Artificial Neural Network based Automatic Modulation Classification over a Software Defined Radio Testbed

Jithin Jagannath[†], Nicholas Polosky [†], Daniel O'Connor [†], Lakshmi N. Theagarajan[‡], Brendan Sheaffer [†], Svetlana Foulke[†], Pramod K. Varshney[‡]

[†]ANDRO Advanced Applied Technology, ANDRO Computational Solutions, LLC, Rome NY, {jjagannath, npolosky, doconnor, bsheaffer, sfoulke}@androcs.com

[‡]Department of EECS, Syracuse University, Syracuse, NY, {ltheagar, varshney}@syr.edu

***Abstract*—Automatic modulation classification (AMC) is an essential component of several intelligent communication systems. In this paper, we design and evaluate a practical AMC system that can be readily deployed to provide robust performance in various real-time commercial scenarios. Thus, our main goal is to develop a robust AMC algorithm with low computational complexity for easy implementation and practical deployment. To this end, we utilize recently revitalized machine learning based approaches used for various classification purposes. In our proposed AMC architecture, we first propose various statistics that serve as features of the AMC signals; next, we design an artificial neural network (ANN) based classifier that performs AMC over a wide range of SNRs. We employ Nesterov accelerated adaptive moment (NADAM) estimation technique to improve the classification performance of our ANN. Further, to establish the practical feasibility of our proposed architecture, we implement it on a SDR testbed. The proposed ANN-based classifier is shown to outperforms the hybrid hierarchical AMC (HH-AMC) [1] system and is flexible enough to easily expand the dictionary of modulation formats for other applications.**

I. Introduction and Background

Software defined radio (SDR) technology has driven communication systems to become more flexible and enables channel dependent adaptation to exploit constrained resources. In SDR communication systems, applications such as authentication, intruder detection, and adaptive transceivers require functions that address detection and modulation classification in the received signal. Within a cooperative system, AMC enables adaptive transceivers to automatically switch modulations based on the channel conditions without the need for a feedback channel between the transmitter and the receiver. In a non-cooperative system, AMC serves to aid intelligence, surveillance, and reconnaissance (ISR) missions by recognizing the modulation of an unknown signal. A wide variety of AMC techniques proposed in the literature can be broadly classified into two types: feature based (FB) [2]–[5] and likelihood based (LB) methods [6]–[10]. It is known that the LB methods provide optimal performance but are often not feasible under restricted computational resources and time per decision (T_{dec}, i.e., the time taken to perform the classification operation) requirements [1], [11]. On the other hand, FB classifiers are computationally efficient and can provide near optimal performance if designed carefully. In this paper, we employ the FB approach to design an artificial neural network (ANN) based classifier that can be implemented on SDRs for realistic civilian applications.

In recent years, different machine learning techniques have been employed to determine the modulation format of the unknown signal via classification. This includes the use of support vector machines (SVMs) [12] and ANNs [13]–[15]. In [13], the authors use a multilayer perceptron (MLP) to classify twelve different modulation formats with high accuracy over a wide range of signal-to-noise ratio (SNR) values. In [14], the authors use six features and evaluate two different ANN architectures trained by the backpropagation method using the standard gradient descent (GD) learning algorithm. Similarly, in [15], eight modulation schemes have been shown to be successfully classified with high accuracy in low SNR conditions. All these studies are limited to simulations and not evaluated on actual hardware. In our previous work [11], we have observed the problems faced during the AMC task while transitioning from simulation to hardware implementation. Due to the assumptions and unanticipated signal distortions that are overlooked during simulations, real-time performance of the above mentioned AMC techniques may be degraded compared to simulations. Our previous work [8] depicted scenarios where the optimization would end up at local minima leading to poor performance. This can also be the case with ANN based AMC systems during their learning phase in the absence of adequate learning techniques. We propose an ANN based architecture for AMC that shall address the issues outlined above. The major contributions of this paper can be summarized as follows,

- We propose two novel statistical features that enable efficient classification of both linear and non-linear modulation formats.
- We propose an expandable ANN architecture that employs Nesterov Accelerated Adaptive Moment Estimation (NADAM) algorithm for learning and performing the AMC task of signals over a wide range of SNR values.
- We establish the feasibility of the proposed solution through hardware implementation and extensive performance evaluation of the proposed algorithm.

[1]ACKNOWLEDGMENT OF SUPPORT AND DISCLAIMER:(a) Contractor acknowledges Government's support in the publication of this paper. This material is based upon work supported by the US Air Force Research Laboratory under AFRL Contract No. FA8750-16-C-0085. (b) Any opinions, findings and conclusions or recommendation expressed in this material are those of the author(s) and do not necessarily reflect the views of AFRL.

The rest of the paper is organized as follows. The system model is presented in Section II. The preprocessing and feature extraction methods are presented in Section III. The training processes and learning algorithms for the proposed ANN are presented in Section IV and Section V, respectively. The results of the hardware experiments are presented in Section VI. Finally, the conclusions are presented in Section VII.

II. System Model

Consider the baseband discrete-time signal at a receiver in a flat fading environment,

$$y(n) = h(n)x(n) + w(n), \quad n = 1, ..., N \tag{1}$$

where $x(n)$ is the discrete-time transmitted signal, $h(n)$ is the complex valued channel gain that follows a Gaussian distribution and $w(n)$ is the additive complex zero-mean white Gaussian noise process at the receiver with two-sided power spectral density (PSD) $N_0/2$. For this system model, the overall ANN based AMC system has three main blocks as shown in Fig. 1.

Fig. 1: System overview of ANN based AMC

(i) **Signal preprocessing.** This component performs signal processing designed to enhance the quality of the received samples. This can include various amplifiers, filters, automatic gain control (AGC), and frequency-offset correction, among others.

(ii) **Feature extraction.** In this block, features pertaining to amplitude, phase, and frequency of the received signal are extracted from the preprocessed samples. Various signal statistics such as moments and cumulants are often used as features.

(iii) **Signal classifier.** This is the final component that houses various machine learning based classifiers that are appropriate for the application. In this paper, we design an ANN based classifier for the AMC task.

III. Preprocessing and Features Extraction

To ensure that the proposed AMC algorithm can be easily ported to platforms with limited computational capability, we set an initial processing constraint and assume that very little resources can be spared for preprocessing. If the target platform has various preprocessing modules that are used by the receiver regardless, AMC can choose to use them to its advantage. In this work, we only implement AGC in the preprocessing block. AGC normalizes the amplitude of the received signal to mitigate the effects of channel gain. This helps to provide numerical stability to the values of the features that depend on the amplitude of the signal.

Next, we discuss several features that are extracted from the preprocessed samples before being fed to the ANN classification block. Some of the features like amplitude variance ($Var(|y(t)|)$), maximum value of the power spectral density (PSD) of the normalized centered-instantaneous amplitude (γ_{max}), and higher order statistics have been used previously [1], [2]. In this work, we introduce two new features, namely, *in-band spectral variation* ($Var(f)$) and *deviation from unit circle* (Δ_o) to be used for classification.

The first feature that we propose is the variance of the received signal amplitude which is given by,

$$Var(|y(n)|) = \frac{\sum_{N_s}(|y(n)| - \mathbb{E}(|y(n)|))^2}{N_s}, \tag{2}$$

where $|y(n)|$ is the absolute value of the over-sampled signal and $\mathbb{E}(|y(n)|)$ represents the mean computed from N_s samples. This feature is employed primarily to distinguish FSK signals (CPFSK, GFSK, GMSK) from the rest of the modulation formats (PSKs and QAMs). The next feature we consider is the maximum value of PSD of the normalized centered-instantaneous amplitude [2] given as,

$$\gamma_{max} = \frac{max|FFT(a_{cn}(n))|^2}{N_s}, \tag{3}$$

where $a_{cn}(n) \triangleq \frac{a(n)}{m_a} - 1$, $m_a = \frac{1}{N_s}\sum_{n=1}^{N_s} a(n)$, and $a(n)$ is the absolute value of the complex-valued received signal. Normalization by the instantaneous amplitude is required to compensate for the channel gain. The feature γ_{max} gives us a measure of deviation of the PSD of the signal from its average value. We use both the $Var(\gamma_{max})$ and $\mathbb{E}(\gamma_{max})$ as a part of the feature vector.

Some higher order statistics like cumulants are efficient in classifying amplitude and phase modulated signals [3], [4]. Cumulants are statistical measures that are known to be invariant to certain distortions in random signals. Hence, they are very suitable for the purpose of modulation classification. The cumulant of a random signal is a function of two parameters, l denotes the order of the cumulant and k denotes the number of conjugations involved in the computation of the cumulant ($k \leq l$). The l^{th} order cumulant of the random signal y with k conjugations can be computed as,

$$C_{lk} = \sum_{p}^{\text{No. of partitions in } l} (-1)^{p-1}(p-1)! \prod_{j=1}^{p} \mathbb{E}\{y^{l_j - k_j} y^{*k_j}\}, \tag{4}$$

where l_j and k_j correspond to the subsets in the partition j. In our design, we use the fourth order cumulants, specifically, we use the ratio C_{40}/C_{42} as the next element of the feature vector. These cumulants can be evaluated as,

$$C_{42} = \mathbb{E}(|y|^4) - |\mathbb{E}(y^2)|^2 - 2\mathbb{E}(|y|^2)^2, \tag{5}$$

$$C_{40} = \mathbb{E}(y^4) - 3\mathbb{E}(y^2)^2. \tag{6}$$

Though the variance of γ_{max} provides a distinguishing feature between the non-linear modulations, the computation of the peak spectral values is often affected due to oversampling and spurious noise. We propose to use an additional feature termed as the *in-band spectral variation*, $Var(f)$, that captures the required frequency variation. Due to the abrupt change in the phase of the signal, GFSK spectrum has multiple spectral crests. The CPFSK spectrum is relatively smoother

TABLE I: Δ_0 for QAMs.

Modulation	$\lim_{SNR\to\infty}\mathbb{E}(\Delta_0)$	$\lim_{SNR\to\infty}\mathrm{Var}(\Delta_0)$
16 QAM	0.2639	0.0479
32 QAM	0.2926	0.0331
64 QAM	0.3050	0.0412
128 QAM	0.2976	0.0383
256 QAM	0.3063	0.0438

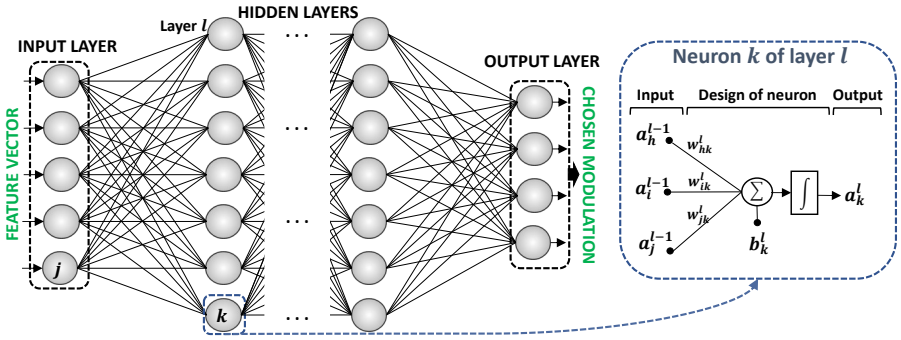

Fig. 2: Structure of the artificial neural network

due to the lack of abrupt phase shifts. Finally, the GMSK spectrum has the advantage of both continuous phase shift and Gaussian pulses; therefore, GMSK spectrum is relatively smoother and decays faster over frequency to reduce out of band emission. Therefore, the degree of smoothness of the spectrum inside the band of interest $[-f_i,+f_i]$ becomes an appropriate classification feature. We define $Var(f)$ as

$$Var(f) = Var\Big(\mathcal{F}\big(y(t)\big)\Big), \tag{7}$$

where $\mathcal{F}(y(t)) = \big\{Y(f)-Y(f-F_0)\big\}_{f=-f_i}^{+f_i}/F_0$, F_0 is the step size, and $Y(f) = FFT(y(t))$.

In linear modulations, we have to classify PSK and QAM. M-PSK signals can be represented as $\{Ae^{-j\frac{2\pi i}{M}}\}_{i=1}^{M}$. Therefore, irrespective of the value of M, normalized M-PSK signals always lie on the unit circle. We exploit this property of the M-PSK signals and propose a new feature to classify between PSK and QAM signals. The feature $Var(\Delta_o)$ computes the variance of the deviation of the normalized received signal from the unit circle. This is given as follows,

$$\Delta_o = \frac{|y(t)|}{\mathbb{E}(|y|)} - 1. \tag{8}$$

For PSK signals,

$$\begin{aligned}\lim_{SNR\to\infty}\Delta_o &= \lim_{n(t)\to 0}\frac{|hx(t)+n(t)|}{\mathbb{E}(hx(t)+n(t))} - 1\\ &= \frac{\Big|A|h|\big(arg(h(t)) - 2\pi i(t)/M\big)\Big|}{\mathbb{E}\Big|A|h|\big(arg(h(t)) - 2\pi i(t)/M\big)\Big|} - 1\\ &= 1-1 = 0.\end{aligned} \tag{9}$$

Further, from (9), we can write

$$\lim_{SNR\to\infty}\mathbb{E}(\Delta_o) = 0, \quad \lim_{SNR\to\infty} Var(\Delta_o) = 0. \tag{10}$$

For QAM signals, the theoretical values of this feature are tabulated in Table I. All these features discussed above and the estimated SNR values are fed into the ANN in the form of feature vector.

IV. Training the ANN

ANNs have been deployed to solve various classification problems in multiple fields. The widespread use of ANNs can be attributed to the plasticity of the default structure to fit the needs of specific problems. Generally, ANNs are comprised of an input layer, an output layer, and one or more hidden layers as shown in Fig. 2. The neurons associated with the network are contained within the different types of layers mentioned above. The input layer contains one neuron for each feature in the feature vector of the training data. The number of neurons within each hidden layer can be interpreted as a metric for model complexity and is set prior to the training of the model.

The weighted connections between layers, and the computation occurring at each neuron is depicted in Fig. 2 and mathematically summarized below. The weighted connection at neuron k in layer l from neuron j in layer $l-1$ is given as w_{jk}^l. At each neuron in the hidden and output layers the weighted sum of the previous layer's outputs is calculated as follows,

$$z_k^l = \sum_j w_{jk}^l * a_j^{l-1} + b_k^l \tag{11}$$

where a_j^{l-1} is the output from neuron j in the previous level and b_k^l is a bias term added at each neuron. This sum is then taken as the input to the activation function where the output of the neuron is calculated. The activation function we use in the hidden layers of the network is the *sigmoid* function and is defined as,

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{12}$$

Thus, the output of a given neuron, k, in hidden layer l is defined as,

$$a_k^l = \sigma(z_k^l) \tag{13}$$

In the output layer, we use the *softmax* function to compute the activation as opposed to the $\sigma(.)$. The *softmax* function is given below for the output activation, a_k^o,

$$a_k^o = \frac{e^{z_k^l}}{\sum_i e^{z_i^l}} \;\; \forall \;\; k = 1,...,M \tag{14}$$

where M is the number of neurons in the output layer which also corresponds to the number of modulation formats considered. The motivation behind using the *softmax* activation function in the output layer as opposed to the $\sigma(.)$ stems from the format of the output labels. We encode a network decision using a one-hot vector, which takes the form of a $1\times M$ vector consisting of $M-1$ zeros and a single index that contains a one. The index containing a one is representative of the modulation that has been selected by the network. Prior to

training, each of the M modulations is associated with a single index in a one-to-one mapping to the one-hot output vector, and retains it's association to that index throughout the training and testing processes. The output of the *softmax* function is, $a_k^0 \in [0,1]$ such that $\sum_k a_k^0 = 1$. This can be thought of as a probability distribution over the different possible outcomes. The modulation corresponding to the maximum value in the *softmax* output vector is the classification result.

In our network design, we use the cross entropy loss function that is suitable for a multiclass output. The gradient of the loss function with respect to the weights is calculated such that the error of the network output can be backpropagated to update the network weights to produce the desired output. The cross entropy loss function for a multiclass output is given as,

$$L = -\sum_k y_k \log(a_k^o) \tag{15}$$

where y_k is the expected output and a_k^o is the output from the *softmax* function in the output layer. The gradient with respect to the weights preceding the output layer can be derived using the formula,

$$\frac{\partial L}{\partial w_{jk}} = \frac{\partial L}{\partial a_k^o}\frac{\partial a_k^o}{\partial z_k^l}\frac{\partial z_k^l}{\partial w_{jk}} \tag{16}$$

The error gradient with respect to a single output in the output layer is given as,

$$\frac{\partial L}{\partial a_k^o} = \frac{-y_k}{a_k^o} \tag{17}$$

The partial derivative of a given *softmax* output, a_k^o, with respect to the weighted sum input, z_k^l, has dependencies on the weighted sums of all of the neurons in the output layer. Thus, the partial derivative $\frac{\partial a_k^o}{\partial z_i^l}$ takes on different values when $k = i$ and when $k \neq i$, given respectively as,

$$\frac{\partial a_k^o}{\partial z_k^l} = a_k^o(1-a_k^o) \text{ and } \frac{\partial a_k^o}{\partial z_i^l} = -a_k^o a_i^o \tag{18}$$

Using the above derivatives, the error gradient with respect to a given weighted sum in an output neuron can be reduced to,

$$\frac{\partial L}{\partial z_k} = \sum_i \frac{\partial L}{\partial a_i^o}\frac{\partial a_i^o}{\partial z_k} = a_k^o - y_k \tag{19}$$

The error gradient with respect to the weighted sum can then be further backpropagated to the weights contained in the hidden layers in a similar fashion. The main difference to consider for backpropagation in the hidden layers is that the outputs from the neurons in the hidden layers were calculated using the $\sigma(.)$, and thus their error gradient needs to be backpropagated using the derivative of the $\sigma(.)$, which is,

$$\frac{\partial \sigma(z)}{\partial z} = \sigma(z)(1-\sigma(z)) \tag{20}$$

Through repeated error backpropagation, the optimal weights are learned by the network. The process by which this consecutive backpropagation occurs is called gradient descent (GD). Stochastic GD examines one training instance at a time and backpropagates the corresponding error before moving on to the following training instance. Batch GD forward propagates all training examples and then backpropagates an error averaged over the entire training set. In our implementation, we use stochastic mini batch GD, which forward propagates a smaller subset of the training set which is followed by a backpropagation of an error averaged over the smaller subset. The advantage of using stochastic mini batch GD is an improvement in runtime from stochastic GD without the drastically inaccurate weight adjustments associated with batch GD.

V. Learning Algorithms

Further improvements in training runtime can be achieved by adjusting the learning algorithm used to update the weights contained in the network. In our network, we implemented the NADAM algorithm, which is a combination of the Nesterov's Accelerated Gradient (NAG) algorithm and the Adaptive Moment Estimation (ADAM) algorithm. The NADAM algorithm has been shown to achieve a lower training and validation loss compared to other learning algorithms in [16]. In the algorithms discussed below, $f_t(\phi)$ denotes the loss function at timestep t parameterized by ϕ. The gradient of the loss function with respect to the parameters at timestep t is given as g_t, and the learning rate is represented by η.

Algorithm 1 Nesterov's Accelerated Gradient

$g_t \leftarrow \nabla_{\phi_{t-1}} f_t(\phi_{t-1} - \mu m_{t-1})$
$m_t \leftarrow \mu\, m_{t-1} + \eta g_t$
$\phi_t \leftarrow \phi_{t-1} - m_t$

The NAG algorithm, shown in Algorithm 1, improves upon the standard GD algorithm by incorporating a momentum term, m_t. The momentum term essentially accumulates a sum of previous parameter updates and then multiplies them by some constant decay factor (μ). This allows the algorithm to move quickly over dimensions with smaller gradients pointing in the same direction in the problem space while slowing the algorithm when the gradient update directions oscillate back and forth in the problem space. The NAG algorithm also updates the parameters with a momentum step prior to calculating the gradient at a given t. This allows the algorithm to obtain a gradient calculation that will lead the algorithm more closely along the optimal path. A learning rate annealing schedule is often used in GD algorithms to help achieve an optimal solution. Learning rate annealing decreases the learning rate by some factor as the number of parameter updates increases. The learning rate annealing factor is omitted from the algorithms presented below as the learning rate is not typically annealed at every t.

Algorithm 2 Adaptive Moment Estimation algorithm

$g_t \leftarrow \nabla_{\phi_{t-1}} f(\phi_{t-1})$
$m_t \leftarrow \mu\, m_{t-1} + (1-\mu) g_t$
$\hat{m}_t \leftarrow \frac{m_t}{1-\mu^t}$
$n_t \leftarrow \nu\, n_{t-1} + (1-\nu) g_t^2$
$\hat{n}_t \leftarrow \frac{n_t}{1-\nu^t}$
$\phi_t \leftarrow \phi_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{n}_t}+\varepsilon}$

The difference between NAG and ADAM is in the definition of the momentum term given in Algorithm 1 and Algorithm 2 respectively. ADAM also incorporates a velocity term, n_t, which is an average of previous squared gradient multiplied by an exponentially decaying term, ν. The use of previous gradients rather than previous parameter updates will allow the algorithm to make progress towards the optimal solution regardless of how much the learning rate has been annealed. This allows for more fine grained convergence near the end of training.

Algorithm 3 Nesterov Accelerated Adaptive Moment Estimation algorithm

$g_t \leftarrow \nabla_{\phi_{t-1}} f(\phi_{t-1})$
$\hat{g} \leftarrow \frac{g_t}{1-\prod_{i=1}^{t} \mu_i}$
$m_t \leftarrow \mu\, m_{t-1} + (1-\mu) g_t$
$\hat{m}_t \leftarrow \frac{m_t}{1-\prod_{i=1}^{t} \mu_i}$
$n_t \leftarrow \nu\, n_{t-1} + (1-\nu) g_t^2$
$\hat{n}_t \leftarrow \frac{n_t}{1-\nu^t}$
$\overline{m}_t \leftarrow (1-\mu_t)\hat{g}_t + \mu_{t+1}\hat{m}_t$
$\phi_t \leftarrow \phi_{t-1} - \eta \frac{\overline{m}_t}{\sqrt{\hat{n}_t}+\varepsilon}$

The two previously mentioned parameter update algorithms can be combined to form the NADAM algorithm which is given in Algorithm 3. The NADAM algorithm essentially takes the properties of both previous algorithms and incorporates them into one algorithm. This allows the resulting combined algorithm to make use of previous parameter updates, previous gradients, and squared gradients similar to the ADAM algorithm. An initial momentum step is incorporated as well, gaining the advantage of a higher quality subsequent gradient step as seen in the NAG algorithm. The NADAM algorithm was used to update the weights of the network used in our implementation.

VI. Performance Evaluation on Testbed

Fig. 3: Software defined radio testbed

The testbed consists of two USRP (universal software radio peripheral) X300s equipped with CBX-120 daughterboards (see Fig. 3), which cover frequency ranges from 1.2 GHz to 6 GHz with up to 120 MHz of instantaneous analog bandwidth. Though USRPs are capable of multiple-input and multiple-output (MIMO) operational mode but we do not use this capability in this experimental setup. The analog-to-digital and digital-to-analog converters on the motherboard use a 200 MHz master clock and sample at 200 MS/s and 800 MS/s, respectively. The Linux-based host PC interfaces with the USRP using a Gigabit Ethernet (GigE) connection. The SDR setup used to perform classification is similar to [17], but uses the ANN based classifier and HH-AMC to perform a comparative study.

One USRP was setup as the receiver while the other was setup as transmitter and configured to transmit BPSK, QPSK, 8PSK, 16QAM, CPFSK, GFSK and GMSK at different transmit (TX) powers. Along with TX gain, attenuators are also used to vary the TX power and achieve a wider range of operational SNR scenarios. Therefore, the experiments are conducted over an estimated SNR ranging from 5 dB to 45 dB. At each SNR level, samples were collected to have as many as 800 training examples and 200 test examples for each of the seven modulations which totaled to 28000 training and 7000 testing examples respectively for each seed. Therefore, each probability of correct classification (P_{cc}) value in the experiments is an average of 10 seeds each consisting of 200 decisions made for each modulations per transmit scenario.

The proposed ANN architecture was implemented using Python and C++ programing languages. ANNs have a large number of hyperparameters including number of layers, number of neurons, learning rate, and activation function that have to be set before the training process. The chosen activation functions and learning rate have been discussed in Section IV and Section V. In this work, to examine some hyperparameters, we implemented three different ANN configurations to compare the proposed approach. In our first implementation, we used two hidden layers with 50 and 25 neurons respectively. Adding more hidden layers enables the network to model a higher complexity mapping from inputs to outputs; however, it is often the case that lower complexity architectures outperform high complexity architectures in practice because they generalize well to unseen data. Therefore, the other two implementations of the network consist of one hidden layer and three hidden layers respectively. To examine how the proposed framework behaves when enhanced by adding a new feature, we evaluate the three configurations with (Fig. 5) and without (Fig. 4) the $Var(f)$ in the feature vector. In these experiments, each configuration was also evaluated with and without the estimated SNR in the feature vector.

According to Fig. 4, providing the estimated SNR value improves the performance of the classifier regardless of the ANN configuration. At the same time, it is important to recognize that the difference in performance is not significant. Therefore, if the resources required to estimate SNR on the target platform are unavailable, the proposed classifier can still perform satisfactorily in the absence of estimated SNR. Overall, the one layer configuration of ANN outperforms the other, more complex, configurations. The smaller number of model parameters allows the single layered network to generalize better to unseen data. Comparing Fig. 4 and Fig. 5, the classification performance of different configurations improves with the addition of feature ($Var(f)$). This shows that the ability of the proposed ANN framework can be further expanded by adding more features to improve P_{cc} and/or expand types of modulation formats that can be classified.

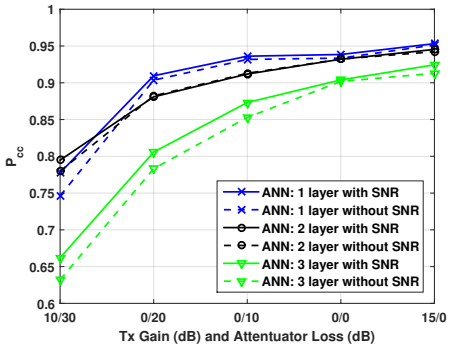

Fig. 4: P_{cc} vs SNR (without $Var(f)$)

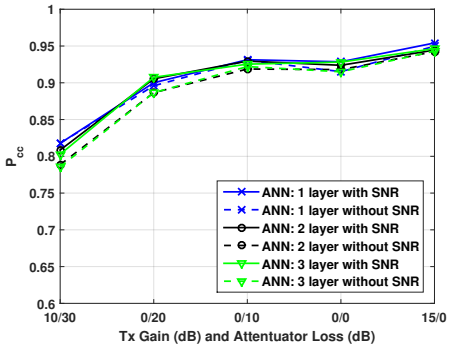

Fig. 5: P_{cc} vs SNR (with all features)

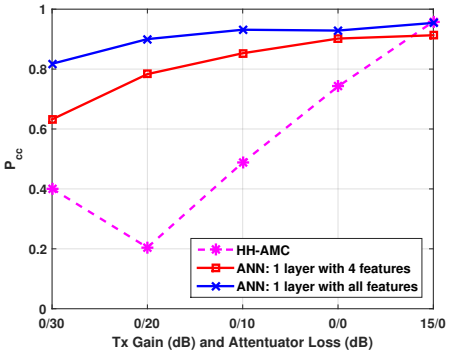

Fig. 6: P_{cc} vs SNR

Next, the ANN configuration with the associated hyperparameters that performed the best in our experiments, namely the one-layer ANN with all the features and SNR, is compared with HH-AMC [1] that employs the same features along with a decision tree to perform classification. Since we introduced new features in the proposed classifier, for fairness, we evaluated an ANN-based classifier that only uses features used by HH-AMC. It can be seen from Fig. 6 that even the ANN that uses only the features used by HH-AMC (only 4 features) outperforms the decision tree based HH-AMC at lower SNR values. Furthermore, it shows how adding the new features introduced in this paper further enhances the ANN-based classifier and achieves high P_{cc} on actual SDR hardware based experiments. The sudden spike of $P_{cc} = 0.4$ for HH-AMC at the lowest SNR ($0/20$ on x-axis) is attributed to the classifier favoring three modulations regardless of actual transmission. This led to high P_{cc} for the favored modulation formats leading to a sudden spike in average P_{cc}. This is still ineffective as a reliable classifier. Finally, we want to highlight that the T_{dec} is not impacted by the proposed ANN-based classifier, as all ANN configurations were trained offline. The worst case average T_{dec} among different transmitted modulation for ANN is 285 ms as compared to 395 ms of HH-AMC.

VII. Conclusions

In this work, we have introduced a novel practical ANN-based AMC approach that yields reliable, near-real time classification performance for commercial applications. To accomplish this, we used features that are currently used by the community and also introduced two more features that further enhance classification performance. We used stochastic mini batch gradient descent to efficiently train the ANN and employed the NADAM algorithm for gradient descent optimization. The proposed solution was implemented using SDRs and shown to outperform computationally efficient HH-AMC achieving P_{cc} upto 0.98. In the future, the proposed ANN-based architecture can be further expanded by including more features to improve P_{cc} and/or increase the number of modulation formats that can be classified.

References

[1] J. Jagannath, D. O'Connor, N. Polosky, B. Sheaffer, L. N. Theagarajan, S. Foulke, P. K. Varshney, and S. P. Reichhart, "Design and Evaluation of Hierarchical Hybrid Automatic Modulation Classifier using Software Defined Radios," in *Proc. of IEEE Annual Computing and Communication Workshop and Conference (CCWC)*, Las Vegas, NV, Jan 2017.

[2] A. Hazza, M. Shoaib, S. AlShebeili, and A. Fahd, "Automatic modulation classification of digital modulations in presence of HF noise." *EURASIP Journal on Adv. in Signal Processing*, vol. 2012, p. 238, 2012.

[3] D. C. Chang and P. K. Shih, "Cumulants-based modulation classification technique in multipath fading channels," *IET Communications*, vol. 9, no. 6, pp. 828–835, 2015.

[4] A. Swami and B. M. Sadler, "Hierarchical digital modulation classification using cumulants," *IEEE Transactions on Communications*, vol. 48, no. 3, pp. 416–429, Mar 2000.

[5] L. Han, F. Gao, Z. Li, and O. Dobre, "Low Complexity Automatic Modulation Classification Based on Order-Statistics," *IEEE Transactions on Wireless Communications*, vol. PP, no. 99, pp. 1–1, 2016.

[6] F. Hameed, O. Dobre, and D. Popescu, "On the likelihood-based approach to modulation classification," *IEEE Transactions on Wireless Communications*, vol. 8, no. 12, pp. 5884–5892, December 2009.

[7] T. Wimalajeewa, J. Jagannath, P. K. Varshney, A. Drozd, and W. Su, "Distributed asynchronous modulation classification based on hybrid maximum likelihood approach," in *Proc. of IEEE Military Communications Conference (MILCOM)*, Tampa, FL, Oct 2015.

[8] Y. Zhang, N. Ansari, and W. Su, "Optimal Decision Fusion Based Automatic Modulation Classification by Using Wireless Sensor Networks in Multipath Fading Channel," in *Proc. of IEEE Global Telecommunications Conference (GLOBECOM)*, Houston, TX, Dec 2011.

[9] B. Dulek, O. Ozdemir, P. K. Varshney, and W. Su, "Distributed Maximum Likelihood Classification of Linear Modulations over Nonidentical Flat Block-Fading Gaussian Channels," *IEEE Transactions on Wireless Communications*, vol. 14, no. 2, pp. 724–737, Feb 2015.

[10] O. Ozdemir, T. Wimalajeewa, B. Dulek, P. K. Varshney, and W. Su, "Asynchronous Linear Modulation Classification with Multiple Sensors via Generalized EM Algorithm," *IEEE Transactions on Wireless Communications*, vol. 14, no. 11, pp. 6389–6400, Nov 2015.

[11] S. Foulke, J. Jagannath, A. Drozd, T. Wimalajeewa, P. Varshney, and W. Su, "Multisensor Modulation Classification (MMC): Implementation Considerations – USRP Case Study," in *Proc. of IEEE Military Communications Conference (MILCOM)*, Baltimore, MD, Oct 2014.

[12] H.-Y. Liu and J.-C. Sun, "A modulation type recognition method using wavelet support vector machines," in *Proc. of IEEE Intl. Congress on Image and Signal Processing (CISP)*, Tianjin, China, Oct 2009.

[13] J. J. Popoola and R. v. Olst, "A novel modulation-sensing method," *IEEE Vehicular Technology Magazine*, vol. 6, no. 3, pp. 60–69, Sept 2011.

[14] M. M. Roganovic, A. M. Neskovic, and N. J. Neskovic, "Application of artificial neural networks in classification of digital modulations for software defined radio," in *Proc. of IEEE EUROCON*, St. Petersburg, Russia, May 2009.

[15] J. J. Popoola and R. v. Olst, "Effect of training algorithms on performance of a developed automatic modulation classification using artificial neural network," in *Proc. of IEEE AFRICON*, Pointe-Aux-Piments, Mauritius, Sept 2013.

[16] T. Dozat, "Incorporating nesterov momentum into adam," in *Proc. of International Conference on Learning Representations*, San Juan, Puerto Rico, Feb 2016.

[17] J. Jagannath, H. M. Saarinen, and A. L. Drozd, "Framework for automatic signal classification techniques (FACT) for software defined radios," in *Proc. of IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, Verona, NY, May 2015.