Leveraging Deep Learning Methods for Function Inverse Problems in Communication Systems

Panagiotis Skrimponis and Mustafa F. Ozkoc
NYU Tandon School of Engineering, Brooklyn, NY, USA

Abstract

Wireless communications over millimeter wave and terahertz frequency bands have attracted considerable attention for future cellular systems due to the abundantly available bandwidth. However, communications over these high frequencies are vulnerable to blockage, and shadowing. Moreover, the severe penetration loss in these frequencies further degrades the system performance. New trans-receiver designs using large number of antennas are promising to compensate for the harsh propagation characteristics. Combined with beamforming techniques, these new designs can realize the full potential capacity at these high frequencies. However, the received signal quality may significantly degrade due to hardware imperfections of the radio frequency front-end (RFFE) devices. Specifically, the limitations such as noise figure, non-linear distortion, and phase offsets introduced by the radio equipment can significantly deteriorate the system performance. The problem is exacerbated when the input power is too low or too high due to the limited range of ADC. The RFFE distortion can be represented as a non-linear function where the input is the original signal and output is the distorted signal which generally results in information loss. We make novel use of deep learning methods for this function inverse problem, at the distortion mitigation step of the communication systems. Our results show, deep neural networks are promising tools to improve the communication quality in wireless networks suffering from RFFE distortions. We achieve up to $20\,\mathrm{dB}$ increase in the SNR as compared to the baseline method.

I. Introduction

Next generation of cellular networks are envisioned to give life to new applications in many fields such as industrial automation, healthcare, autonomous vehicles and public safety [1], [2]. The stringent throughput requirements of these applications, drive the search for increased wireless link capacity. Millimeter wave (mmWave) and terahertz (THz) frequencies are key enablers to achieve the high throughput requirements of these new applications, with their abundantly available bandwidth [3], [4]. Frequencies in mmWave and THz bands are expected to be used in 5G networks, currently being rolled out, and future 6G networks, respectively.

Although these high frequency bands are capable to achieve multi-gigabit per second throughput, they are susceptible to blockages and shadowing [5], [6], [7]. Even a human body can cause a $20\,\mathrm{dB}$ loss of the received signal strength [8] in mmWave frequencies. The signal degradation is even worse in THz frequencies, where penetration loss can be up to $3-4$ times higher [9].

A further limitation is the significantly higher path loss in these frequencies as compared to the sub-6GHz bands [10], [11]. The over-the-air attenuation of the signal power is proportional to the square of the carrier frequency [12]. To compensate for the high attenuation of the signal power, new transmit and receiver designs utilizing multiple densely packed antennas are proposed [13]. Furthermore, beamforming techniques are used both at the receiver and the transmitter to increase the signal quality [14], [15], [16]. Using multiple antenna elements, a transmitter can send copies of the signal slightly shifted to create a constructive interference at the receiver.

Common beamforming methods belong in the following three categories: (i) analog; (ii) hybrid; (ii) digital beamforming [17]. In analog beamforming, the common RF source is split among the antennas, and the signals are "shaped" using analog phase shifters. In analog beamforming circuits, the receiver can only use one beam codeword, a shaping vector, at any given time. Thus, the communication may be significantly delayed until a suitable codeword is discovered [18]. In digital beamforming, the phase as well as the amplitude are controlled by digital baseband processing units. To fully utilize digital beamforming gain, a system is required to have a radio frequency front-end (RFFE) chain per each antenna element. One significant advantage of the digital beamforming is the receiver can use multiple beam codewords simultaneously to discover the best beam codeword and utilize the highest gain for specific communication channel [19]. On the other hand, the circuit design process is significantly harder since multiple digital-to-analog (DAC) and analog-to-digital (ADC) converters are required at the transmitter and the receiver respectively. Furthermore, using multiple RFFE chains increases the power consumption of the circuit. The component choices made during the circuit design can further increase or decrease the power consumption [20]. Successful designs provide a much more suitable environment for energy efficient and high capacity wireless communications [20]. However, the component power consumption and the severity of the non-linearities are antagonist to each other and create a design trade-off that needs to be carefully engineered by the circuit designers.

The quality of a communication link is significantly impacted by the non-linearities introduced by the RFFE circuit. Fig.1 illustrates the input output power relationship. In a non-linear receiver, when the input power at the receiver antenna is low, the performance in characterized by the noise-figure and the ADC quantization step. Conversely, when the input power at the receiver antenna is high the performance is characterized by the non-linear distortion and the ADC quantization range [20].

We investigate the problem of recovering the pre-distorted signal from the observed samples at the output of RFFE circuit. We make novel use of deep learning methods to compensate for the non-linearities introduced by RFFE at the receiver.

Our main contributions are as follow:

- We create and host an open source dataset for potential researchers to experiment and reproduce our results.
- We build and train neural networks, that are available in our Github repository. We achieve up to 20 dB increase in the output SNR performance. Furhtermore, the highest increase we observe is indeed in the high power regime where the distortions are exacerbated.
- We make use of transfer-learning-like training method to achieve better performance. First, we train a general model over every input power value. Finally we further optimize a new model for each input power level. This way, we create a 'task specific' model for each potential input power regime, which performs significantly better than the generalized model for the task.

II. Related Work

Skrimponis *et al.* [20] investigated the trade-offs involved in the design of the radio frequency front-end (RFFE) devices for a large class of receivers. They developed a system-level simulation package using accurate models for the RFFE elements extracted from circuit-level simulations. The components included in their analysis are low-noise amplifiers (LNA), mixers, local-oscillators (LO), and analog-to-digital converters (ADC). As described in [20], these components introduce non-linear distortion that can severely deteriorate the the performance of the receiver. The authors describe the impact of the RFFE by a non-linear function $\Phi(\cdot)$. We investigate the problem of recovering the pre-distorted signal $\vec{x}$, from the observed samples of $\vec{y}$,

$$\vec{y} = \Phi(\vec{x}, \vec{\eta}), \tag{1}$$

where $\vec{\eta}$ is the noise with appropriate dimensions. In other words we search for $\Phi^{-1}(\cdot)$.

Inverse problems estimate the causal factors producing a set of results from the results or estimate the mechanism and rules that maps the inputs to the observations [21]. Inverse problems can be found within wide fields of science and technology. To name a few, inverse problems are useful in denoising applications in image processing [22], compressed sensing applications [23], radar applications for soil analysis [24], and image reconstruction [25].

The powerful methods of deep learning open a whole new track of research on inverse problems arising in computational imagining. For example, Ongie *et al.* [26] studied the problem of function inversion for imaging. They consider the recovery of an image using a series of, possibly noisy, measurements. If the prior distribution of the image of interest and the mapping of this forward measurement operator are known classical methods such as maximum likelihood (ML) and *maximum a posteriori* (MAP) estimation can be used to recover the original image. However, the nature of the "measurement" operator can be unknown (or partially known) which degrades the performance of the aforementioned classical methods. For this unknown or partially known cases of measurement operator, deep neural networks can be leveraged and be trained to learn the inverse function.

Zhu *et al.* [25] studied image reconstruction problem from sensor data. They make use of deep networks to learn the underlying "features" from the sensor data. Similarly, they use another network to map the features to images. Our problem is similar to theirs as we also search the inverse of an unknown function with various parameters tuning parameters. However, our system differs from theirs as we are not recovering an image from sensor data but we are recovering a vector from antenna measurements. We implemented adaptation of their supervised learning methodology, AUTOMAP framework, for our inverse problem. However, we found our fully connected neural network achieves a better signal recovery.

III. Technical Details

A. System Model

The end-to-end information transfer over wireless links can be modeled as a combination of cascaded functions applied to the input bit sequence. In wireless communication the information is mapped to the digital bit sequence $\vec{x_i}$. This mapping is done by adjusting real and imaginary power levels, or phases of the signals, i.e., constellations like QAM, 8-PSK, 64-QAM. Then, the bit sequence is passed through transmitter antenna block, essentially a non-linear function, and emitted by the help of the electromagnetic waves. In this work we focus on receiver side of the communication and assume the transmitter antenna has no distortions. Thus, the $\vec{x_i}$ is the emitted signal. The radiated signal is then passed through a possibly unknown channel w_i, essentially a linear function, and received in analog domain of the receiver,

$$\mathbf{r}_i = \mathbf{w}_i \, x_i + \mathbf{v}_i,$$

where $\mathbf{r}_i, \mathbf{w}_i, \mathbf{v}_i \in \mathbb{C}^{1\times n}$ and $\mathbf{v}_i$ is the thermal noise. The signal is measured at the receiver side after converted from analog domain to digital domain by the help of RFFE circuit. After a signal received through the sensors, antennas in this case, and is converted from analog domain to digital domain, the information in the signal is detected/estimated. However, the impact

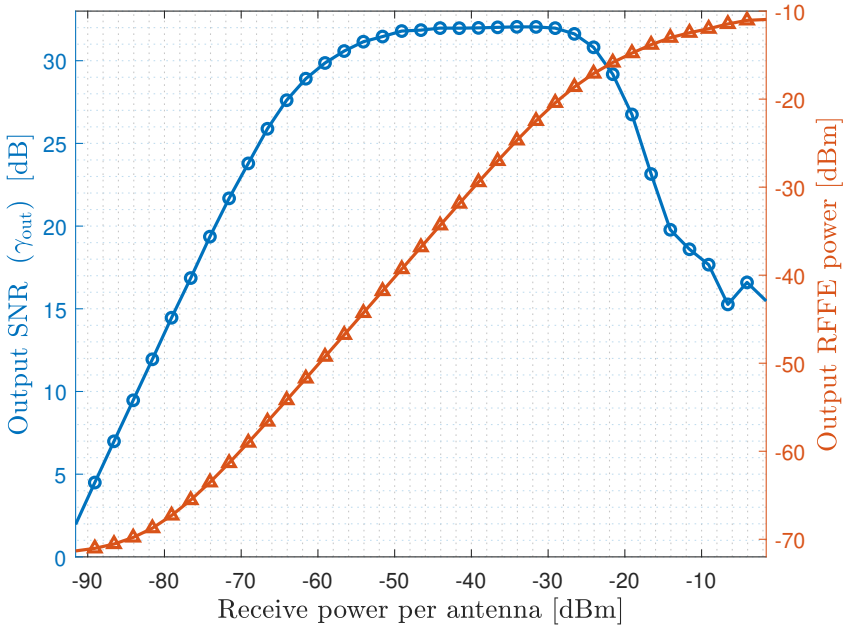

Fig. 1: (Orange) Input and output power relationship of a typical RFFE circuit. As the input power is close to the tails, the output power is saturated. This creates a non-linear effect in the output signal. (Blue) The output signal-to-noise ratio degradation due to saturation.

of the RFFE circuit is essentially a non-linear function. During the measurement process also an unknown noise is added to the signal. The non-linear function applied to the signal by the receiver, may degrade the performance of the transmission. Our interest lies at mitigating the bad effects of the receiver architecture. The actions of the RF front-end are described by a non-linear function Φ:

$$\mathbf{y}_i = \Phi(\mathbf{r}_i, \boldsymbol{\eta}_i).$$

Fig.1 illustrates the input output power relationship. In the high power, regime the output power can not match the high input power. Concomitantly, when the input power is increased, the non-linearities severely degrade the output SNR, which jeopardize the quality of communication. Recovering the original information of the signal using the observed measurements can be treated as an inverse problem.

B. Dataset

To generate the dataset we utilize *mmwComm*, an open-source MATLAB simulation package developed by Skrimponis *et al.* [20]. The package includes circuit-level models for RF components at 140 GHz, and the ability to simulate end-to-end systems using 3GPP NR waveforms. For the dataset we process the data at various input signal-to-noise ratio (SNR) levels. The dataset contains general information of the simulation parameters and the generated data:

- Number of SNR points, n_{snr}
- Number of generated symbols, n_x
- Number of receive antennas, n_{rx}
- Parameters of the components:
  1) **Low-noise amplifier:** (i) third-order input intercept point (IIP3); (ii) noise figure (NF); (iii) gain; (iv) power consumption.
  2) **IF Mixer:** (i) IIP3; (ii) NF; (iii) gain; (iv) power consumption. Note that the IIP3, NF, and gain are function of the input power from the LO.
  3) **Local Oscillator:** (i) output power; (ii) topology; (iii) power consumption.
  4) **A/D Converter:** (i) sample frequency; (ii) bits of resolution; (iii) power consumption.
- Original tx data, $\mathbf{x} \in \mathbb{C}^{n_x \times 1}$
- Channel, $\mathbf{w} \in \mathbb{C}^{n_x \times n_{\text{rx}}}$
- Received data on all power level, $\mathbf{r} \in \mathbb{C}^{n_x \times n_{\text{rx}} \times n_{\text{snr}}}$

Layer (type)	Output Shape	Number of Parameters
dense (Dense)	(None, 32)	1568
dense_1 (Dense)	(None, 64)	2112
dense_2 (Dense)	(None, 128)	8320
dense_3 (Dense)	(None, 256)	33024
dense_4 (Dense)	(None, 512)	131584
dense_5 (Dense)	(None, 1024)	525312
dense_6 (Dense)	(None, 512)	524800
dense_7 (Dense)	(None, 256)	131328
dense_8 (Dense)	(None, 128)	32896
dense_9 (Dense)	(None, 64)	8256
dense_10 (Dense)	(None, 32)	2080
Total parameters		1,401,280
Trainable parameters		1,401,280
Non-trainable parameters		0

(a) Neural network model summary from Tensorflow. This model considers $n_{\mathrm{rx}} = 16$ antenna elements.

Parameter	Value
Optimizer	Adam
Learning Rate	1e-3
AMSGrad	False
ϵ	1e-7
Epochs	10
Batch size	128
Kernel/Bias Initialization	Lecun Uniform
Kernel/Bias Regularization	None
Hidden Layer Activation	softsign
Output Layer Activation	linear
Loss Function	MSE

(b) Hyperparameters of the neural network.

TABLE I: Structure (a) and hyper-parameters (b) of our neural network. In total we have about 1.4M trainable parameters. We use Adam optimizer [27], and MSE as our loss metric.

- RFFE input power, $P_{\mathrm{in}} \in \mathbb{C}^{n_x \times n_{\mathrm{rx}} \times n_{\mathrm{snr}}}$
- RFFE ouput power, $P_{\mathrm{out}} \in \mathbb{C}^{n_x \times n_{\mathrm{rx}} \times n_{\mathrm{snr}}}$

C. RFFE Models

Using a combination of the RFFE components described below we create a set of receivers. Each receiver is characterized by a different non-linear function Φ. The receivers are indexed by a parameter θ.

1) Low-noise amplifier: The LNA performance is often characterized by the IIP3, since it relates to the third-order intermodulation distortion (IMD) to the input signal power. The authors designed and evaluated four LNA designs based on a $90\,\mathrm{nm}$ SiGe BiCMOS HBT technology. The LNAs have multiple stages and are based either on a common base (CB) or a hybrid common base and common emitter (CB/CE) topology.

2) Mixer: A mixer can be designed either as a passive or an active device. To power optimize this component the authors designed and evaluated five double-balanced active mixers based on the $90\,\mathrm{nm}$ SiGe BiCMOS HBT technology, and the conventional Gilbert cell design. The mixer performance can be measured based on the noise figure, gain, and IIP3, which are all characteristics that depend on the input LO power.

3) Local oscillator: This device is considered as a power-hungry device for fully-digital receivers, since we typically require one LO for each antenna. The authors propose an LO distribution model where the mixers in the same tile share a common LO driver. They use a power divider to split the input power from each driver to the input of the mixer. We explore LO power in the $-12\,\mathrm{dBm}$ - $0\,\mathrm{dBm}$ range.

4) A/D converter: This device is typically modeled as a quantizer. With the development of the 3GPP NR low-density parity check (LDPC) codes, modulation and coding schemes (MCSs) up to 64-QAM can be decoded reliably with less than $25\,\mathrm{dB}$. Based on the simulation results we consider $4-6$ bits of resolution for the ADCs.

D. 3GPP NR-Like Simulation

We consider a downlink system with a single NR base station (gNB), and a single user equipment (UE) device. The NR standard, originally developed for 5G, is very flexible and provides a natural baseline design for evaluating 6G systems. The gNB generates a physical downlink shared channel (PDSCH) that includes both information and control signals. The demodulation reference signals (DM-RS) and the phase tracking reference signals (PT-RS).

E. Neural Network Model

After a careful evaluation of various neural network architectures, convolutional neural networks (CNNs), fully connected networks (FCs), and hybrid of CNNs and FCs, we found that the FC network depicted in Fig. 2 achieves the highest SNR gain. This network can provide substantial performance improvement over the traditional method in [20]. The model is comprised by a set of fully-connected (FC) layers, with nonlinear activation for the hidden layers and linear activation for the output layer. The model contains 1,401,280 trainable parameters. In Table I(a) we show the model summary for $n_{\mathrm{rx}} = 16$ receive antennas.

In the repository[1] we have maintained the Jupyter notebooks with the other models. Amongst others, we investigated the use of a CNNs, a hybrid CNN and FC model (similar to the Automap framework), an auto-encoder, and custom FC layer

[1] https://github.com/skrimpon/nonlin

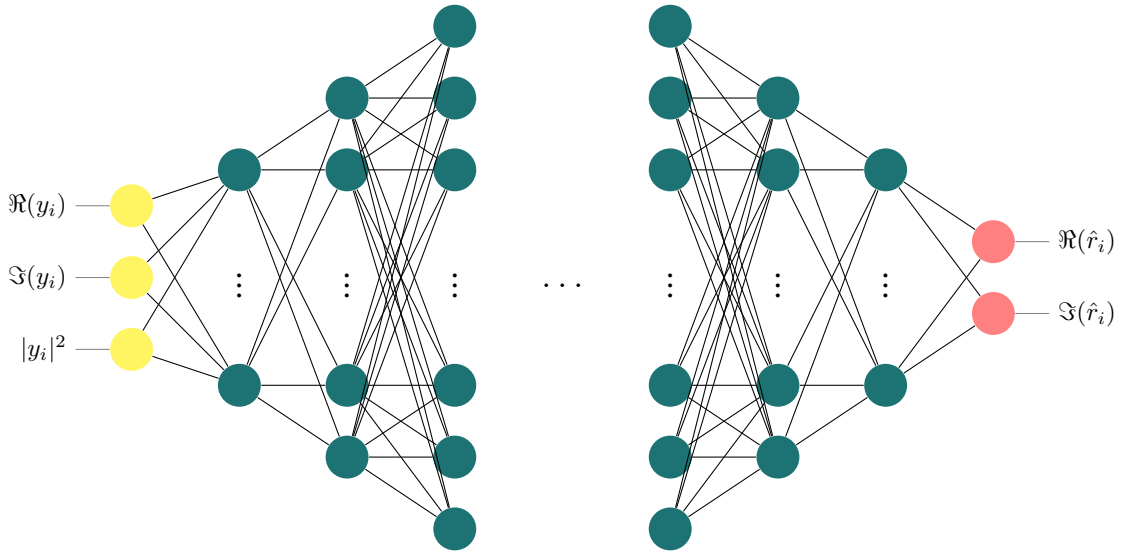

Fig. 2: Fully connected neural network architecture. The input data are first scaled into a higher dimension in order to allow the neural network learn more complex structures and then scaled back to the output dimension. The yellow nodes denote the input layer, the green nodes the hidden layers and the red nodes the output layer. Note that the input and output shape of the neural network depend on the number of antennas on the receiver.

with complex weights and biases. Most of these attempts had either the same or worse performance than the model in Fig. 2. Last time in our project update document, the hybrid CNN-FC architecture was presented. This time we used FC network, trained by a new approach inspired by transfer learning method. This new training approach helped us achieve better overall performance. The details are explained in Section III-F.

1) Input: The baseband processing of modern communication systems utilizes complex arithmetic. However, this is not the case for the majority of ML architectures. Thus, we either need to develop our own neural network layers, loss functions, gradients, that accept complex numbers or find a way to properly format the data into another representation.

There are two ways commonly used to split a complex number into real channels: (i) the exponential form that splits a number, $c = x + i\,y = r\,e^{i\theta}$, into the magnitude, $r = \sqrt{x^2 + y^2}$, and phase, $\theta = \tan^{-1}(\frac{y}{x})$; (ii) and using the real and the imaginary part of the number as two separate channels. The distortion model introduced by RFFE is expressed in amplitude (AM/AM) and phase (AM/PM) distortion. However, after extensive experimentation we found out that the option (ii) performs much better. We guess that is challenging for a neural network to understand the phase wraps around 2π.

The neural network takes the normalized value of the real and imaginary channels as the input. After exploring different options provided form *sklearn.preprocessing* for normalizing the input data (i.e., StandardScaler, Normalizer, PowerTransformer) we found that using our own method we were able to optimize the neural network performance. For a given vector $\mathbf{v}$ we convert each data point v_i to be in range $[0, 1]$ as follows,

$$\mathbf{v} = \frac{\mathbf{v} - \min(\mathbf{v})}{\max(\mathbf{v}) - \min(\mathbf{v})}. \tag{2}$$

This method is applied to both the output of the RFFE ($\mathbf{y}$) and the input ($\mathbf{r}$). Note that, even though it is simple, this method is generally used in mapping of grey-scale images to range between zero and one.

Beside the normalized data coming from the ADCs of all antennas we also give the output power of the RFFE as an input to the neural network. Since the mapping of the nonlinear function $\Phi(\cdot)$ depends on the input power of the signal we consider this as a critical parameter.

The size of the input vector depends on the number of antenna elements in the receiver circuit. The size of receiver antenna is known in the earlier stages of design and prototyping. Moreover, there are not many frequently used configurations. The industry currently focuses on $\{16, 32, 64, 128\}$ antenna elements. Thus, our method can be easily adapted for different number of antenna elements. In this work we illustrate 16 antenna configuration.

2) Output: The output of the neural network is the complex symbols for each antenna. The complex symbols are expressed again as two real channes. Thus, the total number of outputs is twice the number of receive antennas, $2\,n_{\text{rx}}$.

3) Loss Function: Since, our model is trying to approximate $\vec{r}$ as close as possible we have decided to use the mean squared error (MSE) as a loss function. Beside MSE, we use the post-equalization SNR as a performance metric which is widely used for evaluating the performance of communication systems. More details about the output SNR are discusses in Section IV-B.

As a future work, we would like to explore a "Tikhonov like" regularization [28] which is reported to be effective in other inverse problems.

4) Hyperparameter Selection: For the final hyperparameter selection phase, we used the NYU high performance computing (HPC) clusters. We were able to generate models and explore various optimizers (i.e., SGD, Adam) with their parameters, data normalization methods (i.e., standard scale, normalizer), activation functions (i.e., linear, relu, tanh), kernel/bias initiliazers (i.e., lecun uniform), and kernel/bias regularizers (i.e., l1, l2). In Table I(b) we show the critical hyper-parameters used in our implementation.

F. Training Details

To successfully train the neural network we have developed an algorithm inspired by transfer learning. The algorithm involves multiple independent datasets and processing data in different power levels. Initial goal is for the model to learn the inverse function of the RFFE, instead of specific characteristics of the waveform or the channel. Finally, we fine tune one model for each input signal power level, so that the models also learn the specific characteristics of the input signal power levels. This is where our transfer learning approach comes into effect.

1) Dataset generation: The first step is to generate the datasets. These datasets are used for training and testing. To generate the datasets we use MATLAB R2021a and the scripts in **nonlin/project/createDataset** and **nonlin/project/mmwComm**. We generate 20 datasets, each with 10,000 samples for a total of 200,000 samples. For a given receiver front-end configuration i, the datasets are saved in **nonlin/datasets/rx_i**.

2) Model creation and Pre-processing: To create the neural network model described above we utilize Tensorflow and Keras. To preprocess the data we need to first split them based on the power level. For each SNR we use (2) to normalize the data in [0,1]. Then, we split data in train and test with 160,000 samples of train and 40,000 samples of test data.

3) Training:

- We start the training from the highest input power (higher distortion) to lowest input power (lower distortion). Since, we would like to emphasize on learning the nonlinear relationships and use this as a basis for next the SNR level. For each input SNR level S_i, we train for 10 epochs and we save the model and move on to next highest input SNR level S_{i-1}. After training over all S values we repeat this process 5 times. So, our model is initially trained 50 epochs for each sample. Then, we select the models that had the best overall performance for each level of input SNR S_i.
- Now that we decided on a promising model for each input SNR level S_i, we train these model specifically for their target SNR value. We use similar approach to step 1 but this time we only use SNR levels in the neighborhood of target SNR S_i. This concept of retraining an already trained model is inspired by transfer learning. The execution time can be significantly reduced by restraining neighborhood range into the nearest neighbor only. Since, data from nearby SNRs have similar distortion. The training procedure takes about 20 hours on a GPU with cuda score of 7.5. Utilizing the NYU HPC resources, the execution time can be significantly reduced.

4) Further Optimizations: We would like the neural network model to be transparent of the given the data. Using the basic training method we can use the already trained model to learn the nonlinear/linear relationship on other datasets. Furthermore, to learn more efficiently the nonlinear behavior of the RFFE the dataset can be generated to contain more datapoints in the high input power SNRs rather than the lower input power.

IV. Results

To organize more efficiently the development of our project we created a git repository[2]. In this repository we include *mmwComm* as a submodule. We developed some MATLAB scripts to generate the datasets for a given RFFE configuration. The dataset is saved in a set of *.csv* files. These files can then be parsed in python using the *Pandas* library. We include a demo Jupyter notebook that introduces the dataset to a potential user. We also include a Jupyter notebook that showcases the results presented below using the trained models.

A. Symbol-Level Simulation

As a first step to the evaluation of our system, we create a symbol-level simulation. We create a single-input-multiple-output (SIMO) system with $n = 16$ receive antenna and one tx antenna. The transmitter generates $m = 10^5$ complex baseband symbols, $\mathbf{x} \in \mathbb{C}^{m\times 1}$, as i.i.d. based on a given distribution. The transmit symbol can also belong to a constellation (i.e., QAM, QPSK). The signal goes through a channel, that is considered known at the receiver. The channel matrix $\mathbf{W} \in \mathbb{C}^{m\times n}$ is generated as i.i.d. based on a given distribution. Thus, the received symbols at each antenna i are:

$$\mathbf{r}_i = \mathbf{w}_i\, x_i + \mathbf{v}_i,$$

[2]https://github.com/skrimpon/nonlin

where $\mathbf{r}_i, \mathbf{w}_i, \mathbf{v}_i \in \mathbb{C}^{1\times n}$ and $\mathbf{v}_i$ is the thermal noise. The actions of the RF front-end are described by a non-linear function Φ:

$$\mathbf{y}_i = \Phi(\mathbf{r}_i, \boldsymbol{\eta}_i)$$

B. Output SNR

Following the analysis in [20], we use output-SNR as metric to characterize the overall system performance. We first apply the beamforming with the known (or estimated) channel matrix $\mathbf{W}$ to get an estimate of the received symbols $\hat{\mathbf{x}}$

$$\hat{x}_i = \frac{\mathbf{y}_i \mathbf{w}_i^H}{\|\mathbf{w}_i\|^2}$$

To measure the output-SNR we express $\hat{\mathbf{x}}$ as a linear estimate ($\hat{\mathbf{x}} = \mathbf{a}\,\mathbf{x} + \mathbf{d}$) with $d \sim CN(0, \mathbb{E}|\hat{x} - a\,x|^2)$. First we find a,

$$a = \frac{\mathbb{E}\left[\hat{\mathbf{x}}^H \mathbf{x}\right]}{\mathbb{E}\|\mathbf{x}\|^2}$$

Then, we find the noise variance of d:

$$\sigma_d = \mathbb{E}\left[|\hat{x} - a\,x|^2\right]$$

Finally, we measure the output-SNR:

$$\gamma_{out} = \left(\frac{|a|\,\sigma_x}{\sigma_d}\right)^2$$

C. Baseline Performance vs. Non-linearities

In Fig.1 we show the baseline performance and the input-output power relationship for a given receiver front-end. Initially, the performance is increasing linearly. Then, the performance saturates because of the ADC quantization. As the nonlinear distortion from the RFFE increases the performance starts to radically drop.

In this system, we use a sample frequency $f_s = 122.88$ MHz as system bandwidth, with an overall noise figure of the system as, $\mathrm{NF} = 11.50$ dB. To calculated the noise floor in mW we use the well-known formula,

$$N = 10\log_{10}(f_s\,k\,T) + \mathrm{NF}$$

where $k = 1.3807e^{-23}$ J/K is the boltzmann constant, and $T = 290$ K the ambient temperature. Thus, the noise floor is at $-81.6\,\mathrm{dBm}$. Thus, the input-output power relation looks to be non-linear when the signal is below the noise floor but that's just a simulation artifact. Since, we can see that the performance grows linearly.

D. Performance Comparison

Fig. 3 illustrates that, our design performs much better than the currently utilized method in [20]. Especially in the high-input power regime we can see significant gains up to $20\,\mathrm{dB}$. It is worth noting that the performance is saturated around $35\,\mathrm{dB}$ because of the ADC resolution. Thus, we expect that for a higher ADC resolution we might get an even better performance improvement.

We consider that our results have significant value for the research community. Since, they can enable the design of RFFE with severe nonlinear distortion without impacting the overall performance. This could result in optimization strategies for understanding the power consumption trade-offs between the analog and digital processing. This could eventually result in the design of an optimal receiver architecture that can simultaneously achieve low power consumption as well as a high performance.

E. Reproducibility: Training a model or Using the trained model

Currently our GitHub[3] repository contains all of our code-base. For your convenience we provide a Jupyter notebook[4] in colab. This notebook can be used to reproduce our results, once the already trained models are given. Currently our trained models are saved in the Google Drive and can be easily downloaded by the provided notebook. This script provides a testing method on already trained models. For training the neural network a user should use the python/MATLAB scripts in our repository. The details are described in the repository.

[3] https://github.com/skrimpon/nonlin

[4] https://colab.research.google.com/drive/1Cg6ToHTp2Wmk7j7j$_d$9oViONBlHZD6K6?usp = sharing

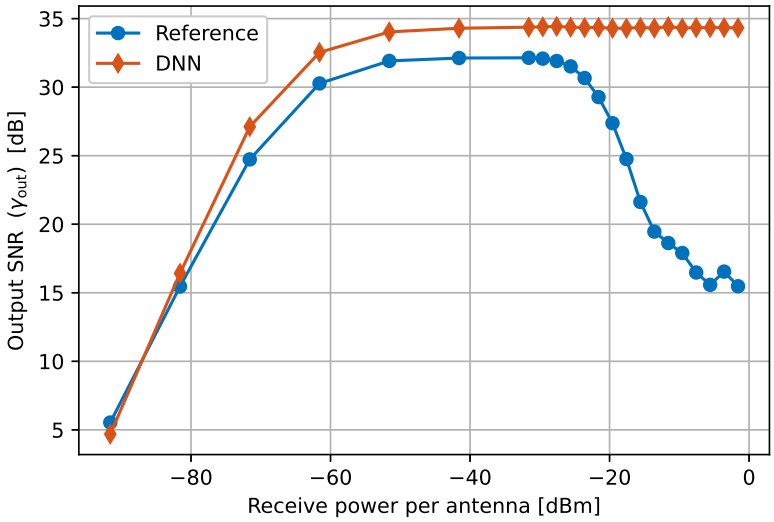

Fig. 3: The performance of our DNN compared with the state of the art baseline. We observe a performance improvement up to 20 dB compared to the baseline. More importantly, our model significantly improves the performance specifically in the non-linear region.

V. Conclusions

The signal distortions at the RFFE circuit limits the performance of wireless communications. The performance degradation is much more prominent in high input power regime where the output power can not match to the input power. We showed that DNNs are promising to alleviate the non-linear distortions introduced at the receiver antenna chain. Our method achieves up to 20 dB more SNR as compared to baseline method. Furthermore, the highest increase we observe is indeed in the high power regime where the distortions are exacerbated. We believe our work paves the way for the use of power efficient but less precise radio components. This will significantly expand the design space and relax the design procedure. We believe correcting the distortions using DNNs can be a way for future power efficient RF circuits, targeting freqencies beyond 100 GHz.

A. Goal evaluation and Future work

Through this project we managed to accomplish the following goals:

- Make use of a deep learning algorithms to invert the distortions introduced by the non-ideal hardware at the radio frequency chain and recover an approximate of the original signal $\vec{r}$.
- Develop scripts for generating datasets for nonlinear receiver front-ends with SIMO and/or 5G NR data.
- Develop a general methodology to improve the performance of receiver front-ends.
- Through this project we learned various deep learning algorithms and techniques

One of our initial goals is this work to evolve into a peer-reviewed scientific publication. Of course, this could not be accomplished in the short time frame of this class. However, we are happy with the current results and we think that they are very promising for us to achieve this goal in the following months.

As future goals, we would like to explore more power efficient neural network architectures. Since, power consumption is one of the most difficult technical challenge in developing commercial mobile devices above 100 GHz. For instance, pruning [29] seems to be an interesting and promising way to reduce the network size. Moreover, we search for shallow but high performing networks with a few layers, so that the performance degradation is still tolerable. Similar to [30], [31], we desire to explore fix-point arithmetic options to reduce the power consumption of the digital components.

References

[1] Qualcomm, "Leading the world to 5G," Qualcomm, Tech. Rep., Feb. 2016. [Online]. Available: https://www.qualcomm.com/media/documents/files/qualcomm-5g-vision-presentation.pdf

[2] Z. Zhang, Y. Xiao, Z. Ma *et al.*, "6G wireless networks: Vision, requirements, architecture, and key technologies," *IEEE Veh. Technol. Mag.*, vol. 14, no. 3, pp. 28–41, 2019.

[3] T. S. Rappaport, S. Sun, R. Mayzus *et al.*, "Millimeter wave mobile communications for 5G cellular: It will work!" *IEEE Access*, vol. 1, pp. 335–349, May 2013.

[4] W. Saad, M. Bennis, and M. Chen, "A vision of 6G wireless systems: Applications, trends, technologies, and open research problems," *arXiv preprint arXiv:1902.10265*, Feb. 2019.

[5] I. K. Jain, R. Kumar, and S. Panwar, "Driven by capacity or blockage? A millimeter wave blockage analysis," in *Proc. of IEEE ITC*, Sep. 2018.

[6] I. K. Jain, R. Kumar, and S. Panwar, "The impact of mobile blockers on millimeter wave cellular systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 4, pp. 854 – 868, Feb. 2019.

[7] K. Allen, N. DeMinco, J. Hoffman, Y. Lo, and P. Papazian, "Building penetration loss measurements at 900 MHz, 11.4 GHz, and 28.8 GHz," NTIA, Tech. Rep., May 1994.

[8] G. R. MacCartney, T. S. Rappaport, and S. Rangan, "Rapid fading due to human blockage in pedestrian crowds at 5G millimeter-wave frequencies," in *Proc. of IEEE GLOBECOM*, Dec. 2017.

[9] T. S. Rappaport, Y. Xing, O. Kanhere *et al.*, "Wireless communications and applications above 100 GHz: Opportunities and challenges for 6G and beyond," *IEEE Access*, vol. 7, pp. 78 729–78 757, Jun 2019.

[10] D. Tse and P. Viswanath, *Fundamentals of wireless communication*. Cambridge University Press, 2005.

[11] J. M. Jornet and I. F. Akyildiz, "Channel modeling and capacity analysis for electromagnetic wireless nanonetworks in the terahertz band," *IEEE Trans. Commun.*, vol. 10, no. 10, pp. 3211–3221, Aug. 2011.

[12] A. Goldsmith, *Wireless communications*. Cambridge university press, 2005.

[13] W. Roh, J.-Y. Seol, J. Park, B. Lee, J. Lee, Y. Kim, J. Cho, K. Cheun, and F. Aryanfar, "Millimeter-wave beamforming as an enabling technology for 5g cellular communications: theoretical feasibility and prototype results," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 106–113, 2014.

[14] A. Alkhateeb, O. El Ayach, G. Leus, and R. W. Heath, "Channel estimation and hybrid precoding for millimeter wave cellular systems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 5, pp. 831–846, 2014.

[15] O. E. Ayach, S. Rajagopal, S. Abu-Surra, Z. Pi, and R. W. Heath, "Spatially sparse precoding in millimeter wave mimo systems," *IEEE Transactions on Wireless Communications*, vol. 13, no. 3, pp. 1499–1513, 2014.

[16] R. W. Heath, N. González-Prelcic, S. Rangan, W. Roh, and A. M. Sayeed, "An overview of signal processing techniques for millimeter wave mimo systems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 3, pp. 436–453, 2016.

[17] M. N. Hamdy, "Beamformers explained," Commscope, Tech. Rep., 2020. [Online]. Available: https://www.commscope.com/globalassets/digizuite/542044-Beamformer-Explained-WP-114491-EN.pdf

[18] C. N. Barati, S. A. Hosseini, M. Mezzavilla, T. Korakis, S. S. Panwar, S. Rangan, and M. Zorzi, "Initial access in millimeter wave cellular systems," *IEEE Transactions on Wireless Communications*, vol. 15, no. 12, pp. 7926–7940, 2016.

[19] S. Dutta, C. N. Barati, D. Ramirez, A. Dhananjay, J. F. Buckwalter, and S. Rangan, "A case for digital beamforming at mmwave," *IEEE Transactions on Wireless Communications*, vol. 19, no. 2, pp. 756–770, 2019.

[20] P. Skrimponis, N. Hosseinzadeh, A. Khalili, E. Erkip, M. J. W. Rodwell, J. F. Buckwalter, and S. Rangan, "Towards energy efficient mobile wireless receivers above 100 GHz," *IEEE Access*, vol. 9, pp. 20 704–20 716, 2021.

[21] T. Ogawa, Y. Kosugi, and H. Kanada, "Neural network based solution to inverse problems," in *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98CH36227)*, vol. 3, 1998, pp. 2471–2476 vol.3.

[22] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image Processing*, vol. 15, no. 12, pp. 3736–3745, 2006.

[23] M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright, "Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 586–597, 2007.

[24] Y. Oh, K. Sarabandi, and F. T. Ulaby, "An empirical model and an inversion technique for radar scattering from bare soil surfaces," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 30, no. 2, pp. 370–381, 1992.

[25] B. Zhu, J. Z. Liu, B. R. Rosen, and M. S. Rosen, "Image reconstruction by domain transform manifold learning," *CoRR*, vol. abs/1704.08841, 2017. [Online]. Available: http://arxiv.org/abs/1704.08841

[26] G. Ongie, A. Jalal, C. A. Metzler, R. G. Baraniuk, A. G. Dimakis, and R. Willett, "Deep learning techniques for inverse problems in imaging," *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 1, pp. 39–56, 2020.

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[28] A. N. Tikhonov, "On the stability of inverse problems," *Proceedings of the USSR Academy of Sciences*, vol. 39, pp. 195–198, 1943.

[29] D. Blalock, J. J. G. Ortiz, J. Frankle, and J. Guttag, "What is the state of neural network pruning?" 2020.

[30] P. Skrimponis, S. Dutta, M. Mezzavilla, S. Rangan, S. H. Mirfarshbafan, C. Studer, J. Buckwalter, and M. Rodwell, "Power consumption analysis for mobile mmWave and sub-THz receivers," in *2020 2nd 6G Wireless Summit (6G SUMMIT)*, 2020, pp. 1–5.

[31] P. Skrimponis, S. H. Mirfarshbafan, C. Studer, and S. Rangan, "Power efficient multi-carrier baseband processing for 5G and 6G wireless," in *Proc. IEEE Asilomar Conference on Signals, Systems, and Computers (to appear)*, 2020.