Distributed Learning of Universal Receivers Over Fading Channels in IoT Communications

Mark Powell
Department of Computer Science and Electrical Engineering
Queen Mary, University of London
London, United Kingdom
ec201080@qmul.ac.uk

***Abstract*—Within the field of channel estimation and signal detection, deep learning has shown great promise in training low complexity, flexible receivers, that are robust to varied channel conditions. For the most part this is done using simulated data, or with data collected and then transmitted back to the base station (BS) for centralized training. It is preferable to train using real world data over simulated data, but transmitting datasets to a BS may be impractical and create large communication overheads. Distributed learning methods provide a way to train globally optimized networks using local device datasets, leaving the BS model agnostic to the raw data. This paper shows a comparison of the two main distributed learning methods, split learning and federated learning, when applied to training DNN receivers over fading channels. In a simple single input single output (SISO) system, split learning is compared to the previously proposed FedRec receiver model. Both split learning and the FedRec receiver approach the performance of statistical model based techniques, but due to the simplicity of the model, split learning fails to improve performance while increasing communications overhead. In the case of orthogonal frequency division multiplexing system (OFDM), both Federated and split learning are compared to a centralized model. They both approach the performance of the offline trained model. In this instance, the input data to the model is more complex, therefore a more highly parametrized model is needed. Even with this more complex model, federated learning has a smaller communications overhead and is therefore the preferable model.**

I. Introduction

In the current age of the internet of things (IoT) and 5G technologies, there are more and more devices online requiring extremely fast data transfer across wireless channels. This need for mass high speed data has brought challenges to current communication approaches, where the increasingly complex techniques utilised, have been seen to degrade the performance of conventional communications systems [1]. As a response to these challenges, the application of deep neural networks (DNN) in the communications field has gained much interest from researchers in recent years.

This paper will focus on the application of DNNs for channel estimation and symbol detection in fading channels. The need for channel estimation arises from the fact that any digital communication system transmits digital bits from one place to another through some kind of physical medium. For transmission, bits must be encoded on a carrier signal. This encoded data must then be decoded correctly at the receiver so that there are minimal errors in the received data. This is a trivial task if the transmitted signal is unchanged between the transmitter and receiver. Unfortunately, the physical space the signal must travel through, known as the channel will cause attenuations in the power of the received signal. This attenuation is known as fading and is due to scattering by objects in the line of sight between transmitter and receiver.

Channel estimation is often achieved using pilot symbols. Pilot symbols are known symbols that are transmitted to the receiver prior to transmitting the actual data symbols for each coherence duration. For channels with very short coherence durations, this can lead to considerable overhead, as new pilot symbols must be transmitted very frequently. Often the complex large scale systems used in 5G systems lead to communications links with rapidly changing channel conditions [1]. This leads to fast fading, where the coherence time is small compared to the data rate. In this instance, a statical model can be utilised to form a detection rule that accounts for all channel conditions [2]. Statistical models tend to approximate certain specific channel conditions quite well, but the performance will drop when applied to a channel that does not conform to the specific channel's fading conditions. Deep learning techniques have the potential to learn complex receivers that can adapt to the challenging channel conditions in 5G systems, with no reliance on knowledge of complex models.

A. Related work

The current work surrounding channel estimation and signal detection using DNNs is discussed. DNN channel estimation in an orthogonal frequency division multiplexing (OFDM) system is explored in [3]. This is done by viewing the time-frequency response of the pilot symbols in the OFDM frame as a sparse matrix. The channel is estimated at the pilot positions and a convolutional neural net (CNN) is used to accurately interpolate the rest of the values for the frame. The pilot values are considered as a low resolution image and a general deep learning image reconstruction pipeline using deep image processing techniques is used to estimate the channel. CNNs have also widely been used for channel estimation in multiple input multiple output (MIMO) systems. In this instance, the input to the CNNs are the pilot symbols. The real and imaginary parts of the symbol values are arranged into a set of channels, with the absolute value of the complex

baseband symbol making up a third channel [5] and the angle of the complex number [4]. The CNNs are trained using simulated channel data based on statistical models, allowing the CNN's parameters to be optimized with respect to the simulated channel matrices.

Other papers explore symbol detection directly from the received signal. [6] explores both symbol by symbol detection and also sequence detection directly from received pilot symbols. During symbol by symbol detection, the network is a classifier that takes some features of the received signal and uses fully connected layers with a soft max layer to predict the most likely sent symbol. Treating the incoming signals as a data stream allows for the task to be framed as sequence detection. For sequential classification tasks, such as within natural language processing, recurrent neural networks (RNN) have been shown to be extremely effective. RNNs learn an internal hidden state from the training sequences, which is an abstract vector learned from the data sequence throughout the previous time steps. This hidden state acts as a summary of the training sequences, so if trained on sufficient data, during deployment, accurate data sequences can be predicted. RNNs are useful in sequence detection. This is because inter symbol interference (ISI) causes adjacent symbols to carry information about each other and the hidden state memory can account for this information leakage due to ISI. [6] proposes a bidirectional RNN, where the symbol sequence is input forwards to one hidden state and backwards to another and the two merged. This allows for a hidden state that not only takes into account past symbols but future ones too.

Papers such as [7] propose end to end deep learning communications systems. By modelling the transmitter, receiver and channel and training as an auto-encoder, the transmitter and receiver implementations are jointly optimised from data. The auto-encoder term refers to the systems ability to learn an intermediate representation of the input symbol that is robust to the channel distortion. The encoder and decoder are jointly optimised to accurately recover the symbols, where a noise layer is applied between the encoder and decoder to simulate channel distortions. [7] explores end to end auto-encoders for SISO, [8] extends this idea to an OFDM system and [9] to MIMO.

B. Motivations

In many of the above applications of machine learning (ML) in communications the models are trained offline using simulated data. In practice, it is preferable to collect accurate real world data from user devices to use in training, but if a model is to be trained offline, then huge training datasets must be transmitted back to a base station after collection. A recent advancement in the ML sphere has been the advent of distributed learning techniques, as a direct response to the growing privacy concerns and ethical dilemmas when collecting personal data for training machine learning models. Federated learning is one of these techniques and uses the federated averaging of user models trained on local device datasets to train a globally optimized model [10]. This allows the training of machine learning models that leave the BS model agnostic with regards to the raw data. As a communications system is almost always made up of multiple user devices transmitting to and from a BS, this is an ideal scenario for the implementation of distributed learning. The privacy advantages of distributed learning techniques are important here as well, as the symbols used for training may contain sensitive information from the user devices. In [11] federated learning is applied in MIMO channel estimation. In this instance, the received pilot symbols and the estimated minimum mean squared error channel matrix for a transmission block are taken as the input data and labels, respectively, of one sample in the dataset. Over many transmission blocks, these samples are stored on each device to form a local dataset, allowing a CNN that estimates the channel based on the pilots symbols in a transmission block to be trained across all devices.

As in any machine learning problem, to train a model that can be generalised and does not over-fit to any specific channel conditions, the model must be trained on data that spans a diverse range of channel realizations. Therefore, one main motivation to using distributed learning techniques in symbol detection is the fact that multiple user devices can collect training data in diverse channel conditions. If this training data contains symbols distorted by diverse enough channel conditions, the distributed DNN should be a universal receiver for the channel, regardless of the conditions at that moment. The paper [2] proposes using federated learning to train a universal receiver, termed FedRec. In this context, federated learning allows for the DNN to be trained on data collected by users in diverse channel conditions, so a data driven detector can be jointly learned over all devices within the communication range of the transmitter.

C. Contributions

The first contribution of the paper will be to implement an online trained universal receiver using a state of the art technique, known as split learning. This technique splits the network at a cut layer between the BS and devices. The cut layer activations and gradients are sent between devices and the BS during a forward and backward propagation [12]. Split learning's main advantage over federated learning is the fact that most of the training computations can be completed on the server side, meaning more complex networks can be trained on lower power user devices more efficiently. Another big draw to exploring split learning as a technique, is the fact that only the cut layer needs to to be transmitted, rather than the gradient updates for all parameters, leading to potential communication efficiency improvements in some cases, over federated learning.

Using a modified version of an algorithm from [12], and applying it to symbol detection in fading channels, a universal receiver using split learning is designed. The channel is modelled in the same way as [2], where fast fading is assumed and the transmitted symbols are distorted by values drawn from a simple statistical model. After training the network, the transmitted symbols are recovered from the received symbols

with no knowledge of the underlying statistical fading model. By finding the bit error rate (BER) at different signal to noise values and with variable users, the performance and communication overhead of split learning can be compared to FedRec, and the feasibility of the system investigated.

The idea of using an online trained receiver for symbol detection will then be extended to a more complex OFDM system. These systems use orthogonal sub-carrier signals, each carrying a data symbol to greatly increase the data carrying capacity of a communication system. The initial model for this symbol detector comes from [13], where pilot symbols and data carrying symbols affected by simulated channel distortions are input into DNN. This model aims to implicitly estimate the channel and recover the sent symbols, directly from received symbols. This ML based joint signal detection/channel estimator will be implemented using federated and split learning algorithms. By comparing to the offline trained model, the feasibility of training an OFDM detector on user devices will be investigated.

In general, the goal of this paper is to assess the performance of various distributed learning techniques in DNN signal detection, first in a SISO system and then OFDM. In these contexts, the channel is implicitly estimated and sent symbols recovered using models trained on received symbols exposed to the varied channel conditions of multiple user devices.

The paper is organised as follows: section II will cover preliminary topics, section III will outline the system models, section IV will cover the methodology of optimizing the online learning models, section V will cover the simulation results and conclusions will be drawn in section VI.

II. Preliminaries

A. Multipath fading

Multipath fading over a wireless channel is caused by scattering due to obstructions in the path between the transmitter and receiver. The receiver antenna receives N paths of the transmitted signal, each with a different phase ϕ_n, amplitude A_n and delay τ_n. The signal at the receiver is a superposition of these paths and this superposition causes random fluctuations of the signal amplitude. For transmitting a generic Dirac impulse $\delta(t)$, the impulse response of the channel $h(t)$ is modelled as

$$h(t) = \sum_{n=0}^{N-1} A_n e^{j\phi_n} \delta(t - \tau_n). \tag{1}$$

Fig. 1 shows a physical representation of this model of multipath propagation. The delay spread of a channel is the difference in the time of arrival of the first and last detectable path of the multipath signal.

Statistical models are often used to model the randomness of this channel coefficient $h(t)$. The attenuation of the transmitted signal due to (1) is modelled as a random variable taken from a known statistical distribution [2]. The Rayleigh fading

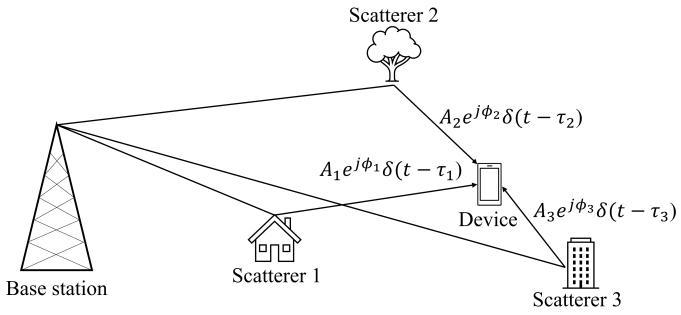


Fig. 1: Simple multipath fading channel with 3 scatterers, showing the multiple paths indecent on the user device

distribution is one of the most common statistical channel models. It assumes there is no dominant line of sight between the transmitter and receiver, so the attenuation of the received signal due to the many scattering objects is modelled as a zero mean Gaussian random variable.

In this paper, flat fading coefficients are considered for channel modelling. This is related to the coherence bandwidth of a channel, this is the range of frequencies over which the channel response is consistent. The frequency response of a channel is flat if the the coherence bandwidth of that channel is larger than the passband bandwidth. During frequency selective fading, the fading varies across the bandwidth of the transmitted signal. As the symbol period is inversely proportional to bandwidth, the higher the symbol rate the more likely a channel will have frequency selective response. In this paper, it is assumed that the bandwidth of the transmitted signals are narrow enough for flat fading.

B. Modulation

In wireless communications the data is transmitted by altering the features (amplitude, phase, frequency) of a high frequency carrier sinusoid, this is known as modulation. This carrier sinusoid is demodulated at the receiver and the transmitted data retrieved.

In digital communications the data to be transmitted is binary. Often multiple bits are represented by a single modulation, known as a symbol. Phase shift keying (PSK) is a technique where symbols are encoded onto the carrier signal as distinct phase shifts. There must be 2^Q different phases in an PSK system to transmit Q bits per symbol. PSK is a special case of a more general modulation scheme, quadrature amplitude modulation (QAM). Where in PSK the amplitude of the transmitted wave is fixed and only the phase is varied to encode the symbols in the carrier wave, in QAM the phase and amplitude varies between symbols. QAM is achieved by modulating the amplitudes of two orthogonal carrier waves, the in-phase (I) and quadrature (Q) components, with two separate bit streams. These I and Q components are added together for transmission and the superposition of the relative amplitudes cause phase and amplitude shifts in the transmitted signal. QAM symbols are represented by complex numbers, where

the real part refers to the amplitude of the I component and imaginary part the amplitude of the Q component.

As they can be represented by two values, QAM symbols can be plotted in a 2d constellation plot. Considering AWGN at the receiver, the constellation points will spread out as the I and Q amplitudes measured at the receiver are affected by this noise. Points from different symbols that overlap due to this spread, will cause symbols errors at the receiver. The channel will also cause distortion of the constellation points. This is where channel estimation, using pilot symbols, is used to detect these distortion affects at the receiver, so the sent symbols can be recovered with minimal errors.

III. System Models

A. SISO

This section is adapted from [2]. In this model, a single base station antenna communicates with a set of users $\mathcal{U}$ indexed by u. Each user transmits a continuous base band signal $\boldsymbol{x}(t) \in \mathbb{C}$ at time t that is input into a channel with flat fading coefficients $h_u(t)$. White Gaussian noise $w_u(t)$ that is independently identically distributed (i.i.d) across users is added to the signal at the receiver, giving the received channel output at user u as

$$\boldsymbol{r}_u(t) = h_u(t)\boldsymbol{x}(t) + w_u(t). \tag{2}$$

As fast fading is assumed, the coefficients $h_u(t)$ are drawn from a probability distribution for each transmitted symbol. The distributions can be i.i.d or non-i.i.d across users, depending on the specific channel conditions. Unlike classical model-based channel estimation methods, the data-driven detector assumes no knowledge of this distribution and implicitly estimates it from the channel input-output relationship in (2).

During downlink transmission, the base station encodes a symbol s_u that is uniformly distributed over $\mathcal{M} \triangleq \{1, ...M\}$ into a signal of duration T_s seconds commencing at t_d. This signal modulated with symbol s_u is denoted by $\boldsymbol{x}_{s_u}(t)$, where $t \in t_d + [0, T_s)$. The base station transmissions consist of pilot and data carrying symbols. These known pilot symbols contain the input-output relationship essential for the data driven detection. N_T pilot symbols are transmitted at $t = 0$ and form a set of symbols $\{s_i^p\}_{i=0}^{N_T}$. This set of pilot symbols along with the received signals $\boldsymbol{r}_u(t)$ for each pilot, form the local dataset for each user, where the channel outputs $\boldsymbol{r}_u(t)$ are the input to the DNN and $s_u(t)$ the labels. The local datasets on each user are, $D_u = \{s_i^p, \mathcal{R}_i^u\}_{i=1}^{N_T}$, where $\mathcal{R}_i^u$ is the pilot $\boldsymbol{r}_u(t)$ received at each pilot interval N_T.

The received signals $\boldsymbol{r}_u(t)$ are high frequency modulated carrier signals that have been transmitted across that channel. Sampling this signal, for example with Nyquist rate sampling to avoid aliasing, would form a high dimensional dataset of discrete time sinusoids. Often in ML tasks, high dimensional inputs can be handled by DNNs, where automatic feature extraction is part of the optimization of the network. This requires networks with millions of parameters, which for online training would be compute and communication expensive. Alternatively, in [2], domain knowledge is used for feature extraction. This knowledge exploits the use of matched filters in demodulation. During QAM demodulation, I and Q oscillators extract their respective components from $\boldsymbol{r}_u(t)$. Then, by sampling the output of a matched filter, the amplitudes of the I and Q values are retrieved. So, to significantly reduce model complexity, it is proposed that the input vectors to the model are the I and Q amplitudes of the received QAM symbols. The labels are the corresponding symbols from $\mathcal{M}$.

B. OFDM

Adapted from [13]. This model inserts a trained receiver to replace the conventional channel estimation and equalization stage in OFDM. To create a conventional OFDM symbol each orthogonal sub-carrier is modulated with a symbol s_u. These baseband complex symbols are input into an inverse discrete Fourier transform. This gives the transmitted signal in the time domain $\boldsymbol{x}(n)$. Although the spreading of symbols across orthogonal sub-carriers increases the effective symbol duration T_s, reducing the ISI, there is still delay spread in a multipath channel, so ISI must be mitigated in some way. This is done using a cyclic prefix (CP), where the last n discrete values in the transmitted signal are appended to the start of the signal. For effective ISI mitigation, n must not be smaller than the maximum delay spread of the channel.

In this instance, the channel is modelled as a sample spaced multipath channel. Sample spaced means the impulse response of the channel is split into a finite number of bins, so some multipath components may arrive in the same time bin, where the magnitude and phase shift of these paths are summed. When using this channel model, random complex vectors $\boldsymbol{h}_u(n)$ describe the channel at a user device. Therefore, the received signal at a device u is given by

$$\boldsymbol{r}_u(n) = \boldsymbol{x}(n) \otimes \boldsymbol{h}_u(n) + \boldsymbol{w}_u(n), \tag{3}$$

where $\boldsymbol{w}_u(n)$ is AWGN i.i.d across devices and $\otimes$ is a circular convolution due to the cyclic prefix. At the receiver, to recover the symbols on each sub-carrier, the CP is removed and the discrete Fourier transform is performed on $\boldsymbol{r}_u(n)$ to give

$$\boldsymbol{R}_u(k) = \boldsymbol{X}(k)\boldsymbol{H}_u(k) + \boldsymbol{W}_u(k), \tag{4}$$

where capital letters denote the frequency domain variables.

Unlike during SISO transmissions, the input to the DNN is not a single baseband symbol but all of the received symbols in an OFDM frame. These frames are made up of two OFDM symbols, the first containing N_T pilot symbols interspersed with data symbols, the second consisting solely of data carrying symbols. The pilot symbols are kept consistent throughout training and deployment and any data carrying symbols s_u are uniformly distributed over $\mathcal{M} \triangleq \{1, ...M\}$. With flat fading, the channel impulse response $\boldsymbol{h}_u(n)$ is assumed constant over the whole frame, but varies between frames. The OFDM symbols in each frame are perturbed by a channel vector taken from a statistically generated set, modelled on a typical urban delay profile. The model takes a vector of the I and Q amplitudes of all the sub-carrier symbols

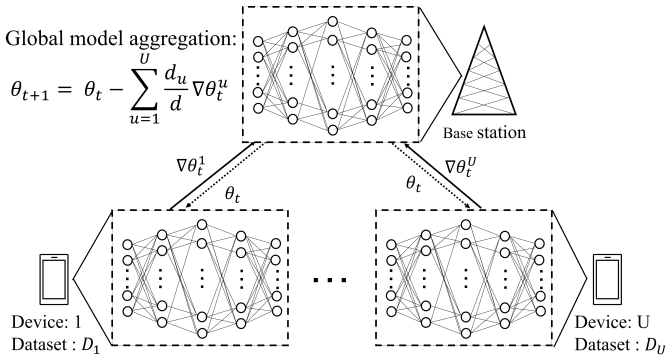


Fig. 2: Schematic diagram for a federated learning system

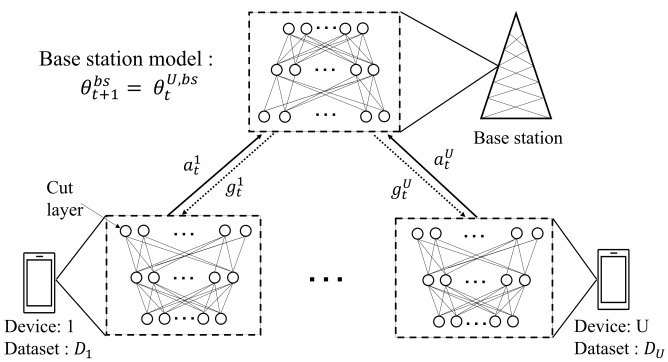


Fig. 3: Schematic diagram for a split learning system

in the received frame, i.e $\boldsymbol{R}_u(k)$, and is trained to recover the I and Q amplitudes of the transmitted symbols $\boldsymbol{X}(k)$.

C. Problem formulation

The aim here is to train models that can recover sent symbols $\boldsymbol{y}$ from received symbols $\boldsymbol{x}$ by creating a non-linear mapping function $f(\cdot)$ between $\boldsymbol{x}$ and $\boldsymbol{y}$. The function, $f(\mathbf{x}|\boldsymbol{\theta}) = \hat{\boldsymbol{y}}$, finds predicted outputs $\hat{\boldsymbol{y}}$ from a model parameter vector $\boldsymbol{\theta}$ and $\boldsymbol{x}$. The optimization of $\boldsymbol{\theta}$ minimises the empirical loss between $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$. For both SISO and OFDM, a global model with parameter vector $\boldsymbol{\theta}^*$ that minimizes the global loss function $\mathcal{L}(\boldsymbol{\theta})$, is found using datasets spread across users. Distributed learning techniques, Federated and split learning are used to find $\boldsymbol{\theta}^*$.

IV. Methodology

In this section the specific loss functions are defined for the OFDM and SISO systems. Also discussed are the state of the art techniques used in the distributed learning algorithms, that allow for the loss functions on specific user datasets to minimise the global loss function $\mathcal{L}(\boldsymbol{\theta})$. The training of the models consist of three stages, collection of the data samples, optimization of a global model using distributed learning algorithms on local datasets, then detection of unknown symbols using the trained global model.

A. Loss functions

The SISO receiver is modelled as a classification task, so the local loss to be minimized is the categorical cross entropy loss [2]. For categorical cross entropy loss the output of the model is a mapped to $[0, 1]^M$ using a final soft max layer. The local loss using device parameters $\boldsymbol{\theta}$ is given by

$$\mathcal{L}_u^{SISO}(\boldsymbol{\theta}) = -\frac{1}{d_u} \sum_{i=1}^{d_u} \boldsymbol{y}_i^{(u)} \cdot \log(f(\boldsymbol{x}_i^{(u)}|\boldsymbol{\theta})), \tag{5}$$

where $\boldsymbol{y}_i^{(u)}$ and $\boldsymbol{x}_i^{(u)}$ are the input and label vectors of a data sample i on device u, respectively and $d_u = |D_u|$.

The OFDM receiver is modelled as a regression problem as it predicts a sequence of symbols [13]. The local loss is the mean squared error (MSE) and is given by

$$\mathcal{L}_u^{OFDM}(\boldsymbol{\theta}) = \frac{1}{d_u} \sum_{i=1}^{d_u} (f(\boldsymbol{x}_i^{(u)}|\boldsymbol{\theta}) - \boldsymbol{y}_i^{(u)})^2. \tag{6}$$

B. Federated learning

From [10], federated learning aims to find a global optimal model by finding the weighted average of the $\mathcal{L}_u(\boldsymbol{\theta})$ values for each user device. Giving the global loss function as

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{u=1}^{U} \frac{d_u}{d} \mathcal{L}_u(\boldsymbol{\theta}), \tag{7}$$

where d is the total size of the dataset across all users. Fig. 2 displays a visualisation of federated learning. In this process the global model vector $\boldsymbol{\theta}_t$ is transferred to all users. Then each user utilises this model to train on its local dataset by minimising its local loss function, (5) or (6). Normal gradient descent is used to update the local parameters to $\boldsymbol{\theta}_{t+1}^u$ and the global model is updated at the device. The model update $\Delta\boldsymbol{\theta}_t^u$ is found as the difference between the global model and locally updated one. $\Delta\boldsymbol{\theta}_t^u$ is then sent to the BS and gradient aggregation [10], as shown in Fig. 2, is completed to find $\boldsymbol{\theta}_{t+1}$. A single, or multiple, gradient descent epochs can take place on the device before this gradient aggregation step.

C. Split learning

Split learning seeks to train a model that is split between the BS and devices. In its simplest form, as explored here, a globally optimized full model cannot be found as the client models are not shared. This means that the local models on each user vary depending on the input data. Instead, the globally optimized model is the portion on the BS, meaning the optimization of the BS model using each user device, is the same as a centralized learning model. The only differences are that the client side model parameters will differ between user devices and the activations and gradients are transmitted to and from the device at the cut layer. Fig. 3 shows a visualisation of the split learning system used in this paper. A user completes forward propagation on a batch up to the cut layer. This

TABLE I: Shows the BERs and the communication overheads in floats per epoch for the split learning receiver for different values of n

n	i.i.d BER	non-i.i.d BER	Floats per epoch
2	0.1295	0.1350	80000
4	0.0707	0.0840	160000
8	0.0577	0.0689	320000
16	0.0563	0.0681	640000
32	0.0563	0.0665	1280000
64	0.0564	0.0667	2560000

activation tensor $\boldsymbol{a}_t^1$ is sent to the base station and the forward pass completed. The loss (5) or (6) is then back propagated to the cut layer layer. The gradients $\boldsymbol{g}_t^1$ are sent back to the user device and the weights updated on the device model, this repeated for all batches on all users. The optimized global model for that epoch will be the BS model after using the last device for training, this becomes $\boldsymbol{\theta}_{t+1}^{bs}$.

V. Simulations Results

A. Algorithm Implementations

The federated algorithm implementation is adapted from [2] and uses tensor flow with the Keras API, along with the tensor flow federated library. This library allows for the training models on separate user datasets, and the federated averaging of these models to train a global model. Tensor flow federated is a high level API, so all the sending and receiving of model updates, as well as model aggregation is handled by the methods and functions of the library.

The split learning implementation is adapted from [12] using a message passing interface (MPI) [15]. This allows for the running of a script on multiple virtual processes, each with dedicated memory that can't be accessed by any other. These processes can then act as user devices in the split learning training. One process acts as a sever, where the second half of the network is updated with the training data of each user device. Forward and back propagation are completed as described above on the device and server processes by sending the cut layer activations and gradients back and forth using the MPI. For testing, only the final device worker is utilised, where the forwards pass is completed with no gradient updates.

B. SISO distributed receivers comparison

The NN architecture used in [2] is a soft max regression with 16 classes. This was done to minimize the number of parameter updates needed to be transmitted for federated learning. When using split learning, the only thing that effects the amount of floats needing to be transmitted is the size of the cut layer. With this in mind, there is more freedom in the network architectures. So, the networks on both client and server are made of two linear layers, each with ReLU activation functions. The size of the client side hidden layers are 16 and n, where n is the cut layer size. The server model layers are of length 64 and 32. The input tensor is of size 2,

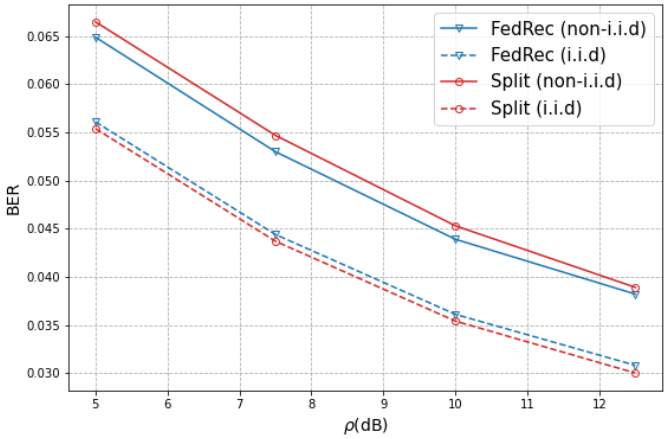


Fig. 4: Shows the BER for the split learning receiver and and FedRec [2] at different signal to noise ratios

which is the I and Q amplitudes of received symbol. 16QAM modulation is used, so $M = 16$ and the output layer is of length 16. This output layer is acted on by a soft max activation function to recover the symbol with the highest probability. The training data set D contains 2×10^4 uniformly distributed 16QAM symbols, evenly split across users, so $|D_u| = 2\times10^4/U$. The test data consists of 1×10^6 symbols.

For each data sample, the transmitted QAM symbols are perturbed by channel coefficients $h_u(t)$ taken from a Rayleigh distribution. The channel coefficients are generated from iid and non-iid distributions across users. In the i.i.d case the variance of the distribution $\sigma_u = 1$ for all users. The non i.i.d case seeks to replicate the differing local statistics of the environment between users and takes the variance of the Rayleigh distribution, for a user, from a uniform distribution with limits between [0.5, 1.5]. The BER is used as the metric for evaluation. It is assumed that there is one incorrect bit in an incorrectly predicted symbol, so the BER is calculated as $1 - accuracy/4$, for the 4 bits per symbol in 16QAM. The signal to noise per bit is denoted here as ρ.

The size of the cut layer tensor will greatly impact the transmission overhead, so the performance of the system is observed as the the size of this tensor n is changed. During the simulations, the number of devices U = 5, training epochs = 30, batch size = 200 , learning rate = 0.001 with an Adam optimizer and $\rho = 5$. Table I shows the bit error rate across a range of n. Also shown is the transmission overhead in floats per epoch, calculated as $2|D| \times n$ [14]. The table shows a sharp decline in the BER as the cut layer tensor size increases, with the decrease slowing around $n = 16$. The performance at low n is expected, as the tensor will not contain enough information about the input data to train to the model on the server side. The performance on non-i.i.d data doesn't seem to improve after $n = 32$, so this is used for the rest of the simulations.

To compare the split learning receiver and the federated receiver in [2], it is useful to see how well the trained receiver

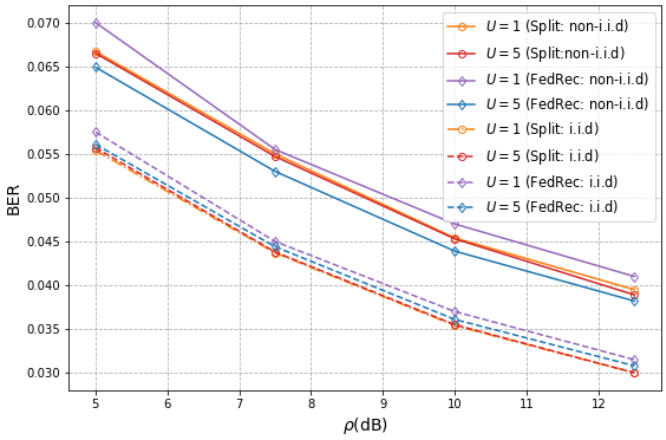
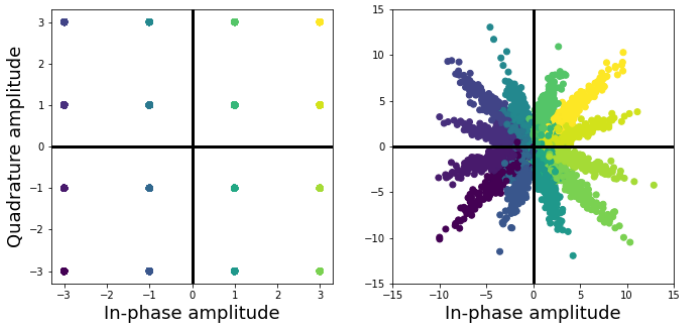


Fig. 5: Shows the BER as the the amount of users in training is varied for the split learning receiver and FedRec [2]



(a) before channel distortion (b) after channel distortion

Fig. 6: Shows the distribution of the input data

performs at different signal to noise ratios. Fig. 4 shows the performance of the receivers across different SNR with i.i.d and non-i.i.d fading, U = 5, epochs = 30, and n = 32. The federated receiver outperforms the split receiver at all SNR values with non-i.i.d fading, whereas the opposite is true for i.i.d fading. The drop in performance may be due to the client model not being shared during split learning, causing each client side model to be slightly different, depending on its local channel conditions.

Fig. 5 shows the effect that the number of users has on the performance of the split learning receiver and FedRec. Using split learning, for both i.i.d and non-i.i.d, the number of users doesn't effect the BER at each SNR value in any significant way. This is different to the behaviour for FedRec, where the performance of the global model significantly drops with the number of users, especially in the non-i.i.d case. These simulation seem to suggest that the split receiver is more robust to the number of users in training, though if there is more that one user device, FedRec outperforms split learning.

From [2], the total communications overhead for FedRec is 1440 floats, compared to the 2.84×10^7 floats of the split receiver. This means that split learning has failed to improve the performance, while increasing the communications overhead greatly. Split learning gives the opportunity to build much larger networks where only the size of the cut layer tensor effects the communications overhead, with the trade off being the cut layer tensor must be transmitted for every training sample. This could not be exploited in this instance due to the low dimensionality of the input data, meaning the improvement that can be made over the soft max regression model used in FedRec [2] is limited. Fig. 6(a) shows the constellation plot of the of the sent pilot symbols and Fig. 6(b) shows the symbols after undergoing Rayleigh fading. Therefore, this is the distribution of the received pilot symbols that are collected by a user device and input into the model for training. As the dimensionality of the data is low, it is easy to see that a linear classifier would do just as well as the more complex non-linear classifier implemented here. There will be a maximum to the accuracy of the classifier no matter the complexity of the model, as the inner constellation points cannot be resolved in this low dimensional feature space. To leverage the advantages of more complex DNNs for detection, and therefore the advantages of split learning, the feature space of the input data must of higher dimensionality.

C. OFDM distributed learning based receivers

The architecture for the OFDM receiver is taken from [12] and is a simple multilayer perceptron with 3 hidden layers of sizes, 500, 250 and 100. The input to the network is a tensor of length 256. This is the real and imaginary parts of the 2, 64 sub-carrier OFDM symbols that make up a frame. QPSK modulated symbols are on each sub-carrier, so $M = 4$. In this example, to test the hypothesis of distributed training of an OFDM receiver, the network is trained to recover the I and Q amplitudes of the first 8 symbols in the data carrying block, so the output layer is of size 16. Again BER is used a the evaluation metric and is calculated by rounding the output values of the DNN to +/- 1 depending on the sign. A bit error is when these values do not match the input symbol I and Q amplitudes. This is possible with QPSK modulation, as each symbol carries two bits of information.

The Channel data is the same as used in [13] and is generated using the WINNER II [16] channel model. The channel model was created with a carrier frequency of 2.6 GHz, 24 paths and a typical urban channel with a maximum delay sampling period of 16. In practice, this means the OFDM symbols in each frame will be convolved with a length 16 channel vector taken from a large dataset of channel vectors generated using the channel model. This simulates an OFDM frame that has undergone multipath fading, due to a dispersive channel.

The results of the simulation, shown in Fig. 7, examine how the BER changes with the SNR with two different number of pilots. Here the model is trained using 5 users each with local datasets of 2×10^4 frames. The federated and split learning models are trained over 100 epochs, where the federated model undergoes 10 gradient aggregation rounds, meaning 10

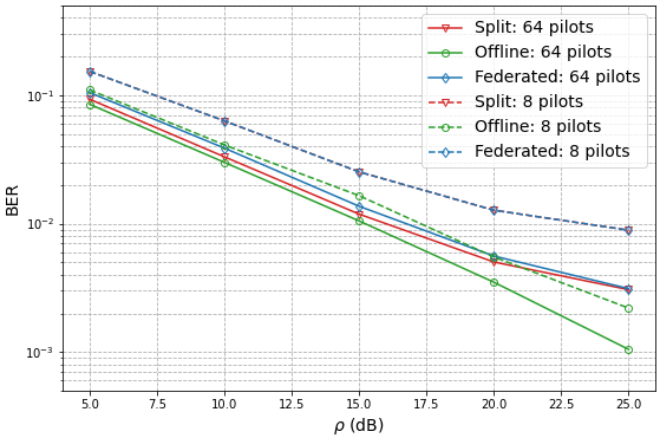


Fig. 7: Shows the change in BER as the SNR is varied for all the DNN models, offline data from [13]

local epochs before model aggregation. They are both trained at $SNR = 30dB$, and the trained model used to test the performance at different SNR values. The test dataset consists of 1×10^5 frames, generated in the same way as the training frames, but using a separate set of channel vectors. The split learning model uses a weight decay of 1×10^{-3} on both client and server to avoid over-fitting to the limited training samples. The network is cut at the 250 length hidden layer. Both models use Adam optimizers with learning rate=0.001, but in the split learning case, the learning rate is reduced by a factor of 0.4 every 10 epochs.

The results show comparable performance between the 3 models, with the offline trained model coming out just ahead. This is most likely due to the fact that the offline model, from [3], is trained using a generated dataset for 1000s of epochs. This means the effective size of the training dataset is not fixed in length and each batch is generated using different random channel vectors continuously as the model is trained. This allows for more generalisation to diverse channel realizations during deployment. The move to online training limits the datasets to being collected and stored on training devices, meaning each user holds a limited set of channel realization and data sequences. Also in this context, training time is limited to 50 epochs. Interestingly, the BER levels out for the online training methods as the SNR goes above 20 dB. When the number of pilots is reduced to 8, the gap in performance is a bit larger. The distributed learning methods have very similar performance in this simulation, with both 8 and 64 pilots. In this instance, Offline learning gives better performance, but the BERs of the online learning methods are within the same order of magnitude, while being trained on a limited dataset for a small number of epochs.

Table II shows a comparison of the communications efficiency of each model. During federated learning, the up-link (UL) overhead is $|\boldsymbol{\theta}| \times U \times a$, where a is the number of gradient aggregation rounds and $|\boldsymbol{\theta}|$ is the number of model parameters. Down-link (DL) is $|\boldsymbol{\theta}| \times a$, as only one global model is transmitted to all users per aggregation round. The offline UL overhead is $d \times |\boldsymbol{x}_i|$, where d is the size of the full dataset and $|\boldsymbol{x}_i|$ is the size of the inputs. The DL overhead is just $|\boldsymbol{\theta}|$. Split learning overheads are calculated as in the previous subsection. These results show federated learning to be more communication efficient than offline learning, assuming the same size dataset is used. Again, split learning has a much larger overhead, this is due to the low model complexity and small number of users [14].

TABLE II: A comparison of the communications overhead in each model, measured in total floats

Model	UL	DL	Total
Offline learning	2.56×10^7	2.86×10^5	2.59×10^7
Federated learning	1.43×10^7	1.43×10^6	1.57×10^7
Split learning	2.5×10^9	2.5×10^9	5×10^9

VI. Conclusions and Future Work

In this paper, DNN based universal receivers, trained using various distributed learning techniques are compared. These receivers exploit channel diversity across multiple user device datasets, to train receivers robust to diverse channel realizations. This is done for SISO and OFDM communication systems, in both cases federated learning is seen to be more communications efficient while overall giving comparable performance to split learning. This is due to the fact that the models used in both cases have a relatively low complexity. In the SISO case, due to the low dimensionality of the input, the more complex model used for split learning is unable to leverage significant performance advantages, while increasing communication overhead hugely. Overall, split learning shows a slight performance drop in the non-i.i.d case, but also shows more robustness to the number of users. For the OFDM receiver, both federated learning and split learning show performance that approaches that of an offline trained model. Due to the OFDM receiver being more parametrised, the communication overheads between split learning and federated learning are less disparate in this instance. Even so, split learning still has an overhead a few orders of magnitude higher and is therefore the more efficient model to implement in this instance.

There are still many potential areas for further study. In this paper, it is assumed that all floats transmitted during training are transmitted between deceives and BS error free. In practice, these floats will be subject to the same channel conditions as the data symbols used to train the models. The effect of these errors due to the channel on the convergence of the distributed learning models could be explored. Recently more complex distributed algorithms, such as hybrid split-federated learning systems have been proposed. These models attempts to leverage the advantages of both algorithms, while avoiding some of the drawbacks. It would be useful to test these newer hybrid methods for signal detection, to explore any possible performances or implementation advantages.

REFERENCES

[1] H. Huang et al., "Deep Learning for Physical-Layer 5G Wireless Techniques: Opportunities, Challenges and Solutions," in IEEE Wireless Communications, vol. 27, no. 1, pp. 214-222, February 2020, doi: 10.1109/MWC.2019.1900027.

[2] M. B. Mashhadi, N. Shlezinger, Y. C. Eldar, and D. Gunduz, "Fedrec: Federated learning of universal receivers over fading channels," arXiv:2011.07271, 2021.

[3] M. Soltani, V. Pourahmadi, A. Mirzaei and H. Sheikhzadeh, "Deep Learning-Based Channel Estimation," in IEEE Communications Letters, vol. 23, no. 4, pp. 652-655, April 2019, doi: 10.1109/LCOMM.2019.2898944.

[4] P. Dong, H. Zhang, G. Y. Li, I. S. Gaspar and N. NaderiAlizadeh, "Deep CNN-Based Channel Estimation for mmWave Massive MIMO Systems," in IEEE Journal of Selected Topics in Signal Processing, vol. 13, no. 5, pp. 989-1000, Sept. 2019, doi: 10.1109/JSTSP.2019.2925975.

[5] A. M. Elbir, A. Papazafeiropoulos, P. Kourtessis and S. Chatzinotas, "Deep Channel Learning for Large Intelligent Surfaces Aided mm-Wave Massive MIMO Systems," in IEEE Wireless Communications Letters, vol. 9, no. 9, pp. 1447-1451, Sept. 2020, doi: 10.1109/LWC.2020.2993699..

[6] N. Farsad and A. Goldsmith, "Neural Network Detection of Data Sequences in Communication Systems," in IEEE Transactions on Signal Processing, vol. 66, no. 21, pp. 5663-5678, 1 Nov.1, 2018, doi: 10.1109/TSP.2018.2868322.

[7] T. O'Shea and J. Hoydis, "An Introduction to Deep Learning for the Physical Layer," in IEEE Transactions on Cognitive Communications and Networking, vol. 3, no. 4, pp. 563-575, Dec. 2017, doi: 10.1109/TCCN.2017.2758370.

[8] A. Felix, S. Cammerer, S. Dörner, J. Hoydis and S. Ten Brink, "OFDM-Autoencoder for End-to-End Learning of Communications Systems," 2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), 2018, pp. 1-5, doi: 10.1109/SPAWC.2018.8445920.

[9] T. J. O'Shea, T. Erpek, and T. C. Clancy, "Deep learning based MIMO communications," arXiv preprint arXiv:1707.07980, 2017.

[10] O. Shamir, N. Srebro, and T. Zhang, "Communication-efficient distributed optimization using an approximate newton-type method," in International conference on machine learning, 2014, pp. 1000–1008.

[11] A. M. Elbir and S. Coleri, "Federated learning for channel estimation in conventional and IRS-assisted massive MIMO," arXiv preprint arXiv:2008.10846, 2020.

[12] Ioannidou X, Straathof, B, T., (2020) Scalable Machine Learning ID2223 Project: An Investigation into Split Learning

[13] H. Ye, G. Y. Li, and B. H. Juang, "Power of Deep Learning for Channel Estimation and Signal Detection in OFDM Systems," IEEE Wireless Commun. Lett., vol. 7, no. 1, Feb. 2018, pp. 114–17.

[14] A. Singh, P. Vepakomma, O. Gupta, and R. Raskar, "Detailed comparison of communication efficiency of split learning and federated learning," arXiv preprint arXiv:1909.09145, 2019.

[15] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. "A high-performance, portable implementation of the mpi message passing interface standard" Parallel computing, 22(6):789–828, 1996.

[16] P. Kyosti et al., "WINNER II channel models," Eur. Commission, Brussels, Belgium, Tech. Rep. D1.1.2 IST-4-027756 WINNER, Sep. 2007.