp. 1–14, 2016. [Online]. Available: http://arxiv.org/abs/1609.04747

[20] Y. Lai, "Taking the Mystery out of the Infamous Formula," SNR= 6.02 N+ 1.76 dB," and Why You Should Care," *Imid 2009*, pp. 1069–1072.