MAC ID Spoofing-Resistant Radio Fingerprinting

Tong Jian, Bruno Costa Rendon, Andrey Gritsenko, Jennifer Dy, Kaushik Chowdhury, and Stratis Ioannidis
Electrical and Computer Engineering, Northeastern University, Boston, MA
{jian,bcostare,a.gritsenko,jdy,krc,ioannidis}@ece.neu.edu

***Abstract*—We explore the resistance of deep learning methods for radio fingerprinting to MAC ID spoofing. We demonstrate that classifying transmission slices enables classification of a transmission with a fixed-length input deep classifier, enhances shift-invariance, and, most importantly, makes the classifier resistant to MAC ID spoofing. This is a consequence of the fact that the classifier does not learn to use the MAC ID to classifying among transmissions, but relies on other inherent discriminating signals, e.g., device imperfections. We demonstrate this via experiments on transmissions generated using two protocols, namely, WiFi and ADS-B.**

I. Introduction

Radio fingerprinting methods identify salient, discriminative characteristics of wireless devices from their transmissions, that can be used to identify a transmitting device. Traditional methods approach this problem by designing hand-crafted, protocol-specific discriminative features [1]–[3]. A series of recent works [**?**], [1], [4]–[11] approach the problem via machine learning methods: transmission datasets, labeled by the identities of their respective sources, are used to train classifiers that detect these transmissions source identities. These methods, and deep learning approaches in particular [**?**], [6], [9], [11], forego the need of hand-crafted/protocol-specific feature design, and can be readily trained directly over raw transmissions (i.e., received sequences of I/Q samples). Beyond the generalizability and protocol-independence of such methods, operating directly on raw samples has additional advantages, such as opening the possibility of real-time inference via dedicated hardware implementations of neural networks [12]. It also allows leveraging the broader arsenal of neural network architectures developed for other machine learning tasks that have been tremendously successful in domains such as image [13]–[15] and speech recognition [16], [17]. Indeed, deep learning methods have recently shown to be extremely well-suited for radio fingerprinting tasks, achieving high classification accuracy even when detecting hundreds of transmitting devices [6], [11].

Despite this success, a long-standing criticism of deep learning methods is their *lack of interpretability* [18]; in contrast to shallow learning methods, features extracted by deep models cannot be easily interpreted. This casts doubt on whether correct classification occurs because a deep learner indeed learns truly discriminative, generalizable input features, or because it is simply picking up artifacts present in the data. This issue is of direct relevance, and utmost significance, in the context of radio fingerprinting from raw I/Q samples. Indeed, in an ideal scenario, a learner should be identifying signal features caused by imperfections of the transmitting device hardware. These include, for example, I/Q imbalances, phase noise, carrier frequency and phase offsets, and harmonic and power amplifier distortions, to name a few [11]. Unfortunately, the nature of wireless communication implies that *almost all transmissions contain a strongly discriminative artifact*, namely, the identity of the transmitting device (e.g., the MAC address in the context of 802.11 transmissions), which is often included in the header of a transmitted packet.

Training a neural network directly on raw I/Q samples therefore runs the risk of, e.g., making it detect the transmitting device's MAC address. Clearly, this is catastrophic from a radio fingerprinting perspective, as it makes the classifier susceptible to MAC spoofing: a device can easily prevent detection by placing a different identifier in its header. Identifying the portion of the transmission that contains the MAC and removing it requires demodulation, which is in itself protocol-specific, and obviously also limits the application of the classifier to transmissions over known protocols. In effect, this negates the inherent generality advantages of deep learning methods outlined above. This motivates us to seek deep learning methods that are *resistant to MAC-learning* (or, more generally, resistant to identifier-learning). Nevertheless, the lack of interpretability of deep learning architectures implies that even the assessment of whether a neural network is indeed learning salient features (like hardware imperfections) or artifacts (like the MAC address) is a challenging task in it own right.

In this work, we demonstrate that a slicing technique, originally developed by Riyaz et al. [11] and Sankhe et al. [6], indeed produces MAC-learning resistant deep learning architectures. In short, the slicing technique splits variable-length transmissions into randomly selected slices of fixed length, and trains a deep neural network (DNN) classifier over the slices. The slices are generated by sliding a window across the variable-length transmission sequence. Classification of a new transmission is performed by again splitting the transmission over slices, classifying each slice individually, and aggregating the results (e.g., through a majority vote) to identify the source of the entire transmission. Riyaz et al. [11] note that this approach has the benefit of (a) enabling the application of a fixed-length input DNN over a variable-length sequence, while (b) producing the slices via sliding window also enhances the *shift-invariance* of the produced classifier.

We show that this sliding window technique, combined with randomization during training, has the additional important benefit of attaining MAC-learning resistance. In particular, we make the following contributions:

1) We extend the slicing scheme by Riyaz et al. [11] and Sankhe et al. [6] to incorporate randomization, by sampling only a subset of all possible slices per

transmission. As a side effect, this also significantly accelerates training.

2) We show through an extensive set of experiments that this slicing technique achieves MAC-learning resistance. Our experiments are designed to overcome the opaqueness of deep-learning methods, establishing that indeed the neural network does not learn the device's identifier. We establish this over transmissions generated by two protocols, namely, WiFi and ADS-B.
3) For ADS-B transmissions, we know the precise location in the device ID within the I/Q sequence. We show that, when training a DNN without slicing, prediction accuracy increases rapidly when exposing the classifier to the portion of the transmission that contains the device ID. This phenomenon, which indicates that the DNN is learning the ID, vanishes when using slicing.
4) We train the classifier over a dataset of WiFi transmissions by 100 devices, and test it over a test set of transmissions in which devices have shuffled their MAC addresses. When using slicing, our classifier is able to detect the devices with a 99.7%; this strongly suggests that the DNN is *not* relying on the MAC when classifying a transmission.

The rest of this paper is organized as follows. We first describe in detail our slicing scheme, followed by a description of our DNN architecture and its adaptation to the radio fingerprinting domain (Sec. II). We then describe the datasets we use and present our experimental results (Sec. III). Finally, we conclude with a discussion on open research issues (Sec. IV).

II. Methodology

Our overall approach is as follows: we are given a dataset of wireless transmissions, each consisting of a sequence of I/Q samples, i.e., complex numbers capturing phase and amplitude information. We treat each such sample as a two-dimensional vector; as such, transmissions are represented as finite, variable length sequences in $\mathbb{R}^2$. We separate this dataset into a training set and a test set. We train a deep neural network classifier which we call RFMLSNet on the training set, with labels being the identities of the transmitting devices. We subsequently test the accuracy of the classifier on the test set, predicting the labels (i.e., source devices) of the test transmissions. During this process, we further separate 20% of the training set into a validation set used to determine network hyperparameters.

Following Riyaz et al. [11] and Sankhe et al. [6], we *slice* transmissions, generating fixed length sequences that are subsequently fed to the neural network. In the remainder of this section, we describe this slicing procedure in detail, along with how the trained neural network is used to classify transmissions in the test set. We conclude with an overview of the DNN architecture we used in our experiments.

A. Random Slicing

Recall that both WiFi and ADS-B transmissions are sequences of varying length of I/Q samples. To be able to process this data with neural networks that take as input only data of fixed length, Riyaz et al. [11] propose a sliding window approach to cut transmissions into a number of slices. More specifically, they first define a desired slice length, L. Then, given a transmission k of length M_k, they generate $M_k - L + 1$ slices, as illustrated on Figure 1.

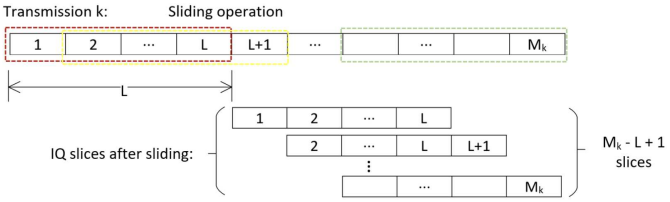

Fig. 1. Generating $M_k - L + 1$ slices from a transmission of length M_k.

Not all generated slices have to be used to effectively train a classifier. To that end, we use only a small portion of slices, chosen randomly among the total number of slices ($M_k - L + 1$). This has several advantages: (a) it is a natural way to satisfy the requirement of fixed-size input for neural networks, (b) it improves classifier's ability to learn shift-invariant features, and (c) it inherently reduces computations during training. More specifically, for each transmission k of M_k length, we randomly sample $n_k = \frac{M_k - L + 1}{L}\kappa$ slices, where L is the slice length, and $\kappa \in [1, L]$ is a hyperparameter. Intuitively, hyperparameter κ captures the number of slices that each I/Q sample of transmission k participates, in expectation; equivalently, this is also the number of times an I/Q is seen by a network during a training epoch.

B. Classifying an Unlabeled Transmission

Given a classifier trained to predict device of an input sequence of I/Q samples and an unlabeled transmission k from the test set, we perform the following steps to identify the device that transmitted k. First, just as in the training set, we slice transmission k and randomly pick n_k slices, according to slicing procedure described in Section II-A. For each slice i, we use the pretrained classifier to classify each slice. As described below (Section II-C), the classifier produces a set of probabilities p_{ij}, indicating whether slice i belongs to device j, where $\sum j p_{ij}$ for every slice $i \in \{1, \ldots, n_k\}$. Therefore, we infer the predicted device label of the transmission via:

$$\hat{j} = \arg\max_j \sum_{i=1}^{n_k} p_{ij}. \quad (1)$$

Intuitively, treating p_{ij} as the probability slice i came from device j, this classification rule identifies the transmission source as the device $\hat{j}$ that labeled the most slices, in expectation.

C. Architecture Overview

Our RFMLSNet architecture is illustrated on Figure 2. The model takes a slice (i.e., sequence of I/Q samples of fixed length) as an input and outputs a vector of probabilities. The length of the output vector equals to the number of devices in the dataset, and each vector's element represents the probability of a given sequence to be transmitted by a corresponding device. Besides the input and output layers, the model comprises 10 convolutional layers (CL), 5 max-pooling

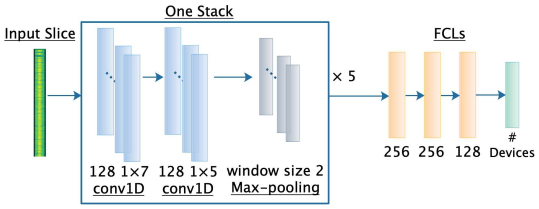

Fig. 2. RFMLSNet architecture. RFMLSNet is a modified version of AlexNet, adapted for use in the radio fingerprinting domain. To adapt to slices which are timeseries with two features (I and Q) per time step, we use 1D CLs. For each stack, the first CL convolves 128 different 1D filters of size 1×7 across the full length of an input sequence of I/Q samples, and the second CL uses 128 1D filters of size 1×5. Each CL produces a vector of 128 feature maps, identifying when a specific type of temporal feature is detected in the input. Non-linearities are introduced via ReLU activation functions after each CL. MPLs down-sample the input. Three FCLs of different size (256 and 128 neurons, respectively) with a ReLU activation function learn high-level non-linear features. Finally, the output layer is also a FCL with a softmax activation function.

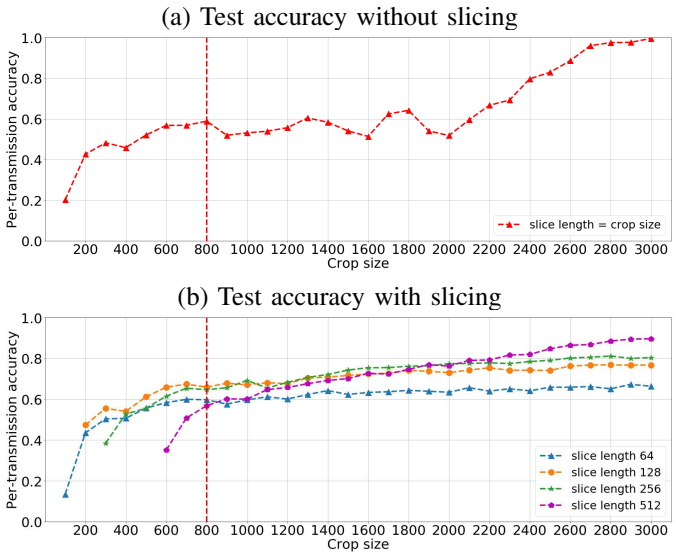

Fig. 3. Test accuracy over ADS-B dataset with and without slicing. The dashed line at 800 samples indicates the position where the packet payload (which includes the identifier) begins. (a) When measuring test accuracy without slicing, we see that the test accuracy hovers around 50% initially; at around 2000 samples, the accuracy increases rapidly, reaching 99.56% accuracy by crop size 3000. This sharp increase strongly indicates that this is due to learning the device ID. (b) Test accuracy with slicing does not exhibit a sharp increase once surpassing the 800 sample mark; accuracy improvement is gradual, which is consistent with the ability of the network to classify better due to being able to observe more data, rather than because it is actually learning the device ID.

layers (MPL) and 3 fully-connected layers (FCL). CLs and MPLs are combined into 5 consecutive stacks, each containing two CLs with rectified linear unit (ReLU) activations followed by a single MPL. Three FCLs, also with ReLU activations, and an FCL output layer with a soft-max activation, learn high-level features; we use a categorical cross-entropy objective during training. We use 20% of the training set as a validation set to determine hyperparameters, such as the number of consecutive stacks/overall depth, the number of final FCLs and the size of filters in CLs, the learning rate, and the stopping time: we terminate learning when the validation accuracy ceases to improve.

III. MAC-Spoofing Resistant Learning

A. Datasets

In our experiments, we use three datasets with different properties designed to confirm the MAC-learning resistance of our approach. We provide a brief description of each dataset below; Table III summarizes each dataset's statistics.

ADS-B Dataset. The first dataset consists of wireless transmissions communicated by 50 devices through the ADS-B protocol. All transmissions are recorded at a center frequency of 1.09 GHz with the sampling rate 100 MS/s. Each device transmits 196 signals with the average length of 9519 I/Q samples. For each device, 141 randomly selected transmissions form the training set and the remaining 55 the test set.

Scrambled MAC WiFi Dataset. The second dataset involves synthetic WiFi signals with a fixed length of 45183 I/Q samples. The training set consists of 100000 signals from 100 unique devices (1000 signals/device), and the test set consists of 100000 signals from those same 100 unique devices. However, the MAC IDs are randomly permuted among the signals in the test set. This dataset aims to confirm whether the network is learning to recognize the MAC ID of a particular device as its discriminating feature.

Bitwise Identical WiFi Dataset. Finally, the bitwise identical WiFi dataset comprises of 11191 transmissions communicated by 19 identical devices (same manufacturer and MAC) in the same channel conditions. The WiFi standard, Commercial off the shelf (COTS) 802.11a, were used for data collection. Transmissions were recorded at a center frequency of 5 GHz with the sampling rate varying between 20-200 MS/s.

B. Results

ADS-B Experiments. In the case of the ADS-B transmissions, we know precisely the portion of the transmission that contains the device ID. Indeed, the ADS-B protocol signal format consists of (a) a sync pulse, which lasts 8 μs (800 samples) followed by, (b) either 56 or 112 μs of data payload, depending on the type of transmission. The device ID appears in the data payload of the packet.

To study whether the neural network learns the device ID, we crop transmissions: each transmission in both the training set and the test set is truncated after a certain number of I/Q samples. We investigated at crop sizes (i.e., truncation thresholds) ranging from 64 to 512. In short, for both training and testing, we only look at a prefix of each packet, completely discarding the remainder. Note that, as a consequence, all (previously variable-length) transmissions have the same length (namely, equal to the crop size).

Subsequently, we train a DNN on top of these cropped transmissions in two ways. The first is without slicing: in this case, the entire cropped transmission is fed to the neural network, both in training and in testing. The second is with the slicing procedure introduced in Section II-C: for test transmissions, the sum of weights rule is used to produce the classification outcome for the entire cropped transmission.

Figure 3(a) shows the test accuracy of the network without slicing, as a function of the crop size. The behavior of the

TABLE I
DATASET DESCRIPTION

Dataset	# Devices	# Train transmission/device	# Test transmission/device	Average transmission length
ADS-B	50	141	55	9519
Scrambled MAC WiFi	100	1000	1000	45183
Bitwise Identical WiFi	19	8953	2238	20088

For each dataset, we provide the total number of devices, number of transmissions per device in training and test sets, respectively, and the average length of transmitted signals.

TABLE II
DATASET DESCRIPTION

Dataset	# Devices	# Train transmission/device	# Test transmission/device	Average transmission length
Scrambled MAC WiFi	100	1000	1000	45183

For each dataset, we provide the total number of devices, number of transmissions per device in training and test sets, respectively, and the average length of transmitted signals.

TABLE III
DATASET DESCRIPTION

Dataset	# Devices	# Train transmission/device	# Test transmission/device	Average transmission length
Bitwise Identical WiFi	19	8953	2238	20088

For each dataset, we provide the total number of devices, number of transmissions per device in training and test sets, respectively, and the average length of transmitted signals.

TABLE IV
TEST ACCURACY WITH SLICING FOR FULL/UNCROPPED TRANSMISSIONS

Dataset	Slice length	Accuracy Per-slice / Per-transmission
ADS-B	1024	0.810/0.919
Scrambled MAC WiFi	1024	0.972/0.997
Bitwise Identical WiFi	128	0.778/1.000

The prediction accuracy of the classifier with slicing over the Scrambled Mac WiFi and Bidwise Identical WiFi datasets indicate that the neural network, combined with slicing, is MAC-learning resistant. In both experiments, if the network was indeed learning the MAC address, the per transmission accuracy on the test set would be no better than random (1% and $\approx$5%, respectively). In contrast, the per transmission accuracy is very high on both datasets (99.7% and 100%, respectively).

TABLE V
TEST ACCURACY WITH SLICING FOR FULL/UNCROPPED TRANSMISSIONS

Dataset	Slice length	Accuracy Per-slice / Per-transmission
Bitwise Identical WiFi	128	0.778/1.000

The prediction accuracy of the classifier with slicing over the Scrambled Mac WiFi and Bidwise Identical WiFi datasets indicate that the neural network, combined with slicing, is MAC-learning resistant. In both experiments, if the network was indeed learning the MAC address, the per transmission accuracy on the test set would be no better than random (1% and $\approx$5%, respectively). In contrast, the per transmission accuracy is very high on both datasets (99.7% and 100%, respectively).

accuracy strongly indicates that the network is indeed learning the device ID. The dashed line at 800 samples indicates the position where the packet payload (which includes the identifier) begins. When only the preamble is used (crop size $<$800 I/Q samples), the accuracy relatively quickly reaches 50%, and remains relatively stable. At around 2000 samples, we observe a sharp increase in accuracy, reaching a 99.56% accuracy by crop size 3000. This regime is precisely the regime where a big fraction of the data (i.e., cropped transmissions) seen by the network is in effect the device ID. When this becomes available to the neural network, the network quickly learns to ignore the remaining portion of the transmission, enabling it to reach an almost 100% accuracy on the test set.

We observe a very different form of behavior under slicing, as indicated in Figure 3(b). In almost all cases, we observe an increase in accuracy as the crop size grows, but at a very slow rate: this is consistent with a behavior in which "more data helps": the network becomes better at detecting the device when it can see a bigger part of the transmission. There is a sharp improvement in the beginning, i.e., when the network can see more than one slice per transmission.

TABLE VI
TEST ACCURACY WITH SLICING FOR FULL/UNCROPPED TRANSMISSIONS

Dataset	Slice length	Accuracy Per-slice / Per-transmission
Scrambled MAC WiFi	1024	0.972/0.997

The prediction accuracy of the classifier with slicing over the Scrambled Mac WiFi and Bidwise Identical WiFi datasets indicate that the neural network, combined with slicing, is MAC-learning resistant. In both experiments, if the network was indeed learning the MAC address, the per transmission accuracy on the test set would be no better than random (1% and $\approx$5%, respectively). In contrast, the per transmission accuracy is very high on both datasets (99.7% and 100%, respectively).

However, all subsequent improvements in accuracy happen at a slow rate: no sharp increase akin to the one observed at Figure 3(a) occurs when the network is exposed to the entire ID. This suggest that the network is not learning the ID, but is learning to focus on other discriminative features present in the transmission.

The test accuracy with slicing over the entire transmission (i.e., without cropping) is shown on the first row of Table VI. Both per slice and per-transmission accuracy are reported. Although the accuracy attained is quite high, at 91.9%, it is below the 99.56% accuracy achieved by the network without slicing, that learns the device ID.

Scrambled MAC-Wifi Dataset. The strongest evidence of the MAC-learning resistance of training the network with slicing is provided by its performance on the MAC-shuffling dataset. Recall that, on this dataset, the MAC addresses of devices in the test set are shuffled: each device broadcasts a transmission with a different MAC ID than the one it used in the training set.

The test accuracy of the neural network with slicing is provided on the second line of Table VI. The observed performance strongly indicates that *the neural network, combined with slicing, is indeed not learning the MAC address* embedded within the training set transmissions! If it were so, the accuracy on the (MAC-shuffled) test set would be no better than random (1%). In contrast, both per slice and per transmission accuracy are very high (97.2% and 99.7%, respectively).

Bitwise Identical Devices. We reach the same conclusion when observing the network performance with slicing over the dataset of bitwise identical devices. Recall that, in this case, all 19 devices transmit the same sequences, broadcasting the same MAC address. The testing accuracy (both per slice and per transmission) is provided on the third line of Table VI. If the network was using the MAC address as a discriminative feature, prediction accuracy over the test set would be no better than random ($\approx 5\%$). Nevertheless, the neural network is again able to classify unseen transmissions in the test set with 100% accuracy (no device is misclassified as another).

IV. Future Research Directions

In this paper, we demonstrate that classifying transmission slices enhances shift-invariance, and makes the classifier resistant to learning MAC IDs as features. Channel variations could be another inherent feature among transmissions which may hamper classifier performance; building classifiers invariant to channel conditions is an important open problem. Another interesting future direction could be using adversarial learning [**?**], [**?**] to increase resistance against MAC ID spoofing, channel variations, or any other feature we should be ignoring during inference.

Acknowledgment

This work is supported by DARPA under RFMLS program contract N00164-18-R-WQ80. We are grateful to Paul Tilghman and Esko Jaska for their insightful comments and suggestions.

References

[1] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless device identification with radiometric signatures," in *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking*, 2008, pp. 116–127.

[2] S. U. Rehman, K. W. Sowerby, S. Alam, I. T. Ardekani, and D. Komosny, "Effect of channel impairments on radiometric fingerprinting," in *2015 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, Dec 2015, pp. 415–420.

[3] T. D. Vo-Huu, T. D. Vo-Huu, and G. Noubir, "Fingerprinting Wi-Fi devices using software defined radios," in *Proceedings of the 9th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2016, pp. 3–14.

[4] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, J. Van Randwyk, and D. Sicker, "Passive data link layer 802.11 wireless device driver fingerprinting," in *ACM USENIX Security Symposium - Volume 15*, 2006.

[5] K. Gao, C. Corbett, and R. Beyah, "A passive approach to wireless device fingerprinting," in *IEEE DSN 2010*, June 2010, pp. 383–392.

[6] K. Sankhe, M. Belgiovine, F. Zhou, S. Riyaz, S. Ioannidis, and K. Chowdhury, "ORACLE: Optimized Radio clAssification through Convolutional neuraL nEtworks," in *IEEE International Conference on Computer Communications*, 2019.

[7] I. O. Kennedy, P. Scanlon, F. J. Mullany, M. M. Buddhikot, K. E. Nolan, and T. W. Rondeau, "Radio transmitter fingerprinting: A steady state frequency domain approach," in *IEEE VTC*, Sept 2008, pp. 1–5.

[8] S. V. Radhakrishnan, A. S. Uluagac, and R. Beyah, "Gtid: A technique for physical device and device type fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, Sept 2015.

[9] T. J. O'Shea and J. Corgan, "Convolutional radio modulation recognition networks," 2016. [Online]. Available: http://arxiv.org/abs/1602.04105

[10] N. T. Nguyen, G. Zheng, Z. Han, and R. Zheng, "Device fingerprinting to enhance wireless security using nonparametric bayesian method," in *IEEE INFOCOM*, April 2011, pp. 1404–1412.

[11] S. Riyaz, K. Sankhe, S. Ioannidis, and K. Chowdhury, "Deep learning convolutional neural networks for radio identification," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 146–152, 2018.

[12] A. R. Omondi and J. C. Rajapakse, *FPGA implementations of neural networks*. Springer, 2006, vol. 365.

[13] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015, pp. 1–9.

[14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[15] Y. He, Z. Zhang, F. R. Yu, N. Zhao, H. Yin, V. C. Leung, and Y. Zhang, "Deep-Reinforcement-Learning-based Optimization for Cache-enabled Opportunistic Interference Alignment Wireless Networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 11, pp. 10 433–10 445, 2017.

[16] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[17] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 6645–6649.

[18] G. Montavon, W. Samek, and K.-R. Müller, "Methods for interpreting and understanding deep neural networks," *Digital Signal Processing*, vol. 73, pp. 1–15, 2018.