Efficient Training of Deep Classifiers for Wireless Source Identification using Test SNR Estimates

Xingchen Wang, *Student Member, IEEE*, Shengtai Ju, *Student Member, IEEE*, Xiwen Zhang, *Student Member, IEEE*, Sharan Ramjee, *Student Member, IEEE*, and Aly El Gamal, *Senior Member, IEEE*

***Abstract*—We investigate the potential of training time reduction for deep learning algorithms that process received wireless signals, if an accurate test Signal to Noise Ratio (SNR) estimate is available. Our focus is on two tasks that facilitate source identification: 1- Identifying the modulation type, 2- Identifying the wireless technology and channel index in the 2.4 GHZ ISM band. For benchmarking, we rely on a fast growing recent literature on testing deep learning algorithms against two well-known synthetic datasets. We first demonstrate that using training data corresponding only to the test SNR value leads to dramatic reductions in training time - that can reach up to 35x - while incurring a small loss in average test accuracy, as it improves the accuracy for low test SNR values. Further, we show that an erroneous test SNR estimate with a small positive offset is better for training than another having the same error magnitude with a negative offset. Secondly, we introduce a greedy training SNR Boosting algorithm that leads to uniform improvement in test accuracy across all tested SNR values, while using only a small subset of training SNR values at each test SNR. Finally, we discuss, with empirical evidence, the potential of bootstrap aggregating (Bagging) based on training SNR values to improve generalization at low test SNR values with scarcity of training data.**

***Index Terms*—Deep Learning for Wireless, Modulation Classification, Channel Identification, SNR Boosting, SNR Bagging.**

I. Introduction

With the advent of fifth generation (5G) wireless communication networks, emerging applications like virtual reality (VR) and internet of things (IoT)-related applications are expected to rely more heavily on an extremely fast and reliable mobile network that can operate in complex and dynamic environments. Deep learning can potentially become an essential part in the design of these networks, because of the difficulty of modeling these environments as well as the small time scale of wireless communications that allows for rapidly collecting large datasets. More specifically, deep learning algorithms will be strong candidates for autonomous communication systems that require little computational and control overhead, and form an intelligent understanding of the spectrum, which starts with the basic task of identifying the source(s) of transmission for a given received wireless signal.

The problem of recognizing the modulation type of a received signal is important for interference source identification, and for reducing control overhead by enabling frequent modulation and coding scheme (MCS) adaptation to the changing environment. In [1] and [2], a synthetic dataset based on the GNU Radio software package was introduced to initiate the investigation of deep neural network architectures and optimization algorithms that are suitable for the task of recognizing one out of ten modulation types, including both analog and digital types. In [3], multiple architectures were presented that deliver state of the art accuracy results for this synthetic dataset. In [4], the problem of designing fast training algorithms for these architectures was considered, and promising results were presented using candidate algorithms that rely on training with only a subset of the samples available in each input vector (subsampling) as well as a subset of vectors that correspond to only a single value for the signal to noise ratio (SNR).

In [5], the problem of channel identification in the 2.4 GHZ ISM band was considered through a synthetic dataset that the authors introduced. The dataset contains received signals corresponding to 15 different channels of WiFi, Bluetooth, and ZigBee. Efficient training algorithms for this problem were investigated in [6] through training with only a subset of frequency bands, a subset of input vector samples, and a subset of SNR values.

In this work, we demonstrate the feasibility of effective and fast training of deep classifiers for wireless source identification tasks, in presence of a good test SNR estimate. We use the aforementioned datasets for modulation classification and channel identification to validate the proposed methods and obtained insights. We first start by improving the preliminary results on training SNR selection of [4] and [6], obtained through training with only the test SNR value. We then show that if we can only estimate a test SNR range, it is better to train with optimistic estimates. We then present a training *SNR Boosting* algorithm that identifies a small set of training SNR values that lead to the best accuracy for every given test SNR. Not only does it lead to faster training, but the identified training set through SNR Boosting is shown to deliver superior performance to training with all available data for both considered tasks. We finalize our discussion of training algorithms that are based on SNR selection, by investigating the potential of Bagging (see e.g., [7, Chapter 7]).

II. Problem Setup

A. Problem Description and Considered Training Algorithms

Given an incoming received signal, provided through one of the datasets described in Section II-B, our goal is to maintain

X. Wang, S. Ju, X. Zhang, S. Ramjee, and A. El Gamal are with the Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA. Email: {wang2930, ju10, zhan2977, sramjee, elgamala}@purdue.edu.

TABLE I: The considered 3 architectures for modulation classification. The convolutional layers column indicates the number of feature maps and kernel size of each layer. The dense layers column indicates the input and output size of each layer. The LSTM column indicates the number of recurrent cells. The last column indicates the number of residual stacks.

Architecture	Activation Functions	Convolutional Layers	Dense Layers	LSTM	R-Stacks
CNN	ReLU, Softmax	$256\ 3*1$, $80\ 3*2$	$10560*256$, $256*10$		
ResNet	ReLU, SeLU, Softmax		$128*128$, $128*128$, $128*10$		5
CLDNN	ReLU, Softmax	$256\ 3*1$, $256\ 3*2$, $80\ 3*1$, $80\ 3*1$	$50*256$, $256*10$	50	

TABLE II: The considered 3 deep neural network architectures for channel identification.

Architecture	Activation Functions	Convolutional Layers	Dense Layers	LSTM	R-Stacks
CNN-1	ReLU, Softmax	$256\ 3*1$, $256\ 3*2$	$31744*1024$, $1024*15$		
CNN-2	ReLU, Softmax	$256\ 3*1$, $256\ 3*1$	$32768*1024$, $1024*15$		
ResNet	ReLU, Softmax		$128*128$, $128*128$, $128*15$		5
CLDNN	ReLU, Softmax	$256\ 3*1$, $256\ 3*2$	$31744*1024$ (before LSTM), $512*256$, $256*15$	256 (2-dim)	

TABLE III: Average accuracy and training times for modulation classification

Architecture	Training data type	Accuracy (%)	Time per epoch (s)	Number of epochs	Training time (s)
ResNet	Single SNR	60.46	1.00	49.65	49.65
	All SNR	63.00	27.50	48.00	1320.00
CNN	Single SNR	54.41	1.00	108.35	108.35
	All SNR	56.48	14.00	208.00	2913.00
CLDNN	Single SNR	51.28	1.25	70.80	88.50
	All SNR	59.87	39.00	80.00	3120.00

TABLE IV: Average accuracy and training times for channel identification

Architecture	Training data type	Accuracy (%)	Time per epoch (s)	Number of epochs	Training time (s)
ResNet	Single SNR	88.35	1.51	24.14	36.41
	All SNR	89.42	19.43	15.00	291.45
CNN	Single SNR	90.19	0.98	34.52	33.84
	All SNR	89.69	22.02	12.00	264.24
CLDNN	Single SNR	89.44	1.23	65.57	80.92
	All SNR	89.83	18.56	10.00	185.60

a high classification accuracy while driving the total training time as low as possible for computational efficiency[1]. We consider two classification tasks: 1- Identifying one out of ten modulation types for the RadioML2016.10b dataset of [1]. 2- Identifying one out of 15 channels for the channel identification dataset of [5]. Both datasets are synthetic and based on simulating random channel and hardware imperfections, that are difficult to model in closed form, and the descriptions of the random generators of these imperfections are available in [1] and [5]. In the sequel, we consider the following three training scenarios: (a) Training with all available data, which represents our initial benchmark. (b) Training only with data of the same SNR value as a test SNR estimate, as described in Section IV-A. (c) Training with a selected subset of available data corresponding to specific SNR values, which are determind through an SNR Boosting algorithm, detailed in Section IV-B.

[1]Code for modulation classification was approved for publication at IEEE Code Ocean at https://codeocean.com/capsule/7188167/tree/v1. Code for channel identification was approved for publication at IEEE Code Ocean at https://codeocean.com/capsule/0a545fca-d0c3-410b-bc37-f253e585ac39/tree

B. Datasets

To demonstrate the universality of the presented results, we use two datasets on tasks that are strongly related to wireless source identification, a modulation classification dataset and a channel identification dataset. We use the RadioML2016.10b dataset introduced in [1] and [2] for the task of modulation classification. Ten widely used modulation types are chosen; eight digital and two analog modulation types. These consist of BPSK, QPSK, 8PSK, QAM16, QAM64, BFSK, CPFSK,

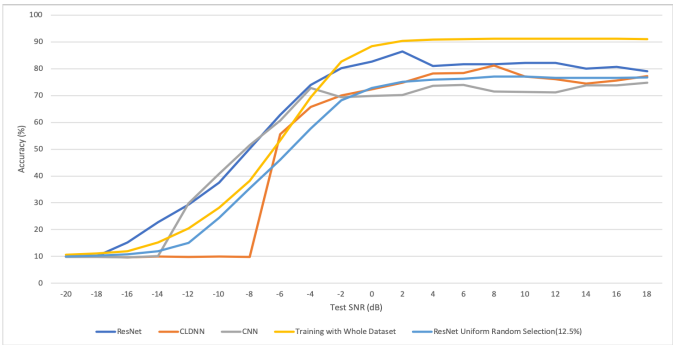

Fig. 1: Single SNR Selection with Considered Architectures for Modulation Classification (ResNet is used for "Training with Whole Dataset")

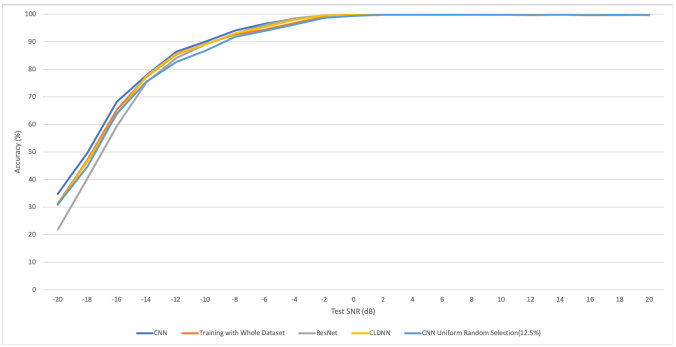

Fig. 2: Single SNR Selection with Considered Architectures for Channel Identification (CNN is used for "Training with Whole Dataset")

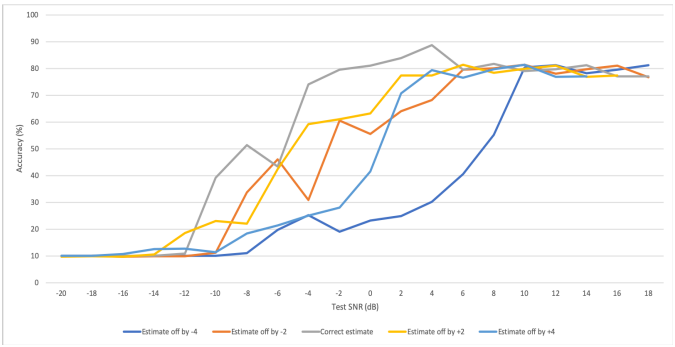

Fig. 3: SNR Sensitivity with ResNet for Modulation Classification

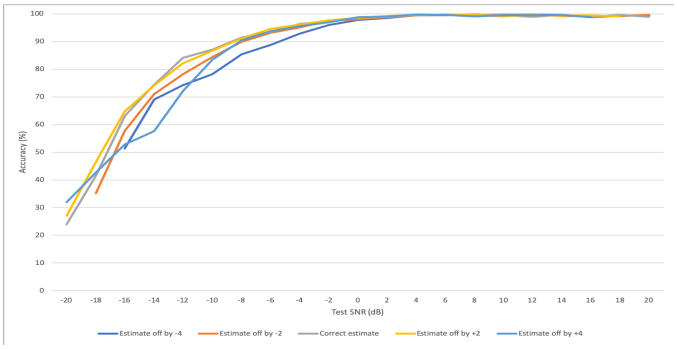

Fig. 4: SNR Sensitivity with CNN for Channel Identification

and PAM4 for digital modulations, and WB-FM, and AM-DSB for analog modulations. The dataset consists of 160,000 sample vectors; each consisting of 128 2-dimensional (real and imaginary I/Q format) samples, taken every 1 μs from a baseband received signal.

The considered channel identification dataset is introduced in [5], and contains in total 225,225 sample vectors for 15 classes. There are 10 Bluetooth channels with center frequencies in the range 2422-2431 MHz, spaced every 1 MHz, and each channel has a width of 1 MHz. Also, there are three WiFi channels with center frequencies of 2422, 2427, and 2432 MHz, and each has a width of 20 MHz. Finally, there are two Zigbee channels with center frequencies 2425 and 2430 MHz, and each has a width of 2 MHz. For the WiFi frames, the Physical Layer Mode is varied between CCK, PBCC, and OFDM. For the Bluetooth frames, the Transport Mode is varied between ACL, SCO, and eSCO. For the Zigbee frames, the ACK-frame is used. Each sample vector consists of 128 I/Q samples, corresponding to 12.8 μs (a sample is taken every 0.1 μs). The I/Q samples of each sample vector are also transformed into the frequency domain by using the Fast Fourier Transform (FFT), as we believe that it facilitates identifying the channel through the occupied frequency range.

Finally, both datasets are split equally among all considered classes, and are also split equally among all SNR values in the considered range with a minimum of -20 dB and a maximum of 18 dB for modulation classification and 20 dB for channel identification, and a step size of 2 dB.

III. Deep Neural Network Algorithms

We consider three different architecture types: A Convolutional Neural Network (CNN), Residual Network (ResNet), and a Convolutional Long Short-term Deep Neural Network (CLDNN). We clarify below the design details of each architecture, and a summary is provided in Table I and Table II.

A. Modulation Classification

The CNN consists of two convolutional layers with the first layer having 256 kernels; each of size of 3×1. The second convolutional layer has 80 3×2 kernels. The two dense layers have sizes of 1024×256 and 256×10, respectively. Activation functions used in the CNN are ReLU for hidden layers and Softmax for the output layer. The ResNet architecture used for single SNR selection contains 3 residual stacks. Each residual stack consists of the following layers: a 1-D convolutional layer with kernel size 1 and linear activation function followed by a batch normalization layer; after batch normalization, two residual units are added, and finally, a max pooling layer is added. The residual unit contains two convolutional layers with kernel size 5, each followed by a batch normalization layer. The first convolutional layer in the residual unit uses ReLU as activation function while the second uses a linear activation function. A shortcut unit is then added for the residual unit

connecting the beginning and the end. The ResNet architecture used in our greedy boosting algorithm has 5 residual stacks with the exact same setup just mentioned. The dimension of the first dense layer for our 3-stack ResNet is 512 and the dimension of the first dense layer for our 5-stack ResNet is 128. The ResNet dense layers have Scaled Exponential Linear Unit (SeLU) activation. For the CLDNN, the LSTM layer follows all convolutional layers, and precedes all dense layers.

B. Channel Identification

For the CNN used for single SNR selection, a dropout layer is added after the second convolutional layer. Then we reshape the output of the dropout layer and feed it through a fully connected layer followed by ReLU and another dropout layer. For the network used for SNR boosting, batch normalization is applied after each convolutional layer. The dropout layer after the second convolution layer is removed in SNR boosting. The CNN architectures used for single SNR selection are labeled CNN-1 and CNN-2 in Table II, respectively. The ResNet used for channel identification has 5 residual stacks. The residual stacks are similar to those used for modulation classification, but both convolutional layers in a residual unit have ReLU activations. For the CLDNN, one dense layer succeeds convolutional layers and precedes the LSTM layer as illustrated in Table II, and the other two dense layers follow the LSTM layer.

C. Programming Environment and Hyperparameters

We used a GPU server with a Tesla P100 GPU and 16 GB of memory. For modulation classification, we used the same hyperparameters and optimization algorithms as in [4]. For channel identification, we used a dropout rate of 0.6 for single SNR selection, and 0.8088 for greedy SNR boosting. All other parameters are kept the same as [6].

IV. Results

A. Single SNR selection

1) Modulation Classification: When training with a perfect test SNR estimate, we observe from the results in Figure 1 that the accuracy is typically higher than using the whole training set at low SNR values, and lower at high SNR values. The average accuracy suffers from a slight drop as illustrated in Table III. The intuition behind the improved performance at low SNR values is that when training with the whole dataset, the network focuses on patterns that are not relevant to the noisy regime corresponding to the test SNR.

As we are using only 5% of the training dataset, the training time is reduced by 25-35x, as can be observed from Table III. We observe from Figure 1 that when using a larger portion of 12.5% of the training dataset, but uniformly distributed across all SNR values, the test accuracy is uniformly (at all test SNR values) lower than the single SNR selection strategy.

Since the ResNet architecture was found to deliver the best performance for modulation classification, we restrict our attention to it for the remaining experiments on this task. In order to determine the training SNR selection strategy when the test SNR estimate can only specify a range of values, we tested the impact of training with an SNR that is lower/upper than the true value by 2 and 4 dB. As shown in Figure 3, the optimistic estimate is almost always better to train with.

2) Channel Identification: Identical observations to the modulation classification task hold, with the following exceptions: 1- Starting from test SNR of 0 dB, we obtain almost perfect classification accuracy, and hence, the stated observations are evident only at lower test SNR values. 2- The CNN architecture outperforms the considered others, and hence, aside from single SNR selection using perfect estimates, we restrict our attention to this architecture. 3- The penalty incurred due to an inaccurate test SNR estimate used exclusively for training is not as significant as that of the modulation classification task.

B. SNR Boosting

Pseudocode for the SNR Boosting Algorithm is shown below in Algorithm 1. It is important to note that the validation set, consisting of 20% of the set available for training, is used only for the selection of the training SNR values, and is later added back to the training set, when using these values to construct the training set. As can be seen from Figures 5 and 6, training with the selected SNR set successfully boosts the model performance. With the selected SNR set from our SNR boosting algorithm, the obtained accuracy is uniformly higher than - or very close to - that obtained when using the whole dataset. Noting that the training time is reduced significantly as the set typically consists of 3-5 SNR values, out of 20, for modulation classification, and 1-3 SNR values, out of 21, for low test SNR channel identification. The SNR values selected for training are marked in Figures 7 and 8.

Algorithm 1 SNR Boosting Algorithm

Input: Target SNR
Output: Selected Training SNR List

```
while Accuracy increase > 1 and remaining SNR list not empty do
    for single SNR in remaining SNR list do
        append single SNR training data to current set (without
          updating current set), train model, and evaluate on
          validation set
        if current accuracy > previous accuracy then
            appending SNR = single SNR
    if accuracy improvement then
        add appending SNR to selected training SNR list
        remove appending SNR from remaining SNR list
        set Accuracy increase
```

V. Discussion: SNR Bagging

In an attempt to improve generalization performance, we implemented a bootstrap aggregating (Bagging) algorithm that relies on training three identical models using independently and uniformly sampled training sets from the sets corresponding to the test SNR value as well as the two

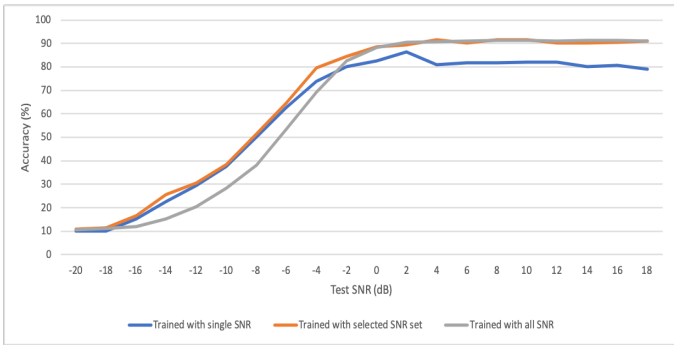

Fig. 5: SNR Boosting (ResNet) for Modulation Classification

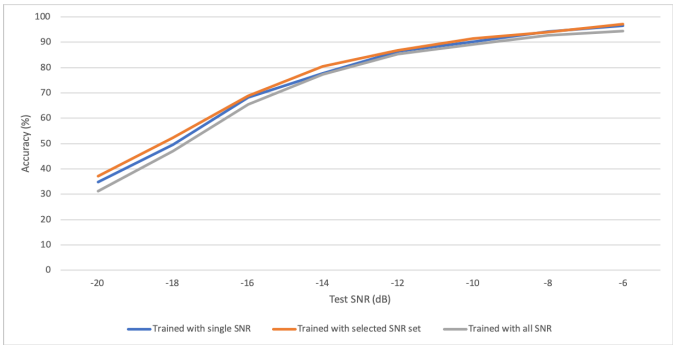

Fig. 6: SNR Boosting with CNN for Channel Identification

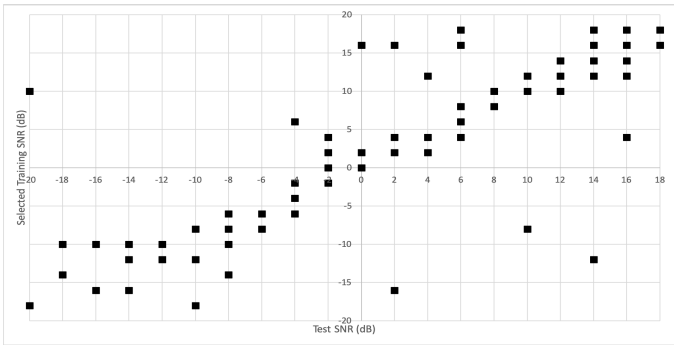

Fig. 7: SNR Set Selected by Boosting Algorithm with ResNet for Modulation Classification

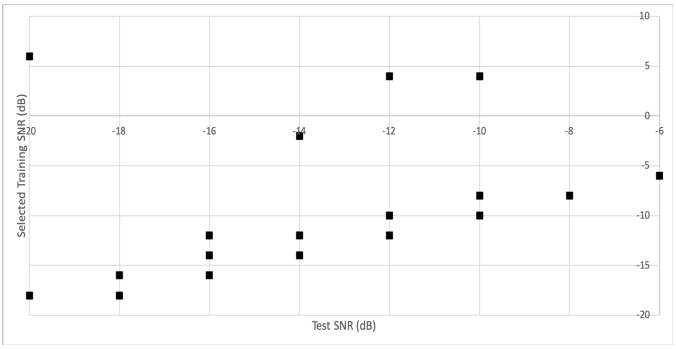

Fig. 8: SNR Set Selected by Boosting Algorithm with CNN for Channel Identification

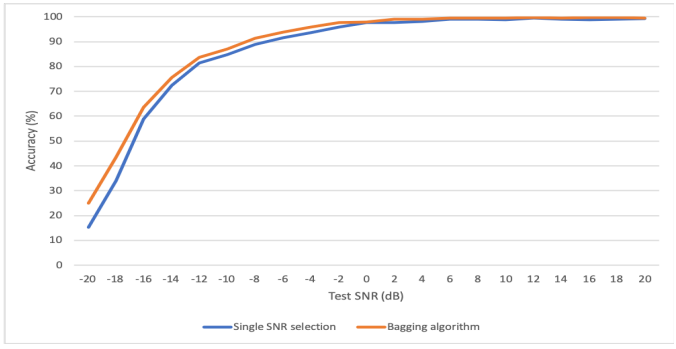

Fig. 9: Bagging Algorithm using 5% of the single SNR training set size with CNN for Channel Identification

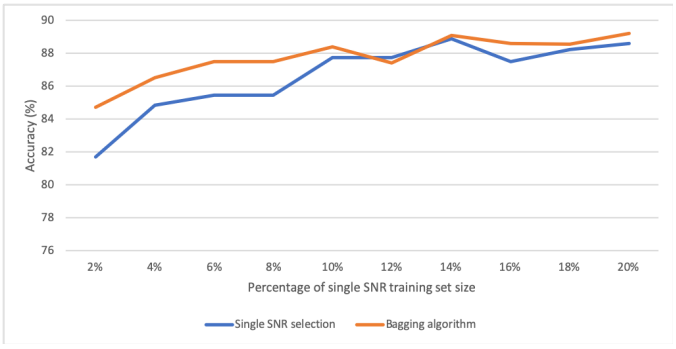

Fig. 10: Bagging at -10 dB for Channel Identification

adjacent values. During inference, we hold a vote among the three models. We compare the test accuracy to that obtained by single SNR selection using a training set of the same size. While no noticeable improvement was observed for modulation classification, we observe from Figures 9 and 10 that noticeable improvements in test accuracy are obtained for channel identification for smaller training sets and lower SNR values. Our intuition here for the explanation is that generalization becomes a dominant factor in determining the learning performance due to the scarcity of data that carries clear patterns.

REFERENCES

[1] T. O'Shea and N. West, "Radio machine learning dataset generation with GNU radio," in *Proc. GNU Radio Conference*, 2016.

[2] T. O'Shea, J. Corgan, and T. Clancy, "Convolutional radio modulation recognition networks," in *Proc. International Conference on Engineering Applications of Neural Networks*, 2016.

[3] X. Liu, D. Yang, and A. El Gamal, "Deep neural network architectures for modulation classification," in *Proc. Asilomar Conference on Signals, Systems, and Computers*, 2017.

[4] S. Ramjee, S. Ju, D. Yang, X. Liu, A. El Gamal, and Y. C. Eldar, "Fast deep learning for automatic modulation classification," *IEEE Machine Learning For Communications Emerging Technologies Initiative*, 2018.

[5] M. Schmidt, D. Block, and U. Meier, "Wireless interference identification with convolutional neural networks," in *Proc. IEEE 15th International Conference on Industrial Informatics (INDIN)*. IEEE, 2017, pp. 180–185. [Online]. Available: https://crawdad.org/owl/interference/20180925/

[6] X. Zhang, T. Seyfi, S. Ju, S. Ramjee, A. El Gamal, and Y. C. Eldar, "Deep learning for interference identification: Band, training SNR, and sample selection," in *Proc. IEEE International Workshop on Signal Processing Advances in Wireless Communications*, 2019.

[7] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning," *MIT Press*, 2016.