Lean Neural Networks for Real-time Embedded Spectral Notching Waveform Design

Anthony Baietto [1,*], Jayson Boubin [1], Patrick Farr [2],
Trevor J. Bihl [3]
Department of Computer Science and Engineering, Ohio State University[1]
Applied Research Solutions[2]
U.S Air Force Research Laboratory[3]

Abstract*—Spectral notching, a waveform design method used in signal processing and radar, mitigates interference caused by an ever-growing number of technologies which saturate the radio frequency (RF) spectrum. Online waveform design is possible with current technology, but extant techniques can not meet real-time latency requirements on the low size weight and power (SWaP) embedded hardware ideal for contexts where online waveform design is most useful. Current techniques rely on convex optimization, leading to non-trivial execution times and excessive compute and power requirements. Artificial Neural Networks (ANNs) could feasibly be used for spectral notching due to their ability to approximate complex non-linear functions; however, ANNs generally require considerable processing power and result in long inference times. Prior work has shown that ANNs can accurately perform real-time radar waveform design, albeit in very specific contexts. In this paper, we extend that work, demonstrating that lean neural networks, which model successful convex optimization algorithms in a lower dimensional representation, can meet real-time latency constraints for general waveform design without sacrificing accuracy on low SWaP embedded hardware. Our results are exemplified in simulation on real embedded and general purpose hardware. Overall, our lean neural network solution decreases inference time on traditional embedded hardware by $8.5x$ when compared to the fastest optimization approach, and uses $10.6x$ less energy. Our system executes quickly on low-power embedded devices, using only 16.6% of the energy a conventional GPU-provisioned system would require.

I. Introduction

Wireless communication has grown as an important communication medium in past decades, recently due to the rise of both mobile computing and the Internet of Things [1]. As we realize the potential of 5G, with widespread, high bandwidth wireless networks, an increase in mobile and IoT devices is leading to an increase in wireless spectrum interference. As the radio frequency (RF) spectrum saturates with interference caused by (potentially nefarious) RF devices, the need for interference mitigation becomes increasingly apparent.

Interference mitigation is the process of eschewing saturated communication channels in favor of clear channels. Communication across saturated or jammed channels can lead to partial or total communication loss. One means of avoiding saturated channels is waveform notching, where waveforms are deliberately changed to transmit in a determined non-jammed pass-band instead of the saturated or jammed stop-band. Notched waveform design has extant optimization solutions, notably the Error Reduction Algorithm (ERA) [2] and the Re-Iterative Uniform Weight Optimization Algorithm (RUWO) [3]. These algorithms, however, come with significant penalties. RUWO is a highly accurate technique that results in high null depths, the difference in power between stop and pass bands of the transmitted waveform, but requires considerably latency. ERA has both lower latency and higher null-depths than RUWO, but does not meet the latency constraints required for online interference avoidance on embedded hardware.

This latency problem is exacerbated by the fact that many technologies which would benefit from real-time waveform design, such as small UAV [4], [5] and connected autonomous vehicles [6] have strict size, weight, and power (SWaP) requirements. For low latency applications, such as embedded signal processing, waveform design algorithms [2], [3] rely on online optimization for latency-sensitive problems like interference avoidance. As adoption of IoT devices increases, and interference mitigation technology becomes increasingly necessary for latency sensitive and low power devices, current optimization techniques will not meet requirements; therefore, new techniques must emerge.

A promising direction for new techniques can be found in prior interference mitigation work using Artificial Neural Networks [7]–[9]. Prior researchers have shown that Artificial Neural Networks could be an alternative technique for solving the interference mitigation problem. Artificial Neural Networks (ANNs) have gained popularity as non-linear function approximators. They use from 10s to millions of creatively arranged neurons to fit non-linear functions which classify images [10], detect speech [11], control robots [4], and more. Neural network development has been simplified by the adoption of programming models [12], [13] and datasets [14] that standardize their design and training, but still require considerable expertise to efficiently implement. Even with these tools and the rise of highly parllel hardware, neural networks do not often meet real-time goals, especially on low SWaP hardware.

A viable approach towards highly accurate yet low-latency waveform design is possible through the use of ANNs [7], [8], but prior work uses a naive neural network to converge to an optimal solution over time. In our work, we design a lean neural network for real-time embedded waveform design which is more general than prior approaches and incorporates problem-specific information derived from extant algorithms into the structure of the ANNs. we use AutoWave [9], a lean neural network architecture for notched waveform design, to implement a low-power embedded waveform design system. AutoWave relies on creative architectural decisions in the neural network design process to minimize network size and depth, which decreases latency and maximizes embedded performance. We demonstrate that AutoWave decreases inference time on embedded devices by $8.5x$ when compared to the fastest convex optimization approach (ERA) while using $10.6x$ less energy. We also show that AutoWave maintains a high level of precision within the waveform design problem, reaching a peak accuracy within 1% of the slow converging but highly accurate RUWO algorithm, and maintaining a null depth comparable to ERA at far lower latency.

II. Background

Sources of RF interference are rapidly increasing, as depicted in Fig. 1, due to the growing abundance of Internet of Things (IoT) devices and consumer technologies [15]. It is therefore imperative to find ways to mitigate this interference in order for radar applications to function at maximum efficiency. We choose to focus on interference avoidance via spectral notching, a method where waveforms are modified not to transmit in the stop-band (frequencies which are saturated with interference) and instead to transmit in the pass-band where interference is less present.

A. Spectral Notching Techniques

Spectral notching is primarily performed using two existing techniques, ERA and RUWO. ERA is a quickly converging optimization algorithm that is generally used for phase reconstruction, but can be modified to produce spectrally notched waveforms [16]. ERA uses an iterative optimization process equivalent to steepest descent gradient search [2]. The most accurate extant technique used to perform spectral notching is the Re-Iterative Uniform Weight Optimization Algorithm (RUWO) [3]. RUWO is

Fig. 1. Example interference environment. IoT devices and consumer technologies, such as smartphones, transportation vehicles, drones, etc., produce RF interference on an unprecedented scale.

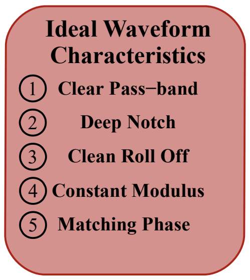
a deterministic process that performs frequency nulling using a covariance matrix and steering vector [3]. While both process are deterministic and converge quickly in an optimization context, both RUWO and ERA take a considerable amount of time to converge compared to the real-time constraints of online waveform design, even when using general purpose hardware [7].

For illustration, the latency garnered from both algorithms can be conveniently compared to common neural network operations. ERA relies on an iterative steepest-descent gradient optimization approach analogous to neural network training. RUWO, on the other hand, is conventionally performed for 50 iterations, each of which requires a series of matrix operations analogous to a forward pass through a layer of a neural network. Given the nature of these optimization mechanisms, it stands to reason that a single pass through a lean neural network could considerably outperform both of these operations in terms of latency and energy consumption.

B. IoT Devices, Interference, and Latency

Expanded use of IoT devices [17] to commercial, government, and civil use in all aspects of life has driven their widespread adoption. This proliferation of WPAN devices, such as Z-Wave and Raspberry Pi [18], [19], connected autonomous vehicles [6], UAV [4], and other internet connected devices has increased environmental RF interference. Often, such devices have little remaining power or compute capacity needed to mitigate interference [20] if they so chose to. It is imperative, therefore, that a waveform design solution either uses the embedded hardware already provisioned on these systems, or that additional hardware is low-SWaP.

Even if a technique can design waveforms on an embedded device, it must meet some latency constraint to be useful. A bare minimum benchmark for success



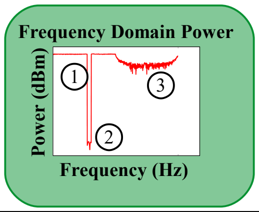
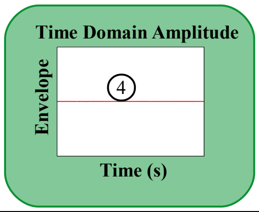
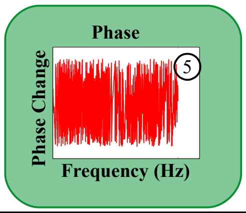
Waveform Characteristic			
Reflection in Loss Function	$\frac{1}{n}\sum_{i=1}^{n}(20*\log_{10}(FFT(\lvert z_i\rvert)) - 20*\log_{10}(FFT(\lvert\hat{z}_i\rvert)))^2$	$\frac{1}{n}\sum_{i=1}^{n}(IFFT(\lvert z_i\rvert) - IFFT(\lvert\hat{z}_i\rvert))^2$	$\frac{1}{n}\sum_{i=1}^{n}(\angle FFT(\lvert z_i\rvert) - \angle FFT(\lvert\hat{z}_i\rvert))^2$

Fig. 2. Desired waveform characteristics: (1) Clean pass-band (2) Deep notch in stop-band (3) clean roll off (4) Constant time-domain modulus(5) Matching phase (6) correctly timed notch

or a technique's latency could be that the waveform it is designing is created in less time than it will take to transmit, meaning the design of a one second transmission should take less than one second create with a successful technique on an embedded device at a minimum. Using this formulation, we demonstrate a neural technique that can successfully design 1024 Hz LFM chirp signals on embedded hardware.

III. Design

Given the inherent constraints of IoT devices, and the significant compute power required for modern waveform design techniques, we sought to 1) design an ANN technique that approximates waveform techniques which use optimization, and 2) assure that it operates in real time as per our formulation in section 2 on the embedded platforms that these devices use. In this section, we detail the design of a neural network that approximates the quality of RUWO and ERA, and the miniaturization process that allows it to operate on embedded devices.

A. *Key Challenges*

To design our efficient neural network for online waveform design, we took inspiration from existing work solving this problem with both neural networks [7] and optimization [3]. An existing neural solution to this problem, NeuroWav, solved this problem by matching input waveforms to output waveforms stored in a lookup table. NeuroWav used a binary mask to simplify the classification process, resulting in milisecond-level classifications on general purpose computers. This approach has considerably shortcomings. First, NeuroWav doesn't design waveforms itself. Waveforms are precomputed and stored in a lookup table, limiting the generalization and usefulness of the overall system, and constructing barriers to adoption. Second, NeuroWav performs a series of signal processing steps before classification to convert raw complex input waveforms into the time domain. This process is time consuming and potentially unnecessary, providing room for optimization.

Conversely, classical solutions like RUWO, as mentioned in section 2, far exceed real-time guarantees. Researchers found that RUWO used for spectral notching violates real-time constraints by multiple orders of magnitude, takings 100s of milliseconds to 10s of seconds to converge [7] on general purpose computers. The goal for our design is, therefore, to use RUWO data to train a creatively designed neural network which takes as input raw complex signal data and outputs a complex notched signal in real-time.

B. *Neural Network Design*

Our neural network relies on two creative design decisions to approximate RUWO with high accuracy. First, we designed a custom loss function which prioritizes desirable waveform features. Second, we separated the imaginary and real components of the complex input into two separate vectors which are fed to two different neural networks which perform inference in parallel.

1) Custom Loss Function: Neural networks depend on loss functions to evaluate the quality of their predictions [21]. The "Loss" generated from a specific prediction describes the error between that prediction and its actual value. Our neural network intakes sampled waveforms represented as vectors of complex number. Applying a commonly used loss function to this problem, such as mean-squared error, will eventually result in outputs that approximate the target RUWO waveforms, but will evenly evaluate error of all individual complex samples from the output waveform. While our output waveforms should closely approximate the RUWO output data, they should also maintain certain characteristics that training with a mean-squared error loss function does not inherently reinforce.

For this reason, we created a custom loss functions which rewards valuable waveform characteristics in the time and frequency domains that are not easily determined by the complex representations of our output waveforms over absolute accuracy between the input and output complex representations. As shown in Fig. 2,

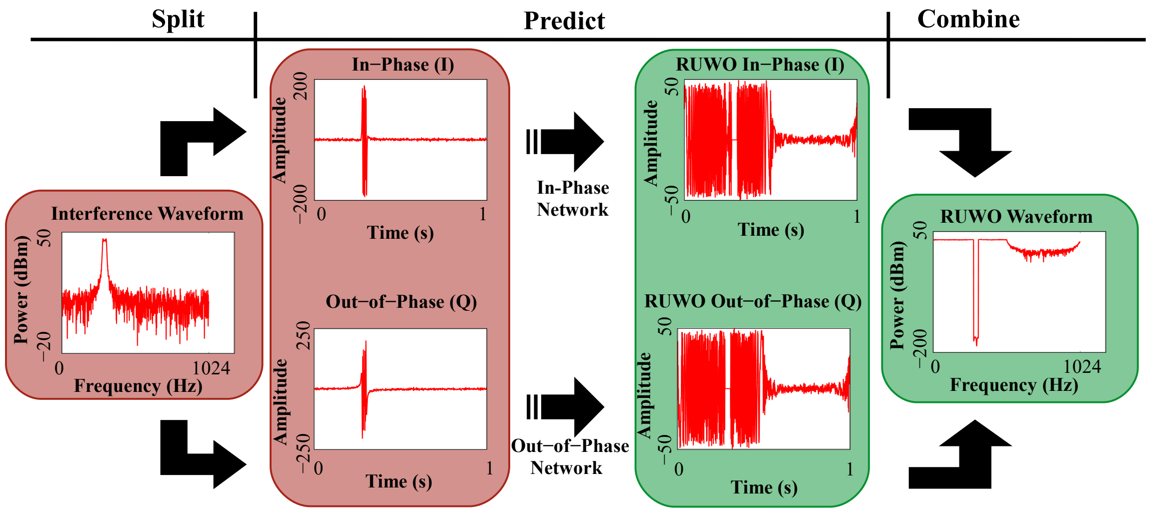


Fig. 3. Our split neural network architecture transforms input interference waveforms directly into transmittable notched waveforms, similar to the traditional neural network architecture, but now does so in 3 parallel stages: (1) Split complex signal into the two corresponding real-valued signals I and Q (2) Feed real signals through two separate neural networks (3) Combine new real-valued signals into complex signal.

we seek to prioritize 5 key waveform characteristics: A clear pass-band, a deep notch in the stop-band, clean roll off, a constant modulus in the time domain, and matching phase. Our loss function calculates error based on 4 equations: Mean-squared error between the input and output waveforms, and the three equations shown in Fig. 2 which make these time and frequency domain characteristics more apparent to loss evaluation. The final error returned by our loss function is the average loss reported by these four functions.

2) Separate I and Q Models: Our second design improvement focuses on the structure of the radar interference mitigation problem. The waveforms involved in this problem are composed of quadrature signals: 2 waveforms that are offset by 90° and summed together for transmission. These quadrature waveforms are represented as a single complex-valued signal where the real and imaginary components stem from the 2 signals. However, because neural networks operate solely on real values, these complex signals are first converted into coefficient vectors via a zipper merge. These coefficient vectors do not capture the quadrature relation between the 2 waveforms and so information is lost.

Our novel idea is to process the 2 signals separately to preserve each signals integrity. Using 2 separate neural networks, we give each network only 1 of the 2 signals and then combine the outputs of each network to form the solution(see Fig. 3). This approach avoids the complex data conversion problem and also ensures that the data and relations within each signal are protected. Additionally, because the complexity of the input data is reduced, the size of each neural network can also be reduced thus lowering latency.

C. Embedded Design

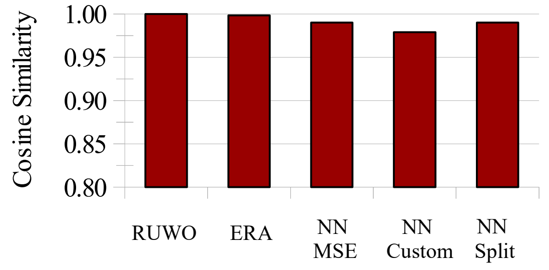
Once our neural network solution operated correctly on traditional hardware, we began the process of embedding our design into low-power devices. Executing our neural network solution on the Raspberry Pi platform required no architectural modifications given the universal compatibility of TensorFlow machine learning libraries. However, due to the relatively low computational power of the platform, we had to ensure that our solution only used optimized operations which eliminated the usage of any pre or post-processing to avoid additional latency.

IV. Implementation

In this section, we provide details for the data generation, algorithm setup, and neural network implementation toward replication and advancement of our work. We include the specific hardware used for simulation as well as any software libraries, and their respective parameters, used for machine learning.

To train our neural network, we generated training waveforms using both ERA and RUWO using MATLAB 2021a. We generated 262,144 1 second sample LFM waveforms with a 1024 Hz sample rate and 512 Hz transmit bandwidth with a Gaussian noise amplitude of 0.1 and variance of 1. Sample waveforms were created using the Air Force Research Laboratory's Mustang supercomputer.

The AutoWave neural network was build using Keras with the Tensorflow backend in Python 3.6. Our network was, like our data generation, trained on the Mustang super computer using an NVIDIA Tesla P100 GPU. Our best-performing neural network, after hyperparameter tuning, had a single hidden layer with a width of 256



Null Depth (dBm)	202.2	31.9	28.5	22.3	29.8

Fig. 4. Null Depth and Cosime Similarity between RUWO, ERA, and our three neural network implementations: 1) our NN architecture with a mean-squared error loss function, 2) our NN with our custom loss function, and 3) our NN architecture with both a custom loss function and split IQ model.

neurons and a tanh activation layer. Our networks were trained for 100 epochs with 10-fold cross validation.

For our experiments, we used the low-cost Raspberry Pi 3 Model B which features a quad core 1.2GHz Broadcom BCM2837 ARMv8 64-bit CPU and 1GB of LPDDR2 RAM but has a small credit-card sized profile. This device is used in many embedded systems platforms for its small profile and energy footprint, programmability, and low cost. For comparison, we executed all techniques on a general purpose computer. We used a Dell r720 with dual intel E5-2670 CPUs, 144GB of RAM, and an NVIDIA GT 1030 GPU.

V. Results

We now show that our neural network solution outperforms prior convex optimization attempts at the spectral notching waveform problem on embedded devices. We will show that neural networks deployed to low-power devices can achieve comparable accuracy to state-of-the-art algorithms with drastic latency and power reductions.

Fig. 4 shows the performance of RUWO, ERA, and three neural network models in terms of cosine similarity to RUWO and overall null depth. ERA is 99.9% similar to RUWO, while our ANNs are between 99.0% and and 97.8% simlar to RUWO. The naive implementation, NN MSE, and our split IQ and custom loss model achieve the same accuracy (99%) across our sample set. Accuracy to RUWO, however, is not entirely indicative of performance. A high null depth in our resultant signals is imperative to assure that signals operate well in the high signal to interference and noise ratio (SINR) environments where interference avoidance is necessary. RUWO achives an astonishing null depth of 200 dBm, which far exceeds necessary depth for most radar applications experienced by IoT devices. ERA achieves a null depth of 31.9 dBm, and our neural network techniques achieve null depths respectively of 28.5, 22.3, and 29.8. While

TABLE I
Latency of RUWO, ERA, and our neural networks when executed on an embedded device as well as general purpose hardware.

Algorithm	**Raspberry Pi Latency (ms)**	**CPU / GPU Latency (ms)**
RUWO	453965.43 ± 4131.61	1064.98 ± 10.94
ERA	1982.04 ± 29.27	185.47 ± 3.87
NN MSE	230.98 ± 2.74	23.19 ± 1.86
NN Custom Loss Function	233.92 ± 3.16	20.72 ± 0.44
NN Split	250.90 ± 0.63	23.35 ± 0.29

our split network model does not outperform ERA, it's absolute null depth is 35% greater than our naive MSE network, demonstrating that clever network design can maintain accuracy and improve performance.

While maintaining overall performance is paramount, it is important to determine the latency and power consumption of each technique on embedded devices to truly assess their efficacy for low SWaP interference avoidance. Tab. I shows the average latency of a single execution of RUWO, ERA, and our neural networks run on both a Raspberry Pi 3b embedded device and a GPU-provisioned general purpose computer. RUWO has a total latency of 453.965 seconds on embedded hardware and 1.064 seconds on conventional hardware, which are both untenable for our one second waveform. ERA's latency of 185ms on conventional hardware should be tenable to maintain continuous communication for 1 second waveforms, but it's embedded performance of 1.982 seconds clearly precludes continuous transmission.

Our neural network approaches all maintain latencies below the 1-second sample time, with times of 231ms, 234ms, and 251ms respectively on embedded hardware. These represent a $1965 - 1809x$ improvement in latency over RUWO, and an $8.6 - 7.9x$ latency improvement over ERA on embedded hardware. Interestingly, our embedded neural networks outperform RUWO even when it is provided a GPU by a factor of $4.6 - 4.25x$.

Tab. II shows the energy consumption of our techniques when run on both embedded and conventional devices. RUWO shows increased energy consumption when run on embedded devices compared to conventional devices due to the decreased compute power and access to parallelism on our embedded device. ERA and our neural networks, however, are lighter and less compute bound than RUWO, meaning overall energy consumption improves when using embedded devices. ERA uses a total of 6 joules when run on our embedded device. Our neural networks, however, use only 0.6-0.67 joules per classification on our embedded device, a $10.6 - 9.7x$ decrease in energy consumption compared

TABLE II
ENERGY CONSUMPTION OF RUWO, ERA, AND OUR NEURAL NETWORKS WHEN EXECUTED ON AN EMBEDDED DEVICE AS WELL AS GENERAL PURPOSE HARDWARE.

Algorithm	Raspberry Pi Energy Consumption (J)	CPU / GPU Energy Consumption (J)
RUWO	1510.5304 ± 14.8451	261.2894 ± 6.5143
ERA	6.4959 ± 0.1020	45.5188 ± 1.4046
NN MSE	0.6139 ± 0.0144	3.6731 ± 0.2849
NN Custom Loss Function	0.6160 ± 0.0128	3.6608 ± 0.0793
NN Split	0.6691 ± 0.0093	4.1077 ± 0.0593

to ERA.

Our technique is, while successful within our experimental parameters, is not without limitations. We explore waveform design within a restrictive context, using only 1024Hz LFM signals. This is useful for many low-bandwidth IoT and radar contexts, but will not produce waveforms for higher bandwidth and frequency contexts within latency goals. Fortunately, there is room for our technique to grow. The rise of low-power accelerators like edge TPUs [22], neuromorphic hardware [8], and FPGAs [23] portend a bright future for low SWaP neural solutions to IoT problems.

VI. CONCLUSIONS

Online waveform design is becoming an increasingly necessary on embedded hardware for IoT devices to communicate clearly with one another. Interference mitigation using waveform design is a well-understood problem, but extant optimization solutions are slow and prior neural solutions generalize poorly. We present an ANN solution to waveform design that automatically designs notched waveforms using a single pass through a lean neural network. Our method greatly outperforms prior solutions in terms of latency (at least $7.9x$) and power consumption (at least $9.7x$) and meets real-time standards for waveform design on the low SWaP embedded hardware on which IoT devices rely without sacrificing accuracy or waveform characteristics.

Acknowledgements: The work herein is the work of the authors and does not represent any position of the US Air Force, Department of Defense, US Government, or Air Force Research Laboratory. This work was approved for public release under case AFRL-2022-0450. This research was funded by SLAKE contract FA8650-19-C-1692 and NSF Graduate Research Fellowship DGE-1343012.

REFERENCES

[1] T. Bihl and M. Talbert, "Analytics for autonomous c4isr within e-government: a research agenda," in *Proceedings of the 53rd Hawaii International Conference on System Sciences*, pp. 2218–2227, 2020.

[2] J. R. Fienup, "Phase retrieval algorithms: a comparison," *Appl. Opt.*, vol. 21, pp. 2758–2769, Aug 1982.

[3] T. Higgins, T. Webster, and A. K. Shackelford, "Mitigating interference via spatial and spectral nulling," *IET Radar, Sonar & Navigation*, vol. 8, no. 2, pp. 84–93, 2014.

[4] J. G. Boubin, N. T. Babu, C. Stewart, J. Chumley, and S. Zhang, "Managing edge resources for fully autonomous aerial systems," in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, pp. 74–87, 2019.

[5] H. Genc, Y. Zu, T.-W. Chin, M. Halpern, and V. J. Reddi, "Flying iot: Toward low-power vision in the sky," *IEEE Micro*, vol. 37, no. 6, pp. 40–51, 2017.

[6] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge computing for autonomous driving: Opportunities and challenges," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1697–1716, 2019.

[7] J. Boubin, A. M. Jones, and T. Bihl, "Neurowav: Toward real-time waveform design for vanets using neural networks," in *2019 IEEE Vehicular Networking Conference (VNC)*, pp. 1–4, 2019.

[8] P. Farr, A. M. Jones, T. Bihl, J. Boubin, and A. DeMange, "Waveform design implemented on neuromorphic hardware," in *2020 IEEE International Radar Conference (RADAR)*, pp. 934–939, 2020.

[9] A. Baietto, J. Boubin, P. Farr, T. J. Bihl, A. M. Jones, and C. Stewart, "Lean neural networks for autonomous radar waveform design," *Sensors*, vol. 22, no. 4, 2022.

[10] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

[11] Y. Goldberg, "A primer on neural network models for natural language processing," *Journal of Artificial Intelligence Research*, vol. 57, pp. 345–420, 2016.

[12] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pp. 265–283, 2016.

[13] F. Chollet *et al.*, "keras," 2015.

[14] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010.

[15] H. Guo and J. Heidemann, "Detecting iot devices in the internet," *IEEE/ACM Transactions on Networking*, vol. 28, no. 5, pp. 2323–2336, 2020.

[16] A. M. Jones and P. M. McCormick, "Spectral notching via error reduction algorithm with relaxed papr constraint," in *2019 IEEE Radar Conference (RadarConf)*, pp. 1–6, IEEE, 2019.

[17] S. Li, L. Da Xu, and S. Zhao, "5g internet of things: A survey," *Journal of Industrial Information Integration*, vol. 10, pp. 1–9, 2018.

[18] R. Pi, "Raspberry pi 3 model b," *online].(https://www. raspberrypi. org*, 2015.

[19] M. B. Yassein, W. Mardini, and A. Khalil, "Smart homes automation using z-wave protocol," in *2016 International Conference on Engineering & MIS (ICEMIS)*, pp. 1–6, IEEE, 2016.

[20] S.-C. Lin, Y. Zhang, C.-H. Hsu, M. Skach, M. E. Haque, L. Tang, and J. Mars, "The architectural implications of autonomous driving: Constraints and acceleration," in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 751–766, 2018.

[21] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1. MIT press Cambridge, 2016.

[22] S. Cass, "Taking ai to the edge: Google's tpu now comes in a maker-friendly package," *IEEE Spectrum*, vol. 56, no. 5, pp. 16–17, 2019.

[23] B. Farley, J. McGrath, and C. Erdmann, "An all-programmable 16-nm rfsoc for digital-rf communications," *IEEE Micro*, vol. 38, no. 2, pp. 61–71, 2018.