Deep Learning-Based Zak-OTFS Receiver with Embedded and Interleaved Pilots

Naveed Bin Nazir and Ananthanarayanan Chockalingam
Department of ECE, Indian Institute of Science, Bangalore

***Abstract*—In this paper, we propose a deep learning-based receiver architecture for joint input-output (I/O) relation estimation and signal detection in Zak transform-based orthogonal time frequency space (Zak-OTFS) systems. Embedded and interleaved pilot frames are considered, where pilot and data symbols coexist in a given frame and interfere with each other. A fully-connected deep neural network (DNN) is trained to jointly learn the effective delay-Doppler (DD) channel and optimize detection performance as a single task. The proposed deep learning approach is found to effectively handle inter-symbol interference and achieve better bit error rate performance compared to conventional methods that perform the channel estimation and signal detection tasks separately. Even with reduced guard region overhead in embedded pilot frames (high throughput and increased interference), the proposed receiver is found to deliver robust performance, while traditional techniques perform poorly. Further, the deep learning-based receiver is shown to work effectively with interleaved pilot frames, which offer the benefits of increased throughput (due to no guard region between pilot and data) and reduced peak-to-average power ratio.**

***Index Terms*—Delay-Doppler domain, Zak-OTFS modulation, embedded pilots, interleaved pilots, I/O relation estimation, signal detection, deep neural networks.**

I. Introduction

Orthogonal time frequency space (OTFS) modulation [1] is a state-of-the-art delay-Doppler (DD) domain modulation that multiplexes information symbols in the DD domain. It effectively handles the high time and frequency dispersion of the channel in high-mobility environments and achieves robust performance. The multicarrier OTFS (MC-OTFS) waveform introduced in [1] uses an inner multicarrier modulation block for time-frequency (TF) domain to time domain conversion, along with an outer pre-processing block for DD domain to TF domain conversion. Introduced more recently in [2],[3], a new version of OTFS using Zak transform, called Zak-OTFS, uses a direct transformation from DD domain to time domain using inverse Zak transform [4]-[9]. Zak-OTFS has the advantages of a formal mathematical framework using Zak theory to describe the waveform and it provides better performance compared to MC-OTFS [2],[3]. In this paper, we consider Zak-OTFS with a focus on its receiver implementation in a deep learning framework.

The basic Zak-OTFS carrier waveform is a quasi-periodic localized pulse in the DD domain, which appears in the time domain as a pulse train modulated by a tone, termed a pulsone. Each pulsone carries a data or pilot symbol, and multiple pulsones form a frame. To confine transmission within finite bandwidth and duration, a DD-domain pulse shaping filter is applied at the transmitter. Common choices include sinc, Gaussian, and root-raised cosine filters [3]–[9]. The DD characteristics of this filter determine the extent of aliasing from quasi-periodic replicas and the level of inter-symbol interference, thereby directly influencing receiver performance.

Accurate estimation of the end-to-end DD-domain input-output (I/O) relation and reliable symbol detection are therefore key tasks at the Zak-OTFS receiver. Existing works on Zak-OTFS typically address I/O relation estimation and signal detection as separate tasks. For instance, signal detection has been studied using minimum mean square error (MMSE) methods [3] and local search techniques [9], while I/O estimation has been tackled via model-dependent and model-free approaches. In the model-dependent approach, physical channel parameters are first estimated and then used to construct the I/O relation. In contrast, the model-free approach bypasses explicit parameter estimation by transmitting a pilot symbol and directly reading the corresponding DD-domain output samples in the read-off region.

In this work, we adopt a deep learning framework [10]–[12] that performs I/O estimation and symbol detection jointly. This joint treatment improves overall receiver performance compared to conventional methods. We design the receiver for both embedded and interleaved pilot frames. In embedded pilot frame, a pilot symbol coexists with data symbols, separated by guard space to reduce inter-symbol interference (ISI), but at the cost of throughput loss. Interleaved pilots remove guard space, thus alleviating this loss. A fully connected DNN is trained to jointly learn the effective DD-domain channel and optimize detection performance as a single task. Simulation results with sinc and Gaussian pulse shaping filters and vehicular-A (Veh-A) fractional DD channels [13] demonstrate that the proposed deep learning-based receiver effectively mitigates ISI and achieves superior BER performance compared to conventional two-step methods.

II. Zak-OTFS system model

The block diagram of Zak-OTFS transceiver is shown in Fig. 1. The basic information carrier in Zak-OTFS is a quasi-periodic DD domain pulse localized in a fundamental DD period, defined by a delay period τ_{p} and a Doppler period ν_{p} such that $\tau_{\mathrm{p}}\nu_{\mathrm{p}} = 1$. The fundamental DD period is defined as $\mathcal{D}_0 = \{(\tau, \nu) \mid 0 \leq \tau < \tau_p, 0 \leq \nu < \nu_p\}$, where τ and ν represent the delay and Doppler variables, respectively. A quasi-periodic DD domain pulse, when viewed in the time

This work was supported in part by the J. C. Bose National Fellowship and the NSF-DST project (International Cooperation Division), Department of Science and Technology, Government of India.

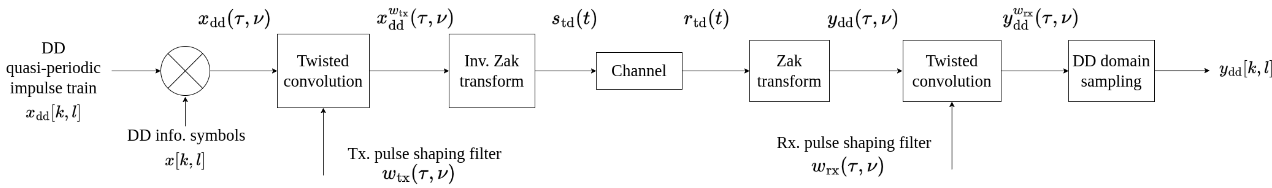


Fig. 1: Zak-OTFS transceiver signal flow diagram.

domain, is a pulsone which is a time domain pulse train modulated by a frequency tone. The delay period $\tau_{\rm p}$ is divided into M delay bins and the Doppler period $\nu_{\rm p}$ is divided into N Doppler bins, and MN information symbols are multiplexed on MN DD pulses located at these MN DD bins. M and N are chosen such that $MN = BT$, where B and T are the bandwidth of transmission and time duration of a Zak-OTFS frame respectively. That is, the resolution along the delay axis is $\Delta\tau = \frac{1}{B} = \frac{\tau_{\rm p}}{M}$ and the resolution along the Doppler axis is $\Delta\nu = \frac{1}{T} = \frac{\nu_{\rm p}}{N}$. In order to limit the bandwidth and time duration to B and T, respectively, a DD domain pulse shaping filter is used at the transmitter.

The information symbols $x[k,l]$s, $k = 0,\ldots,M-1$ and $l = 0,\ldots,N-1$, are drawn from a modulation alphabet $\mathbb{A}$. These MN symbols are placed on a uniform DD grid defined as $\Lambda_{\rm dd} \triangleq \left\{\left(k\frac{\tau_{\rm p}}{M}, l\frac{\nu_{\rm p}}{N}\right) \mid k = 0,\ldots,M-1,\ l = 0,\ldots,N-1\right\}$. The grid symbols $x[k,l]$s are extended into a quasi-periodic discrete DD-domain sequence as $x_{\rm dd}[k+nM, l+mN] = x[k,l]\, e^{j2\pi n\frac{l}{N}},\quad n,m \in \mathbb{Z}$, which is then used to construct a continuous DD-domain signal by modulating it onto a 2D impulse train as $x_{\rm dd}(\tau,\nu) = \sum_{k,l\in\mathbb{Z}} x_{\rm dd}[k,l]\,\delta\left(\tau - k\Delta\tau\right)\,\delta\left(\nu - l\Delta\nu\right)$, where $\delta(\cdot)$ denotes the Kronecker delta function. This signal $x_{\rm dd}(\tau,\nu)$ retains quasi-periodicity in both delay and Doppler dimensions, i.e., $x_{\rm dd}(\tau + n\tau_{\rm p}, \nu + m\nu_{\rm p}) = e^{j2\pi n\nu\tau_{\rm p}}\, x_{\rm dd}(\tau,\nu),\quad \forall n,m \in \mathbb{Z}$. To ensure that the signal is time and bandwidth limited, it is passed through a transmit DD-domain pulse shaping filter $w_{\rm tx}(\tau,\nu)$, resulting in $x_{\rm dd}^{w_{\rm tx}}(\tau,\nu) = w_{\rm tx}(\tau,\nu) *_\sigma x_{\rm dd}(\tau,\nu)$, where $*_\sigma$ denotes the twisted convolution[1] operation. The time-domain signal for transmission is obtained using the inverse Zak transform[2], given by $s_{\rm td}(t) = \mathcal{Z}_t^{-1}\left(x_{\rm dd}^{w_{\rm tx}}(\tau,\nu)\right)$.

This transmitted signal passes through a doubly-selective channel characterized by P propagation paths. The corresponding DD domain channel impulse response is modeled as $h(\tau,\nu) = \sum_{i=1}^{P} h_i\delta(\tau-\tau_i)\delta(\nu-\nu_i)$, where h_i, τ_i, and ν_i denote the gain, delay, and Doppler shift of the ith path, respectively. The received time domain signal at the receiver is $r_{\rm td}(t) = \iint h(\tau,\nu)s_{\rm td}(t-\tau)e^{j2\pi\nu(t-\tau)}\,d\tau\,d\nu + n(t)$, where $n(t)$ denotes additive white Gaussian noise (AWGN). The received time domain signal is converted to a DD signal using Zak transform[3] as $y_{\rm dd}(\tau,\nu) = \mathcal{Z}_t(r_{\rm td}(t)) = \sqrt{\tau_p}\sum_{k\in\mathbb{Z}} r_{\rm td}(\tau + k\tau_p)e^{-j2\pi\nu k\tau_p} + n_{\rm dd}(\tau,\nu)$, where $n_{\rm dd}(\tau,\nu) = \mathcal{Z}_t(n(t))$ represents the DD domain noise. This DD domain signal is filtered using a receive DD domain filter $w_{\rm rx}(\tau,\nu)$ matched to the transmit DD filter, i.e., $w_{\rm rx}(\tau,\nu) = w_{\rm tx}^*(-\tau,-\nu)e^{j2\pi\tau\nu}$. The output of the receiver DD filter is given by $y_{\rm dd}^{w_{\rm rx}}(\tau,\nu) = w_{\rm rx}(\tau,\nu) *_\sigma y_{\rm dd}(\tau,\nu) = h_{\rm eff}(\tau,\nu) *_\sigma x_{\rm dd}(\tau,\nu) + n_{\rm dd}^{w_{\rm rx}}(\tau,\nu)$, where $h_{\rm eff}(\tau,\nu)$ denotes the effective continuous DD channel, given by

$$h_{\rm eff}(\tau,\nu) = w_{\rm rx}(\tau,\nu) *_\sigma h(\tau,\nu) *_\sigma w_{\rm tx}(\tau,\nu), \tag{1}$$

and $n_{\rm dd}^{w_{\rm rx}}(\tau,\nu) = w_{\rm rx}(\tau,\nu) *_\sigma n_{\rm dd}(\tau,\nu)$ represents the filtered AWGN in the DD domain. The filtered DD signal $y_{\rm dd}^{w_{\rm rx}}(\tau,\nu)$ is then sampled on the information lattice to obtain the discrete quasi-periodic DD domain received signal $y_{\rm dd}[k,l]$, as

$$y_{\rm dd}[k,l] = y_{\rm dd}^{w_{\rm rx}}\left(\tau = \frac{k\tau_{\rm p}}{M}, \nu = \frac{l\nu_{\rm p}}{N}\right),\quad k,l\in\mathbb{Z}, \tag{2}$$

which can be expressed as

$$y_{\rm dd}[k,l] = h_{\rm eff}[k,l] *_{\sigma \rm d} x_{\rm dd}[k,l] + n_{\rm dd}[k,l], \tag{3}$$

where $*_{\sigma\rm d}$ denotes the twisted convolution in the discrete DD domain, defined as

$$h_{\rm eff}[k,l] *_{\sigma\rm d} x_{\rm dd}[k,l] = \sum_{k',l'\in\mathbb{Z}} h_{\rm eff}[k-k', l-l']x_{\rm dd}[k',l']e^{j2\pi\frac{k'(l-l')}{MN}},$$

where the effective channel samples and filtered noise in the discrete DD domain are given by

$$h_{\rm eff}[k,l] = h_{\rm eff}\left(\tau = \frac{k\tau_p}{M}, \nu = \frac{l\nu_p}{N}\right), \tag{4}$$

$$n_{\rm dd}[k,l] = n_{\rm dd}^{w_{\rm rx}}\left(\tau = \frac{k\tau_p}{M}, \nu = \frac{l\nu_p}{N}\right). \tag{5}$$

Due to the quasi-periodicity of the DD domain signal, it suffices to consider the received samples $y_{\rm dd}[k,l]$ only within the fundamental period $\mathcal{D}_0$. These samples are stacked into a vector to express the end-to-end DD domain input-output relationship in matrix-vector form as

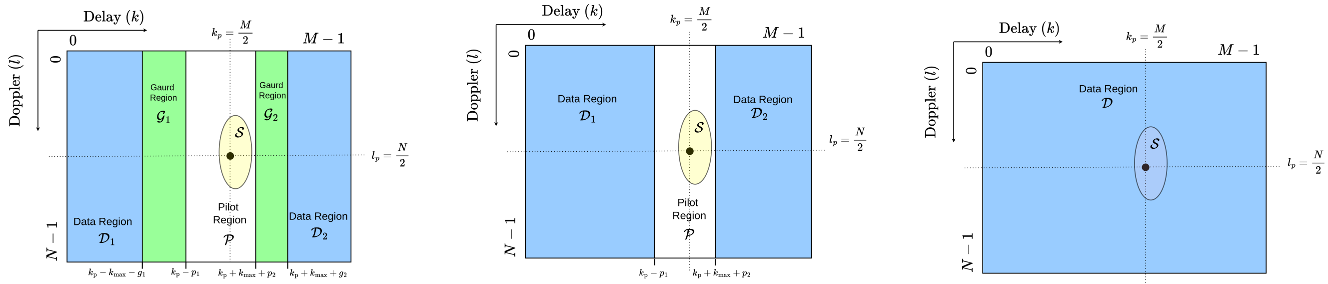
$$\mathbf{y} = \mathbf{H}_{\rm eff}\mathbf{x} + \mathbf{n}, \tag{6}$$

where $\mathbf{x},\mathbf{y},\mathbf{n} \in \mathbb{C}^{MN\times 1}$, such that their $(kN+l+1)^{\rm th}$ entries are given by $x_{kN+l+1} = x_{\rm dd}[k,l]$, $y_{kN+l+1} = y_{\rm dd}[k,l]$, $n_{kN+l+1} = n_{\rm dd}[k,l]$, and $\mathbf{H}_{\rm eff} \in \mathbb{C}^{MN\times MN}$ is the effective channel matrix such that

[1]Twisted convolution operation between two DD functions $a(\tau,\nu)$ and $b(\tau,\nu)$ is defined as $a(\tau,\nu) *_\sigma b(\tau,\nu) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} a(\tau',\nu')b(\tau-\tau',\nu-\nu')e^{j2\pi\nu'(\tau-\tau')}d\tau' d\nu'$. Twisted convolution operation preserves quasi-periodicity.

[2]Inverse Zak transform of a DD signal $x(\tau,\nu)$ is defined as $\mathcal{Z}_t^{-1}\left(x(\tau,\nu)\right) \triangleq \sqrt{\tau_p}\int_0^{\nu_p} x(t,\nu)d\nu$.

[3]The Zak transform of a time domain signal $x(t)$ is given by $\mathcal{Z}_t\left(x(t)\right) \triangleq \sqrt{\tau_p}\sum_{k\in\mathbb{Z}} x(\tau + k\tau_p)e^{-j2\pi\nu k\tau_p}$.



(a) Type-A frame (b) Type-B frame (c) Type-C frame

Fig. 2: Embedded pilot frame structure.

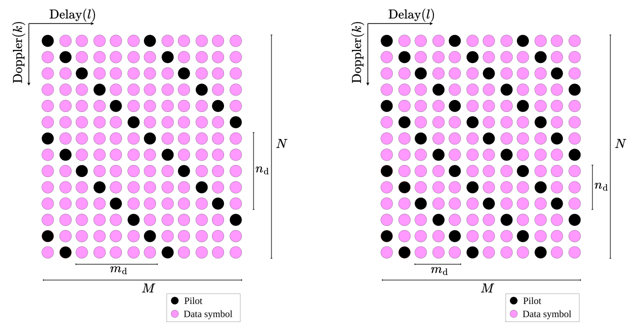
$$\mathbf{H}_{\text{eff}}[k'N + l' + 1, kN + l + 1] = \sum_{m,n\in\mathbb{Z}} h_{\text{eff}}[k' - k - nM, \\ l' - l - mN] e^{j2\pi nl/N} e^{j2\pi\frac{(l'-l-mN)(k+nM)}{MN}}, \quad (7)$$

where $k', k = 0, \ldots, M-1$, $l', l = 0, \ldots, N-1$. Closed-form expressions for $h_{\text{eff}}[k, l]$ and noise covariance for sinc and Gaussian filters are derived in [16].

We consider embedded and interleaved pilot frames as shown in Figs. 2 and 3, respectively. In an embedded pilot frame (see Type-A frame in Fig. 2(a)), there is a pilot symbol at the middle of the frame, which is surrounded by a pilot region $\mathcal{P}$ as the read-off region for model-free I/O relation estimation as done in conventional methods [7], a guard region ($\mathcal{G}_1$ and $\mathcal{G}_2$) to avoid interference between pilot and data symbols, and a data region ($\mathcal{D}_1$ and $\mathcal{D}_2$). $\mathcal{S}$ denotes the support of the effective DD channel. Since pilot and guard regions are left empty, throughput suffers (depending on the choice of the frame parameters in Fig. 2 that define various regions). To increase throughput and assess the effectiveness of the DNN approach, we consider a Type-B frame (see Fig. 2(b)) where we drop the guard region and fill it with data symbols. Also, since our deep learning-based approach does not perform an explicit estimation of I/O relation, we consider a Type-C frame where we drop the pilot region as well and fill it also with data symbols. Note that while the throughput is more in Type-B and Type-C frames, they come with the challenge of increased ISI. In the interleaved pilot frame (see Fig. 3), multiple pilot symbols are interleaved with data symbols without leaving any guard/read-off space. The pilot density (and the hence the throughput) is controlled by the choice of the frame parameters in Fig. 3. Unlike in embedded pilot frame where the full pilot power is put on one pilot symbol, in interleaved pilot frame the pilot power is equally distributed among all the pilot symbols resulting in better peak-to-average power ratio.

III. Proposed Deep Learning-Based Receiver Architecture

The proposed deep learning-based Zak-OTFS receiver architecture for a real-valued modulation alphabet (e.g., BPSK)



(a) $m_d = n_d = 5$ ($N_{\text{p}} = 28$) (b) $m_d = n_d = 3$ ($N_{\text{p}} = 42$)

Fig. 3: Interleaved pilot frame structure.

is illustrated in Fig. 4[4]. It consists of a fully connected DNN designed to jointly learn the I/O relationship and detect the transmitted symbols. The network takes as input a real-valued vector of dimension $2MN$ and produces an output vector of size N_{d}, where N_{d} is the number of data symbols in a frame. Each input-output pair corresponds to one realization of Zak-OTFS frame generated using the system model in (6):

$$\mathbf{y}' = \mathbf{H}_{\text{eff}}\mathbf{x}' + \mathbf{n}, \quad (8)$$

where $\mathbf{x}' \in \mathbb{C}^{MN\times 1}$ is the transmitted signal vector (embedded or interleaved pilot frame) and $\mathbf{y}' \in \mathbb{C}^{MN\times 1}$ is the corresponding received signal vector. To make the data suitable for DNN processing, $\mathbf{y}'$ is converted into a real-valued vector $\mathbf{y}'' \in \mathbb{R}^{2MN\times 1}$ by stacking its real and imaginary parts as

$$\mathbf{y}'' = \begin{bmatrix} \Re(\mathbf{y}') \\ \Im(\mathbf{y}') \end{bmatrix}, \quad (9)$$

where $\Re(\cdot)$ and $\Im(\cdot)$ denote the real and imaginary parts, respectively. This vector $\mathbf{y}''$ serves as the input to the DNN. The network processes it and outputs the estimated data symbols, denoted by $\mathbf{x}_{\text{D}}$. The number of output neurons is therefore N_{d}, with each neuron representing one estimated data symbol. Hence, $\mathbf{x}_{\text{D}} \in \mathbb{R}^{N_{\text{d}}}$.

Training: To train the DNN, a data set comprising multiple input–output pairs $(\mathbf{y}'', \mathbf{x}_{\text{D}})$ is generated, where $\mathbf{y}''$ serves

[4]For complex-valued modulation alphabets (e.g., M-QAM), two separate DNNs like the one in Fig. 4 (one for real and another for imaginary) can be employed to separately process and detect the real and imaginary components of the complex-valued symbols. Using this approach, we have obtained and reported results for 8-QAM in Sec. IV.

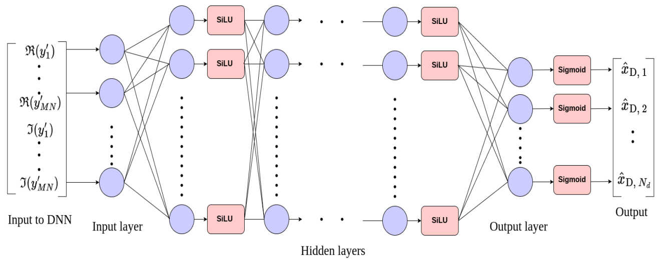


Fig. 4: Proposed deep learning-based Zak-OTFS receiver architecture.

as the input and $\mathbf{x}_\mathrm{D}$ represents the ground truth label. For complex-valued QAM alphabet (e.g., 8-QAM), two separate DNNs are utilized. The first network is trained using the dataset $(\mathbf{y}'', \Re(\mathbf{x}_\mathrm{D}))$ to learn the mapping from $\mathbf{y}''$ to the real part of $\mathbf{x}_\mathrm{D}$, while the second network is trained with $(\mathbf{y}'', \Im(\mathbf{x}_\mathrm{D}))$ to learn the corresponding imaginary part. Here, $\mathbf{x}_\mathrm{D}$ denotes the transmitted symbol vector comprising of complex-valued QAM symbols. The outputs of the two DNNs, $\hat{\mathbf{x}}_{\mathrm{D}_1}$ and $\hat{\mathbf{x}}_{\mathrm{D}_2}$, corresponding to the estimated real and imaginary components, are combined to reconstruct the detected symbol vector as $\hat{\mathbf{x}}_\mathrm{D} = \hat{\mathbf{x}}_{\mathrm{D}_1} + j\hat{\mathbf{x}}_{\mathrm{D}_2}$. This complex-valued symbol output is subsequently demapped to recover the transmitted bit stream.

The network is trained to minimize the root mean squared error (RMSE) loss, defined as $\mathcal{L}(\mathbf{x}_\mathrm{D},\ \hat{\mathbf{x}}_\mathrm{D}) = \sqrt{\frac{1}{K}\sum_{i=1}^{K}\left(x_{\mathrm{D},i} - \hat{x}_{\mathrm{D},i}\right)^2}$, where K is the number of training samples, and $x_{\mathrm{D},i}$ and $\hat{x}_{\mathrm{D},i}$ are the true and predicted values of the i-th data symbol, respectively. The DNN is trained using the Adam optimizer with an initial learning rate of 10^{-3}. A piecewise learning rate schedule is applied, where the rate is reduced by a factor of 0.8 every 3 epochs. Training data sets are generated for sinc and Gaussian pulse shaping filters. The sinc filter is given by $w_{\mathrm{tx}}(\tau,\nu) = \sqrt{BT}\,\mathrm{sinc}(B\tau)\,\mathrm{sinc}(T\nu)$, and the Gaussian filter is given by $w_{\mathrm{tx}}(\tau,\nu) = \left(\frac{2\alpha_\tau B^2}{\pi}\right)^{\frac{1}{4}} e^{-\alpha_\tau B^2\tau^2} \left(\frac{2\alpha_\nu T^2}{\pi}\right)^{\frac{1}{4}} e^{-\alpha_\nu T^2\nu^2}$. To ensure that there is no expansion in bandwidth and time beyond B and T, respectively, a Gaussian filter with $\alpha_\tau = \alpha_\nu = 1.584$ is employed.

IV. Results and Discussions

In this section, we present the simulated BER performance of the proposed deep learning-based receiver for Zak-OTFS using embedded and interleaved pilot frames. We also present a performance comparison between the proposed deep learning-based approach with joint I/O relation estimation and detection and the conventional approach where these two tasks are performed separately. For the conventional approach, we consider the model-free I/O relation estimation in [7], followed by MMSE detection.

System configuration: Simulations are carried out for Zak-OTFS with the following system parameters: Doppler period $\nu_\mathrm{p} = 15$ kHz, corresponding delay period $\tau_\mathrm{p} = \frac{1}{\nu_\mathrm{p}} = 66.66\ \mu$s, frame dimensions $M = 12$ and $N = 14$, giving a total frame duration of $T = N\tau_\mathrm{p} = 0.93$ ms and a bandwidth of $B = M\nu_\mathrm{p} = 180$ kHz. The vehicular-A channel model [13] with fractional DDs having $P = 6$ paths with a maximum Doppler of $\nu_{\max} = 815$ Hz, a maximum delay spread of $\tau_{\max} = 2.51\ \mu$s, and the power delay profile listed in Table I is considered. The Doppler shift for each path is given by $\nu_i = \nu_{\max}\cos(\theta_i)$, where θ_i is uniformly distributed between 0 and 2π. The average energy in a frame is $E_p + E_d$, where E_p and E_d denote the average energy of the pilot and data, respectively. The pilot power to data power ratio (PDR) is defined as $\frac{E_p}{E_d}$.

Path index (i)	1	2	3	4	5	6
Delay τ_i (μs)	0	0.31	0.71	1.09	1.73	2.51
Relative power (dB)	0	-1	-9	-10	-15	-20

TABLE I: Power delay profile of Veh-A channel model.

Parameters	**Proposed DNN**
No. of input neurons	$2MN = 336$
No. of output neurons	N_d
No. of hidden layers	4
Hidden layer activation	SiLU
Output layer activation	Sigmoid
No. of training examples	3,500,000 (BPSK) and 5,000,000 (8-QAM)
No. of epochs	50
Optimization algorithm	Adam
Loss function	RMSE
PDR	0 dB
Data SNR	5 dB and 24 dB
Batch size	2048

TABLE II: Proposed DNN and training parameters for sinc and Gaussian filters.

The DNN architecture designed for the above system has the following layer configuration.

Proposed DNN: *Input → 900 → SiLU → LayerNorm → 700 → SiLU → LayerNorm → 500 → SiLU → LayerNorm → 300 → Sigmoid → Output.*

The network consists of four hidden layers with 900, 700, 500, and 300 neurons, respectively. Each hidden layer employs the Sigmoid Linear Unit (SiLU[5]) activation function [14], followed by layer normalization [15] to stabilize training and accelerate convergence. In terms of complexity, the total number of trainable parameters is computed as $\sum_{l=1}^{L}(n_{l-1}\times n_l + 3n_l)$, where L is the number of fully connected layers, and n_{l-1} and n_l denote the number of input and output neurons, respectively, of the lth layer. The first term in the summation corresponds to the weights, while the second term, $3n_l$, accounts for the biases (n_l) and the LayerNorm parameters ($2n_l$). The total number of trainable parameters is approximately 1.4M for an input size of 336 and an output size

[5] The SiLU activation function, also known as the Swish function, is defined as $\mathrm{SiLU}(x) = x\,\sigma(x)$, where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function. The SiLU activation function is employed in this work because of its non-monotonic behavior, which enables it to retain information from negative input values rather than discarding them entirely, as happens in the case of ReLU. This property helps mitigate information loss and enhances gradient flow, making it especially beneficial for deeper networks.

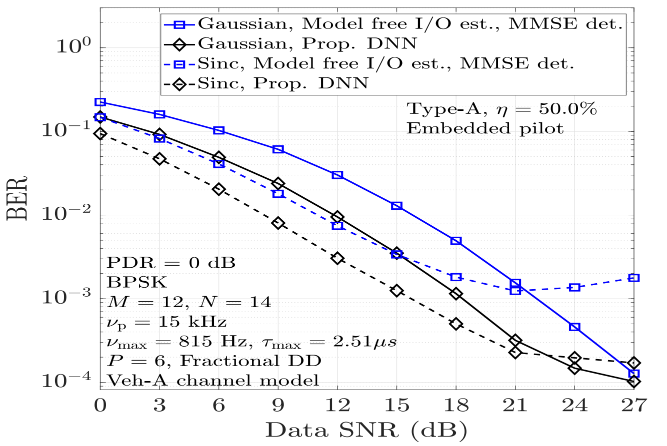


Fig. 5: BER performance of the deep learning-based receiver for sinc and Gaussian filters with Type-A embedded pilot frame.

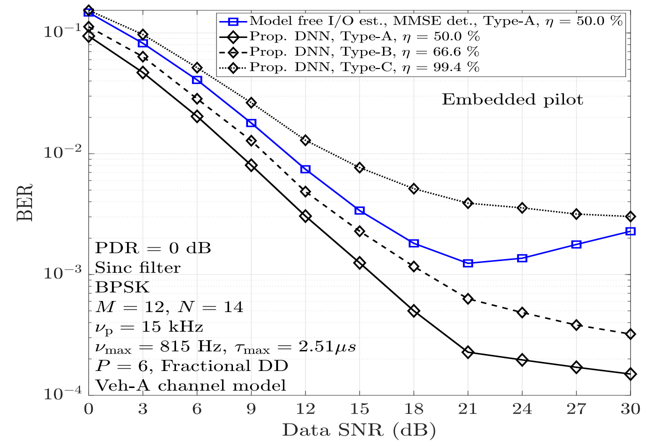


Fig. 6: BER performance of the deep learning-based receiver for Type-A, -B, -C embedded pilot frames with sinc filter.

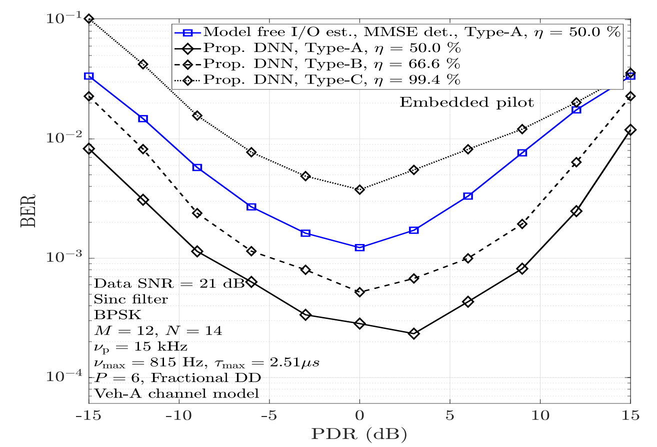


Fig. 7: BER performance of the deep learning-based receiver as a function of PDR for Type-A,-B,-C embedded pilot frames with sinc filter.

of 84. The output layer uses sigmoid activation function for BPSK and linear activation function for 8-QAM. A summary of the proposed deep learning-based architecture and training parameters for sinc and Gaussian filters is provided in Table II.

Results: In Fig. 5, we compare the BER performance of the deep learning-based receiver versus data SNR using sinc and Gaussian filters, along with their model-free I/O estimation and MMSE baselines. A Type-A embedded pilot frame with throughput $\eta = 50$ and PDR $= 0$ dB is considered. With the Gaussian filter, the proposed DNN achieves about 4 dB gain over the model-free baseline at BER 10^{-3}. Between the filters, the sinc outperforms Gaussian up to 22 dB SNR, after which Gaussian becomes slightly better, owing to differences in main- and side-lobe characteristics that affect ISI. At BER 10^{-3}, the sinc filter gives nearly 3 dB gain over Gaussian.

In Fig. 6, we present the BER performance of the deep learning-based receiver for sinc filter under the Type-A, -B, and -C embedded pilot frame configurations. Performance of the model-free baseline under Type-A frame configuration is also plotted. In the Type-A configuration (see Fig. 2a), the pilot symbol is placed at $(k_p, l_p) = \left(\frac{M}{2}, \frac{N}{2}\right)$, with pilot and guard regions defined by $p_1 = 1$, $p_2 = 1$, $g_1 = 1$, and $g_2 = 2$, resulting in $N_\mathrm{d} = 84$ data symbols and a throughput of $\eta = 50\%$. To achieve good performance over a broad SNR range, the DNN is trained using datasets generated at data SNRs of 5 dB and 24 dB. The PDR is fixed at 0 dB. It can be seen that the deep learning-based receiver significantly outperforms the conventional model-free baseline. For example, at a data SNR of 21 dB, the deep learning-based approach achieves a BER of 2.2×10^{-4}, while the model-free approach yields a BER of 1.2×10^{-3}, representing nearly an order of magnitude improvement. For the same system configuration, Fig. 7 shows the BER performance as a function of PDR at a data SNR of 21 dB. Here again, we see that the deep learning-based receiver performance for Type-A frame is much better than that of the model-free baseline. The U-shaped curve arises because of pilot energy leakage into the data region, which interferes with data symbols and degrades performance.

In Figs. 6 and 7, for Type-B configuration, we set $p_1 = 1$, $p_2 = 1$, $g_1 = 0$, and $g_2 = 1$, resulting in $N_\mathrm{d} = 112$ data symbols and a throughput of 66.6%. Despite reduced pilot protection, the deep learning-based receiver maintains robust performance and continues to outperform the model-free baseline, which operates at a lower throughput of 50%. The Type-C configuration maximizes the throughput to 99.4% by removing both the pilot and guard regions (i.e., $N_\mathrm{d} = 167$ out of $MN = 168$ symbols are data symbols). Even without any pilot protection, the deep learning-based receiver continues to deliver reasonable performance with Type-C frames, highlighting its strong generalization capability and robustness across diverse frame structures.

Next, we consider the performance of the deep learning-based receiver for interleaved pilot frames. Figure 8 presents the BER performance as a function of data SNR for interleaved pilot frames with sinc filter. Unlike the embedded approach, where a single pilot symbol is centrally placed with surrounding pilot and guard regions, the interleaved

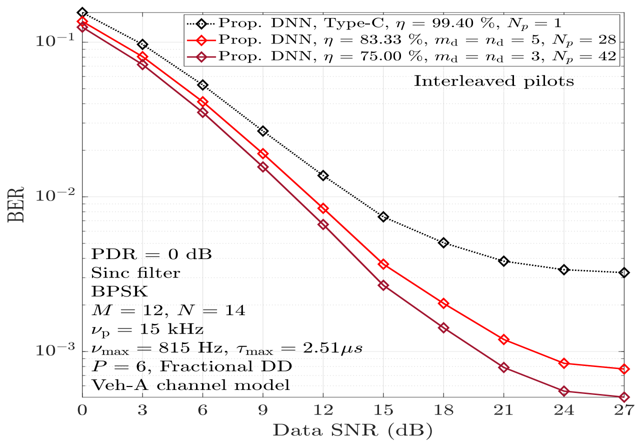


Fig. 8: BER performance of the deep learning-based receiver for sinc filter with interleaved pilot frames.

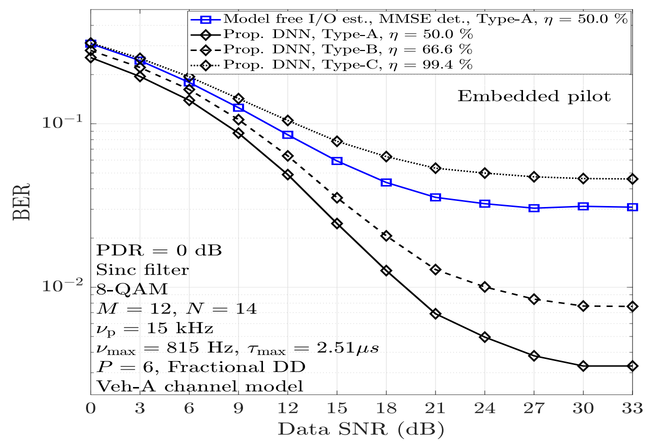


Fig. 9: BER performance of the deep learning-based receiver for 8-QAM with Type-A, -B, -C embedded pilot frames and sinc filter.

approach distributes multiple pilot symbols across the DD grid. This spatial distribution provides enhanced channel coverage, leading to good performance. Furthermore, distributing the pilot power across the grid helps mitigate the high PAPR issue associated with single-pilot designs. In Fig. 8, we consider two configurations: (i) $m_d = n_d = 5$, resulting in $N_p = 28$ pilot symbols and a throughput of 83.3%, and (ii) $m_d = n_d = 3$, resulting in $N_p = 40$ and a throughput of 75%. The performance of Type-C frame with 99.4% throughput with $N_p = 1$ is also included. The results show that increasing N_p improves performance due to better learning of the I/O relation, at the expense of increased pilot overhead. The results highlight the trade-off between pilot overhead and estimation and detection quality.

Finally, we evaluate the performance of the deep learning-based receiver for 8-QAM with embedded pilot frames and the sinc filter. In this setup, two separate DNNs are trained to estimate the real and imaginary parts of the transmitted data symbols. Figure 9 shows the BER performance as a function of data SNR for Type-A, Type-B, and Type-C embedded frame configurations at a PDR of 0 dB. As observed for BPSK, here again the deep learning-based receiver for Type-A ($\eta = 50\%$) and Type-B ($\eta = 66.6\%$) frames achieves a substantial performance gain over the conventional model-free baseline with Type-A frame ($\eta = 50\%$).

V. Conclusions

We proposed a deep learning-based receiver for Zak-OTFS that performs joint I/O relation estimation and signal detection with embedded and interleaved pilot frames. A fully connected DNN was trained on synthetically generated data frames, directly minimizing the RMSE between the true and predicted data symbols. Simulation results demonstrated that the proposed deep learning-based approach achieves superior BER performance compared to conventional methods based on performing model-free I/O estimation and MMSE detection separately. The proposed DNN maintained good performance even as throughput was increased by eliminating guard regions, illustrating its effectiveness. The results show that Zak-OTFS receivers using learning techniques are promising and that investigation of learning approaches for Zak-OTFS with superimposed pilot frames (where a point pilot spread in the DD domain is superimposed on data, achieving 100% spectral efficiency) can be considered for future work.

References

[1] R. Hadani *et al.*, "Orthogonal time frequency space modulation," *Proc. IEEE WCNC'2017*, pp. 1-6, Mar. 2017.
[2] S. K. Mohammed, R. Hadani, A. Chockalingam, and R. Calderbank, "OTFS — a mathematical foundation for communication and radar sensing in the delay-Doppler domain," *IEEE BITS the Inform. Theory Mag.*, vol. 2, no. 2, pp. 36-55, 1 Nov. 2022.
[3] S. K. Mohammed, R. Hadani, A. Chockalingam, and R. Calderbank, "OTFS — predictability in the delay-Doppler domain and its value to communication and radar sensing," *IEEE BITS the Inform. Theory Mag.*, vol. 3, no. 2, pp. 7-31, Jun. 2023.
[4] S. K. Mohammed, R. Hadani, and A. Chockalingam, *OTFS Modulation — Theory and Applications*, IEEE press-Wiley, Nov. 2024.
[5] S. Gopalam, I. B. Collings, S. V. Hanly, H. Inaltekin, S. R. B. Pillai, and P. Whiting, "Zak-OTFS implementation via time and frequency windowing," *IEEE Trans. Commun.*, vol. 72, no. 7, pp. 3873-3889, Jul. 2024.
[6] S. Li, W. Yuan, Z. Wei, J. Yuan, B. Bai, and G. Caire, "On the pulse shaping for delay-Doppler communications," *Proc. IEEE GLOBECOM'2024*, pp. 4909-4914, Dec. 2023.
[7] J. Jayachandran, R. K. Jaiswal, S. K. Mohammed, R. Hadani, A. Chockalingam, and R. Calderbank, "Zak-OTFS: pulse shaping and the tradeoff between time/bandwidth expansion and predictability," online arxiv.org/abs/2405.02718, 4 May 2024.
[8] J. Jayachandran, I. A. Khan, S. K. Mohammed, R. Hadani, A. Chockalingam, and R. Calderbank, "Zak-OTFS with interleaved pilots to extend the region of predictable operation," arXiv:2408.09379v2, 28 Dec 2024.
[9] F. Jesbin and A. Chockalingam, "Near-optimal detection of Zak-OTFS signals," *Proc. IEEE ICC'2024*, pp. 4476-4481, Jun. 2024.
[10] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Trans. Cognitive Commun. and Netw.*, vol. 3, pp. 563-575, Dec. 2017.
[11] L. Dai, R. Jiao, F. Adachi, H. V. Poor, and L. Hanzo, "Deep learning for wireless communications: an emerging interdisciplinary paradigm," *IEEE Wireless Commun.*, vol. 27, no. 4, pp. 133-139, Aug. 2020.
[12] Best Readings in Machine Learning in Communications, https://www.comsoc.org/publications/best-readings/machine-learning-communications.
[13] ITU-R M.1225, "Guidelines for evaluation of radio transmission technologies for IMT-2000," *International Telecommunication Union Radio communication*, 1997.

[14] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," *Neural Networks*, vol. 107, pp. 3-11, 2018.

[15] J. Ba, J. Kiros, and G. Hinton, "Layer normalization," available online: arXiv:1607.06450v1 [stat.ML] 21 July 2016.

[16] A. Das, F. Jesbin, and A. Chockalingam, "Closed-form expressions for I/O relation in Zak-OTFS with different delay-Doppler filters," *IEEE Trans. Veh. Tech.*, 2025. IEEE Xplore early access. doi: 10.1109/TVT.2025.3564419.