Modulation Classification Using Neural Networks

Venkatesh Sathyanarayanan, Jahya Burke, Rui Shang, Richard Bell
Electrical and Computer Engineering University of California San Diego, San Diego, CA 92093
Emails: vesathya@eng.ucsd.edu, jahya1burke@gmail.com, r4shang@ucsd.edu, rcbell@eng.ucsd.edu

***Abstract*—In this paper we explore the performance of neural network based modulation classification systems, specifically the convolutional neural network (CNN) and the residual neural network (ResNet). Using the PyTorch machine learning library, we are able to get comparable results reported by other researchers using different frameworks. We also explore the effects that frequency offset and frame boundary misalignment have on the results.**

I. Introduction

In this section we will go over the motivation behind modulation classification and related work, followed by a brief on what the inputs and ouputs are in the last paragraph. Modulation is the manipulation of the amplitude, frequency or phase of an electromagnetic (EM) wave with the intent of transmitting information. The transmitter and receiver will agree to particular modulation scheme ahead of time to allow this information sharing to occur. There are use cases such as dynamic spectrum sensing in the commercial space and electronic warfare in militiary where one would like to infer the modulation type given just the received signal. This is known as modulation classification. The field of modulation classification is an extension of the field of signal detection, whose theory was developed in the 1950s [1], [2]. The signal detection problem can be considered as a binary classification problem in which the output classes are either; signal is present, or it is not. In the detection theory world these are referred to as hypotheses, where H_0 is the null hypothesis when no signal is present and H_1 is the hypothesis when signal is present. In the sampled data domain, these hypotheses can be expressed as,

$$H_0 : r[n] = w[n]\,, \tag{1a}$$

$$H_1 : r[n] = s[n] + w[n]\,, \tag{1b}$$

where $r[n]$ is the received sampled signal, $s[n]$ is the true transmitted signal and $w[n]$ is a noise signal. The true signal can be expressed as

$$s[n] = s_I[n]\cos(2\pi f_c n) - s_Q[n]\sin(2\pi f_c n)\,, \tag{2a}$$

$$= \text{Re}\left\{\tilde{s}[n]e^{2\pi f_c n}\right\}\,, \tag{2b}$$

where $\tilde{s}[n] = s_I[n] + js_Q[n]$ is the complex baseband representation of the passband signal and f_c is the carrier frequency.

Modulation classification extends the signal detection problem by allowing more output classes to exist. It is not enough to determine whether a signal is present, now it is desired to determine what type of signal is present. Some of the earliest work in this area was done in the 1980s. In these early works, hand crafted features were created from the raw time domain signal, such as zero crossing locations [3], square law classifiers [4], statistical moment classifiers [5] and phase based classifiers [4]. Surprisingly, the first neural network approach at modulation classification was also attempted at this time [6]. The authors in this work still created hand crafted features but fed these features into a neural network for the final classification process.

More modern approaches to modulation classification taking advantage of advancements in computing software/hardware [7], [8] that allow reasonable training and testing times and advances in deep learning research [9]–[13] that reduce the vanishing gradient problem seem to begin in 2016 [14]–[16]. These approaches do not use hand crafted expert features to classify the modulations, instead they depend on data alone to train deep neural networks. It should be noted that cyclostationary based modulation classifiers are still relevant and seem to perform well [17]–[19], though we did not investigate them in this article.

In this project each input frame is a digitized EM wave of five milliseconds which corresponds to a matrix of size 2 by 1024, where the two corresponds to the inphase and quadrature phase components of an EM wave. Our output is one of the eleven modulation types. In the following Section II we describe a wireless sytem model and give an intuition behind what a neural network should learn, in order to classify the modulation type. In Section III, we go over the details of the synthetic dataset generation that was used for the project. We will use the MATLAB code from [20] to create our dataset. In Section IV, we explain the details of the models used. We used the CNN model from [14] and a Resnet from [21] as a template and trained and tested the models on our synthetic data. In Section V, we go over the results of using the CNN with relu, CNN with tanh and ResNet with relu. We present results of further tests which were run to understand the black box of the CNN, namely the effect of frequency error on the performance on the CNN model, performance changes of the CNN model when a carefully chosen subset of classes were used for training and testing, effect of misaligned frame window where data from more than one class is part of a frame. Sections VI and VII are conclusions and individual contributions respectively.

Our main contribution in this project is successfully implementing models in PyTorch for modulation classification using CNN and ResNet and getting good accuracies of over 80%. Additionally we were able to use our physics knowledge

of wireless communications to pry open the black box of CNN and infer that similar modulation types are more prone to misclassification and also that phase sensitive modulation types are most vulnerable.

II. Wireless System Model

In this section we will give an intuition on what the neural network has to learn in order to to classify the modulation type of a wireless signal. An electromagnetic wave is affected by a multitude of non-linear artifacts . An ideal neural network will have to be trained over the space of all occurrences of the non-linear artifacts. This is not practically possible and we therefore need to be smart in ensuring that we use a reasonably exhaustive dataset. This complicates the training data generation phase if synthetic data is to be used because proper care must be taken to impart the necessary distortions to the data.

For example, Figure 1 shows the ideal complex baseband symbol values for three different modulations types; BPSK, QPSK and 16QAM. The undistorted digital baseband representations for these modulations correspond to

$$\text{BPSK} : \tilde{s} \in \{\pm 1\}\,, \tag{3a}$$

$$\text{QPSK} : \tilde{s} \in \{\pm 1 \pm j\}\,, \tag{3b}$$

$$\text{16QAM} : \tilde{s} \in \{\pm 1 \pm j, \pm 1 \pm 3j, \pm 3 \pm j, \pm 3 \pm 3j\}\,. \tag{3c}$$

If a neural network were trained using samples drawn from the clean symbol sets such as in (3) and then tested on live signals using a software defined radio unit such as the universal software radio peripheral (USRP), poor classification performance would result. The reason for this, as detailed in ([22]), is that the real life signal will be a specific realization of 4, with all its non-linear artifacts, where

$$r(t) = Ae^{j2\pi\Delta f t}e^{j\theta}\sum_{k=1}^{K} e^{j\phi_k}\tilde{s}_k g(t-(k-1)T-\epsilon T)\,, \quad 0 \le t \le KT \tag{4}$$

fully represents the standard channel effects that would be imparted onto the signal minus additive complex Gaussian noise. It will be assumed that $r(t)$ contains this missing noise term throughout. In (4), A is the attenuation due to path loss between transmitter and receiver, Δf is the carrier frequency offset due to oscillator imperfections, θ is carrier phase offset due to the distance between transmitter and receiver, ϕ_k is phase jitter due to the inaccuracies of the oscillators, $g(t)$ is the effective impulse response of the channel given by the convolution of the transmitter pulse shaping filter and the channel impulse response, T is a symbol period, $\epsilon < T$ is the time offset from the start of a symbol period and $\{\tilde{s}_k\}_{k=1}^{K}$ are K complex transmitted data symbols drawn from a finite size modulation format such as those in (3). We may represent the set of parameters that define (4) through the vector $\boldsymbol{u} = \left[A, \Delta f, \theta, \{\phi_k\}_{k=1}^{K}, g(t), \epsilon, \{\tilde{s}_k\}_{k=1}^{K}\right]^T$. The goal is to find a function that robustly maps signals with various values of $\boldsymbol{u}$ to the proper modulation type

$$i = f(r(t))\,, \tag{5}$$

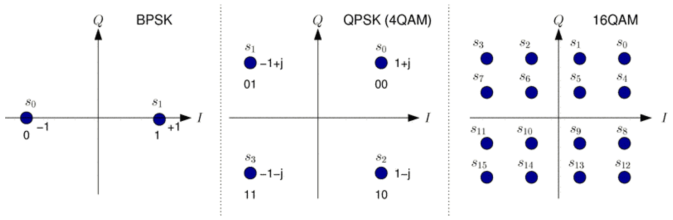

Fig. 1. Constellation diagrams for BPSK, QPSK and 16QAM.

where i is the index over the set of all modulation types expected. Using the fact that neural networks are universal function approximators [23], we use a neural network to approximate the function f. Thus it is expected that the neural network with multitude of parameters, will be able to learn the non-linearities reasonably accurately.

III. Dataset and Features

The field of wireless communications is a mature field and we are able to model all the artifacts as described in equation (4) within an accuracy that would suffice in most commercial applications. Each input frame represents 2×1024 samples of a complex baseband modulation extending over 5 ms of time, where the two corresponds to the inphase and quadrature components of the sampled signal. The output is one of the eleven modulation types. We have adopted the MATLAB code from [20] to create the synthetic dataset of 10,000 samples per class. See Table I for the specifics on the values of the parameters used to generate data. The choice of the values used as shown in the table is based on the implementation in [20]. We use this data for training and testing CNN and ResNet models. Please note that the values chosen are such that there is one strong line of sight path, low values of frequency errors both from clock jitter and Doppler. These are reasonably mild conditions in comparison to a lot of real life cases especially in dense cities. Also the clock jitter on cheaper sources of transmitter can be expected to be much higher. More adverse conditions makes the problem challenging and we are therefore focusing on these milder channel conditions in this project.

To test the effect of frequency error on the performance of the CNN model, we generated test data by changing the maximum clock offset and maximum Doppler parameters. For frame misalignment tests, we reused already generated synthetic data to create new test data by adding controlled portions of frames from two different classes. Please note that since application of neural network to wireless communications is a nascent field, there is not any benchmark data set that is widely used. The closest to a benchmark dataset would be RadioML 2018.01A from [24]. Using this dataset that contains both real and synthetic data would be something that we would want to attempt in the future. We did try using the dataset for testing our model, but its size was too big resulting in the datahub kernel crashing.

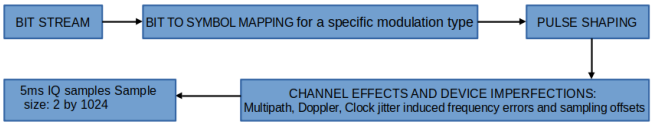

Fig. 2. Steps involved in creating synthetic data

TABLE I
TRAINING DATASET PARAMETERS

Training dataset parameters	
Number of Classes	11
Number of input frames per class	10000
Input dimension	2 by 1024
Duration of each input frame	5 ms
Center frequency	900 kHz for digital modulation types and 100 kHz for analog
SNR	30 dB
Sampling rate	200 kHz
Samples per input frame	1024
Symbols per frame	128
Samples per symbol	8
Max Doppler	5 Hz
Max clock offset	5 ppm
Multipath profile	Rician fading
Rician fading K factor	4
Rician fading delay profile	0, 1.8, 3.4 samples Delay
Rician fading path gains	0, -2, -10 dB

IV. METHODS

As a first step to any learning task it is a good idea to plot the input data to search for features that are striking. If there is a clear path forward after visualization, using a neural network may be overkill or even suboptimal. Figures 3,4 and 5 show the input data in the time domain, constellation domain and frequency domain respectively before any channel effects are added. Even before the addition of channel effects, it is not immediately clear looking at these plots how one might classify the modulation types. There is no clear set of rules that could be used to differentiate between all modulation types. We could work hard to find a set of rules on a set of transformed features, but this would be the classical approach to modulation classification and defeats the purpose of using a neural network.

Using PyTorch two different neural networks were tested; the basic convolutional neural network and the residual convolutional neural network. The input time domain samples are transformed accordingly to Figure 6 and fed to the neural networks. The dataset has a total of 110,000 signals, of which 70% are used for training, 20% are used for testing and 10% are used for validation. Each network is trained for 100 epochs on the training set with early stopping in place to avoid overfitting the model. The results are then calculated on the testing set. To evaluate the performance of each model we generate a confusion matrix, calculate the overall accuracy and look at the loss plots during training.

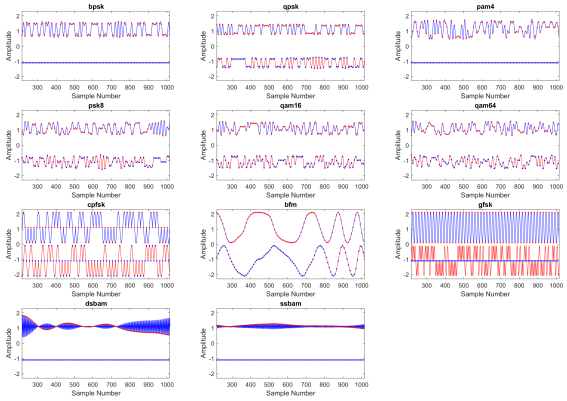

Fig. 3. Time domain input modulations plotted across sample number.

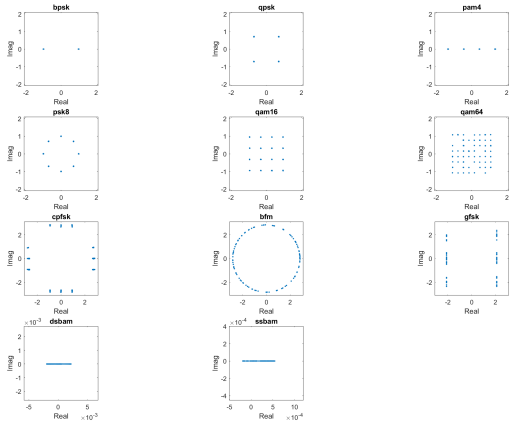

Fig. 4. Constellation diagrams for digital modulation types

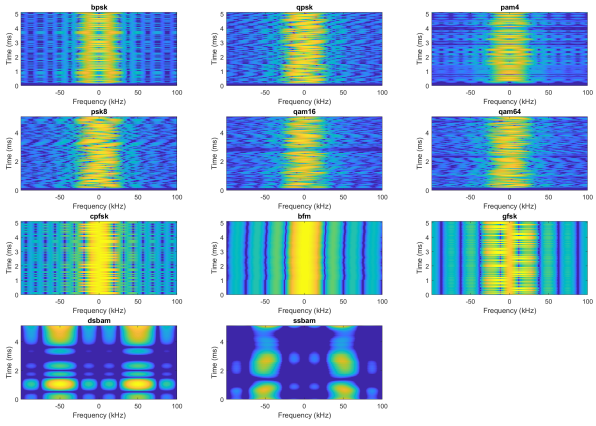

Fig. 5. Spectrograms for each of the modulation types. Time is given by the vertical axis with frequency along the horizontal axis

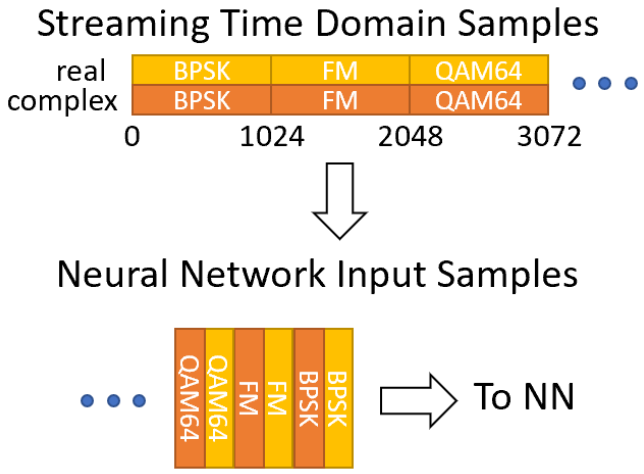

Fig. 6. Transformation process from streaming input samples to blocked NN input

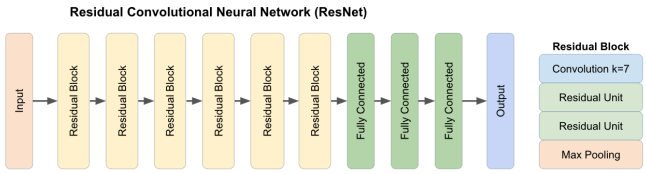

Fig. 7. Basic convolutional neural network architecture

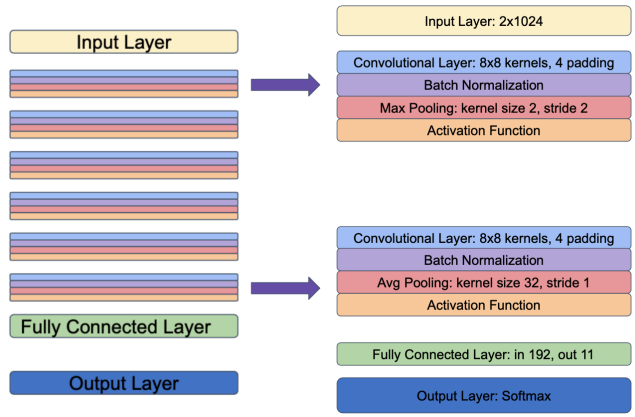

Fig. 8. Detailed convolutional neural network architecture

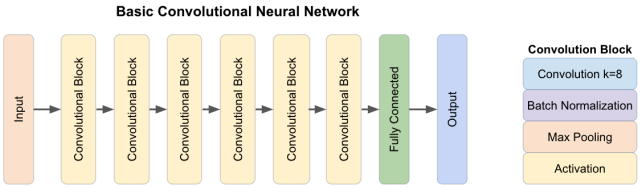

Fig. 9. Residual convolutional neural network architecture

A. Convolutional Neural Network

The basic convolutional neural network architecture used contains six convolutional blocks followed by a fully connected layer and a softmax. Each convolutional block contains a convolutional layer, a batch normalization, a max-pooling and an activation function in sequence. During the exploration, two activations have been used. One is ReLU, the most commonly used activation function, and the other one is tanh. RelU has been proven to outperform tanh in other applications. However, we wanted to insure that it was the best choice for modulation classification.

Figure 8 shows the detailed architecture of the CNN model. The first five Convolutional blocks are essentially the same with the only difference in the input and output dimensions. The last one block has an average pooling instead of max pooling. Compared with max pooling, average pooling smooths out the feature and emphasizes on down-sampling of the overall feature.

B. Residual Convolutional Neural Network

The residual convolutional neural network (ResNet) has proven to be a valuable tool for classification in many applications. It relies on the usage of skipped connections in the network to train the model to learn residual features that can be helpful for classification.

The ResNet architecture is shown in figure 9. It contains 6 residual blocks followed by three fully connected layers. The residual blocks are made up of one convolutional layer, two residual units and a max pooling layer. Within the residual units there is a skipped connection that goes over two convolutional neural networks. In total this network contains 30 convolutional layers which is five times the number of convolutional layers in the basic CNN architecture.

V. Tests, Results and Discussion

A. Convolutional Neural Network

Using ReLU as the activation function, a test accuracy of 82% is obtained. The loss plot in Figure 10 shows that the validation loss stays consistently close to the training loss. This indicates that the model does not overfit. After 70 epochs the training is ended due to early stopping to avoid overfitting. The confusion matrix in Figure 11 has a strong diagonal. So, for most classes the accuracy of classification is above 95%. However, some classes prove to be more difficult to differentiate than others. The modulation types 16QAM and 64QAM are the most frequently misclassified types. The classes QPSK and 8PSK are also difficult for the network to classify. This is due to the similarities between these types of modulations, which can be seen in the visualizations of the signals in Figures 3,4 and 5. These classes are persistently difficult to classify for all networks used in this investigation.

The performance of tanh was relatively worse than ReLu in the CNN model. Test accuracy for this method is 78%. The loss plot shown in figure 12 is slightly more noisy than the plot for relu and indicates that the training ended due to early stopping after 45 epochs. This could mean that with the tanh activation function the model is more prone to overfitting. The confusion matrix for the CNN model with tanh activation

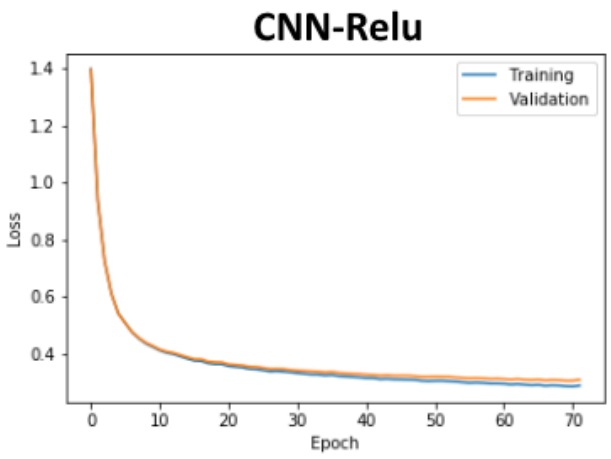

Fig. 10. CNN-Relu Training and Validation Loss

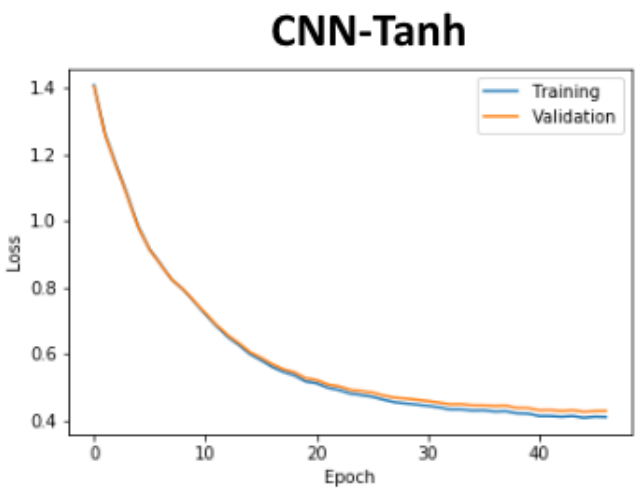

Fig. 12. CNN-Tanh Training and Validation Loss

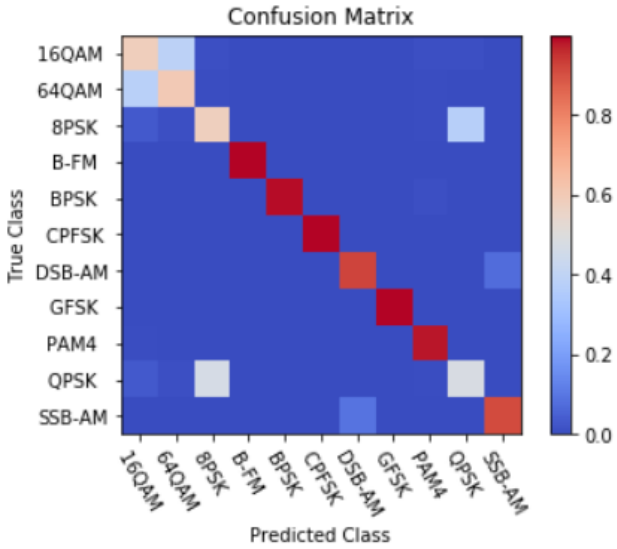

Fig. 11. CNN-ReLU Confusion Matrix

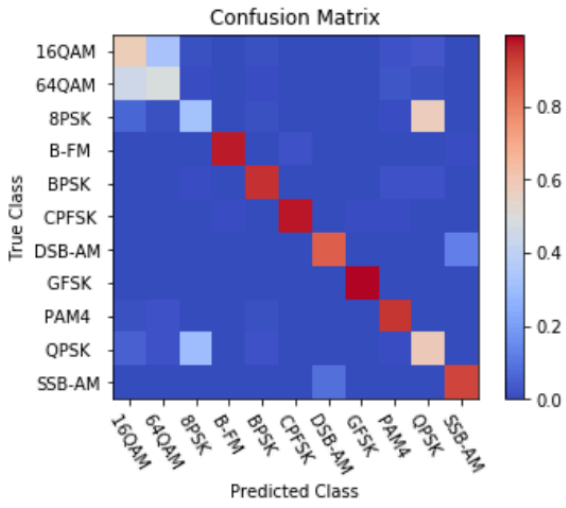

Fig. 13. CNN-Tanh Confusion Matrix

function is seen in figure 13. The diagonal accuracy is still relatively high but there are more misclassifications across all modulations types. This causes the confusion matrix to appear more fuzzy. Moving forward relu is used as activation in the ResNet model due to these findings.

B. Residual Convolutional Neural Network

The overall test accuracy obtained with the ResNet model is 81%. This is slightly less than the results for the basic CNN with ReLu. The loss plot for the ResNet model is shown in figure 14. The training proceeded for 100 epoch and did not stop from early stopping. The validation loss clearly diverges from the training loss, which is indicative of overfitting. The ResNet model has 5x the number of parameters than the basic CNN model. Models with more parameters require more data to properly train those parameters. For this reason larger models are prone to overfitting. Since the same amount of training data is used as with the smaller CNN network, it makes sense that overfitting is observed in this case.

The confusion matrix for the reset model is shown in figure 15. The prominent diagonal shows that the accuracy is relatively high for most classes, but the same difficulties are observed as with the basic CNN. The 16QAM and 64QAM modulation types are often confused, as are the QPSK and 8PSK modulation types. Overall the basic CNN has better performance, showing that sometimes simpler solutions are more effective especially when datasets are limited. Please note that we did attempt a larger dataset towards the end, however datahub would crash on loading dataset above 4GB.

C. Effect of frequency error on CNN

We used a nominal value of MaxClockcOffset and MaxDoppler in creating synthetic data as shown in Table I. Real life scenarios typically have higher values of frequency error.

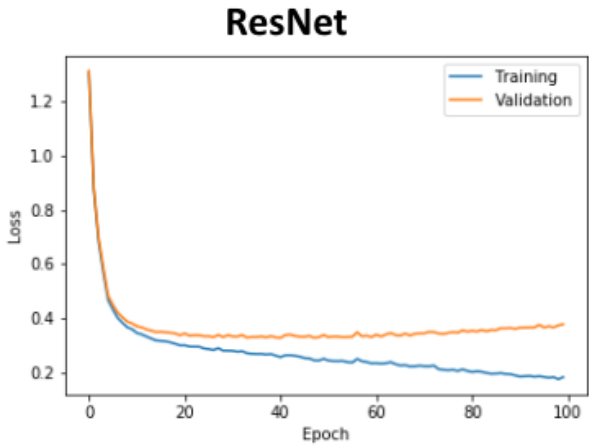

Fig. 14. ResNet Training and Validation Loss

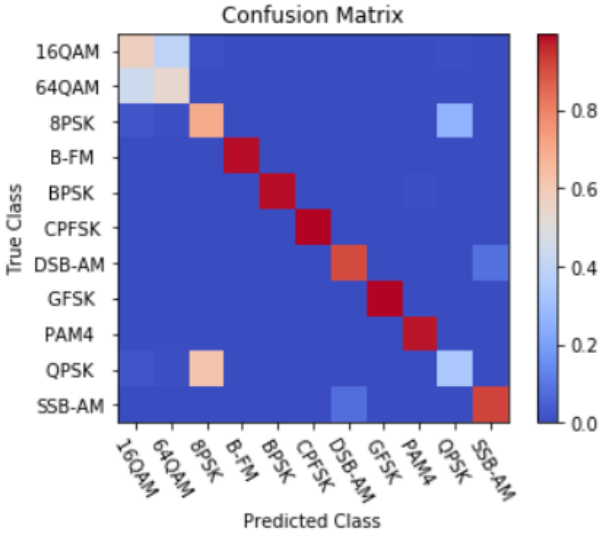

Fig. 15. ResNet Confusion Matrix

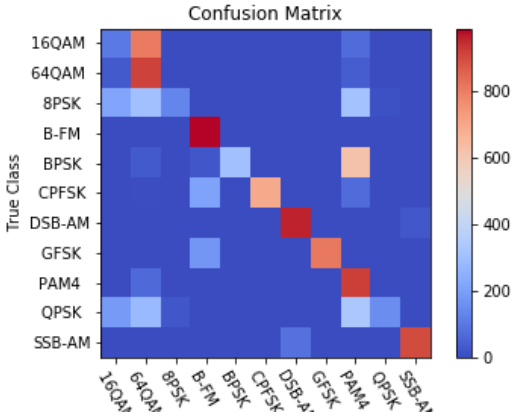

Fig. 16. Confusion matrix for data with 500Hz Frequency error.

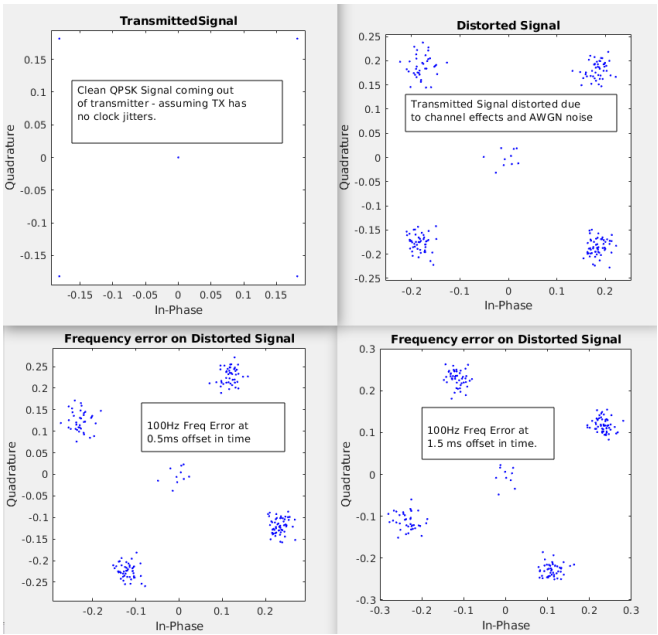

Fig. 17. Clean signal affected by channel followed by freqeuncy errors.

We chose milder conditions of frequency error for establishing a baseline performance of a model and test it on data of higher frequency error. In this test, I create a new test data set of 1000 samples each class with varying freqeuncy errors and test it on a pretrained CNN model. The results of the test can be seen in the confusion matrix in Figure 16. Notice that a lot of results are wrongly predicted as 64 QAM or 16QAM. To understand this, lets look at an intuition behind what a CNN learns.

One way of looking at what a CNN learns would be that it first makes a transformation of the IQ samples to a 2D polar co-ordinate system similar to Figure 1 and is classifying the frames in this transformed domain. Thus one would have 1024 points on a 2D plane each of them affected by non-linear multiplicative correlated noise effects. These channel effects can be thought of spreading the constellation points, that are initially in a regular grid as shown in Figure 1, into an irregular cluster of points. The frequency error additionally rotates the clusters continuously with respect to time. In equation 6 where $r(t)$ is the received signal and $r'(t)$ is the received signal with phase rotation due to frequency errors, notice that $r(t)$ is multiplied by a value with unit amplitude but a a phase that is constantly changing at a rate proportional to f_{err}. The phase is changing continuously since t is changing. In Figure 17 illustrates our explanation above in four sub plots. We are initially seeing the clean constellation points coming out of a transmitter, followed by channel distortion with minimal frequency errors. If we assume higher frequency errors on top of the distortion, what we see are the rotations of the cluster as shown in the bottom subplots of Figure 17. In a single frame of 5 ms, we will have points that have rotated upto 180 degrees. Thus when there is a high frequency error, what the CNN receives will be 1024 points where each sample is rotated

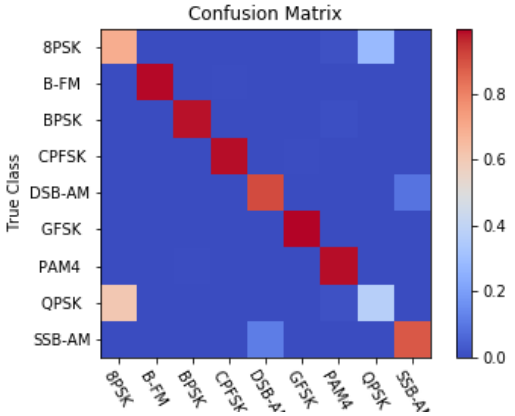

Fig. 18. Confusion matrix for classes without QAM modulation types.

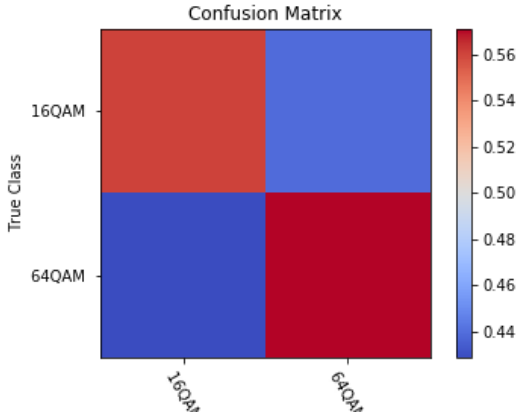

Fig. 19. Confusion matrix for classes with QAM modulation types.

slightly more than the previous sample. Thus it will be a dense cloud spread throughout the constellation diagram. This will mostly resemble the densely packed grid of modulation types such as 16QAM and 64QAM. Thus in cases of higher frequency errors, it makes sense that phase modulation types are most affected and also that a lot of them are wrongly classified as QAM types.

$$r'(t) = r(t)e^{j\theta_{err}}, \theta_{err} = 2\pi f_{err} t, \tag{6}$$

D. Performance of CNN using a subset of classes

When we trained and tested the CNN, from the results in Figure 11 we noticed that the top two mis-classified classes were 16 QAM and 64 QAM. We wanted to study whether the neural network performance can improve if we separately train and test them on non-QAM classes and QAM only classes. The reason was that we wanted to understand if learning other modulation types is somehow confusing our model and making it miscasslify QAM and vice versa.

1) No QAM modulation - CNN performance: The overall test accuracy for a CNN trained on 11 versus 9 classes improved from 82% to 86%. Notice the confusion matrix in Figure 18. The confusion matrix for the 9 classes looks as though it is taken out of the larger confusion matrix for 11 classes as shown in Figure 11, thus clearly indicating that the CNN classification of non-QAM types is not much affected by QAM types.

2) Only 16 and 64QAM - CNN performance: The overall test accuracy for a CNN trained and tested on only 16QAM and 64QAM is 56%. Notice the confusion matrix in Figure 19. The 2by2 confusion matrix looks as though it is carved out of the larger 11×11 confusion matrix in Figure 11. Thus we can conclude by saying that the way CNN learns to classify QAM modulation types is delineated from the non-QAM modulation types, under low frequency error conditions.

E. Effect of frame misalignment on CNN

In real life scenarios say particular bandwidth has QPSK modulation data, it can suddenly change to 16QAM. We will not know where the boundary in time is, for this transition. It is therefore an interesting problem to study how CNN behaves when we supply it with a signal of a particular modulation type for one part of the frame and another modulation type for the rest. In this test, we pick a pair of modulation types say for example BPSK and GFSK and we generate a frame that contains say 5% of BPSK and 95% of GFSK i.e. first 52 samples are BPSK and next 972 samples are GFSK. See Figure 20 for an illustration of how a misaligned frame is constructed. We slide this percentage from 5% to 95% in steps of 5. For each of the frame thus calculated, we test the data on pretrained CNN. We calculate the normalized scores and make a note of the scores belonging to the two classes in focus. For each pair we thus get 19 frames of data and thus 19 readings of normalized scores per class. We also add the normalized scores for the two class and get a sum of scores number. We thus have three set of readings one for each class and one for the sum of scores, all of which we plot against percentage contribution from the first class. We repeat this test for analog modulation types FM and SSB-AM, phase modulation types QPSK and 8PSK, QAM modulation types 16QAM and 64QAM. This test was run on MATLAB using a pretrained CNN network loaded from MATLAB [20], that is supposed to almost the same as our model. We were running into issues when we tried testing it in python and due to shortage of time, we decided to take this approach.

Please note that the normalized scores can also be thought of percentage of contribution of each class to a frame. The expectation of the test results is that, if a signal contains 40% BPSK and 60% GFSK we should get a score of 0.4 for BPSK and 0.6 for GFSK. Thus a plot should ideally have two straight lines, crossing each other similar to the two diagonals of a rectangle. Also note that the plots are for one data frame picked from each class. That particular data frame picked might not be truly a global representation of the class data and therefore the results should be looked at accordingly. Nevertheless, the results are interesting and will definitely add a good deal to our understanding of the workings of the CNN.

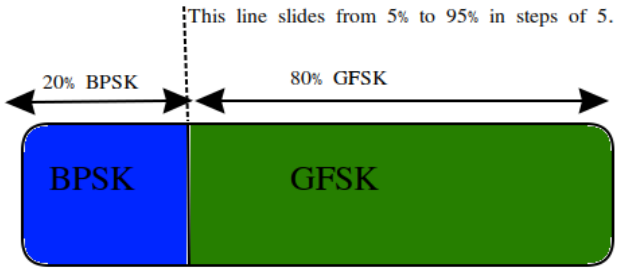

Fig. 20. A single frame with first 20% BPSK data and next 80% GFSK data

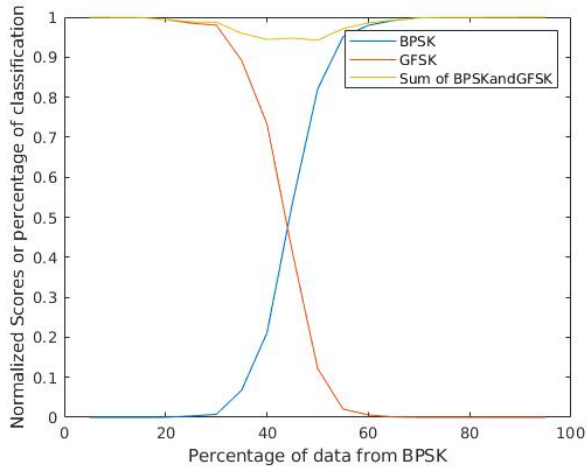

Fig. 21. Plot of normalized scores for a frame containing BPSK and GFSK

A future work item would be to have a plot averaged over all data. Also whenever I mention the word contribution in the sections below, it represents the percentage of the frame a particular class occupies.

1) BPSK - GFSK: In Figure 21, the score is predominantly close to zero for BPSK for any contribution up to 30% and the same is true for the score of GFSK when its contributions are weak and less than 30%. Beyond 30% contribution points, the two lines start ramping up steeply and completely dominate the score when they are close to having 60% contribution. Thus we can say that the CNN is able to very confidently classify a signal with a score of almost 1, as long as the contribution is more than 60%. Also note that the CNN gets confused around 50% region when the total sum is less than 1, which means that the signal is thought of as partly belonging to some other class other than the two of interest.

2) FM - AM: In Figure 22, we see results similar to the case of BPSK-GFSK. However, FM seems to moderately dominate SSB-AM, meaning the CNN more often thinks the signal is FM than AM. Also most interestingly, the sum of scores dips quite a bit to 0.6 at around 35% region of contribution from AM, thus indicating that the frame is thought of belonging to some other class.

3) QPSK - 8PSK: In Figure 23, we see that QPSK dominates 8PSK. Please note that this result is counter-intuitive

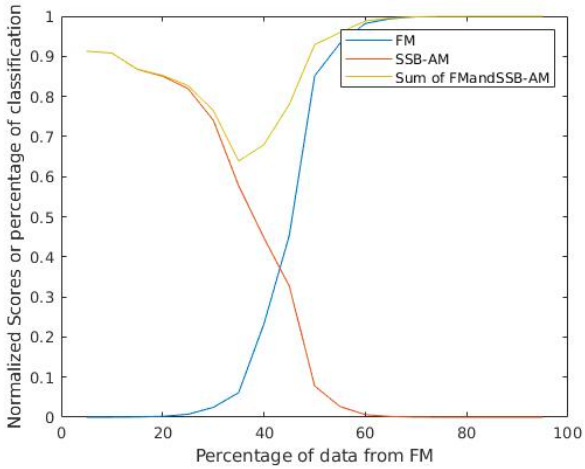

Fig. 22. Plot of normalized scores for a frame containing FM and SSB-AM

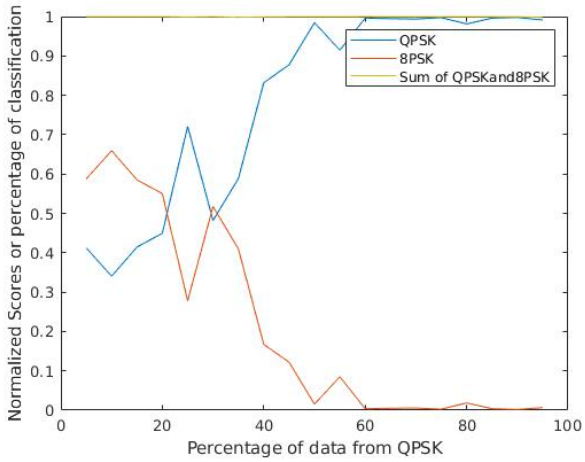

Fig. 23. Plot of normalized scores for a frame containing QPSK and 8-PSK

since QPSK can be thought of as a subset of 8PSK. Thinking on the same line of argument posed in the subsection titled Effect of frequency error on CNN, one would be tempted to think that 8PSK would dominate QPSK. One explanation could be that QPSK data picked had stronger channel artifacts and the 8PSK data was relatively cleaner. The results from the confusion matrix in Figure 18 also indicate that more often than not, QPSK is wrongly classified as 8PSK and thus this result should be an anomaly.

4) 16QAM - 64QAM: In Figure 24, we see that 64QAM dominates 16QAM. This is consistent with our expectations as detailed previously in our our analysis of QPSK and 8PSK.

VI. Conclusion

It has been shown that the performance of neural networks for the modulation classification task can achieve very high levels of accuracy over a large range of modulation types under nominal channel distortions. The convolutional neural

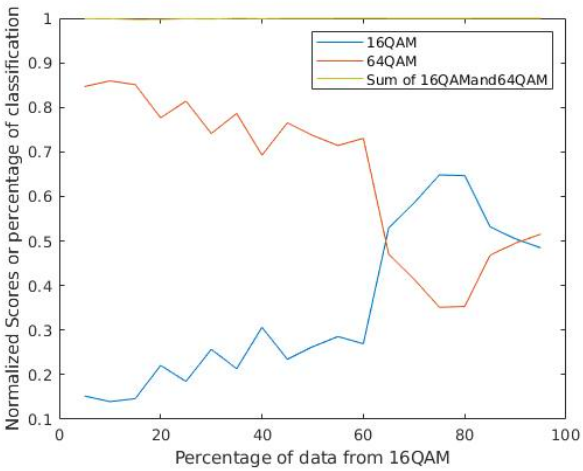

Fig. 24. Plot of normalized scores for a frame containing 16QAM and 64-QAM

network and residual neural network achieved accuracies of over 80%. Further we were able to use our physics knowledge of wireless communications to pry open the black box of CNN and infer the following. In the frequency error tests, we noticed that phase sensitive modulation types are most vulnerable to frequency errors and we were also able to provide a strong reasoning on why lots of non-QAM modulation types are misclassified as QAM types. From our frame boundary misalignment tests, we understood that similar modulation types are more prone to being misclassified as one another in general. Further, in the tests done using a subset of classes, we learnt that the way CNN learns to classify QAM modulation types is delineated from the non-QAM modulation types, under low frequency error conditions.

VII. Contributions

Venkatesh Sathyanarayanan: Co-ordination among team members, Testing of frequency error effects on CNN, Frame misalignment test and subset of classes test

Jahya Burke: implemented the basic CNN network model in pytorch with Rui. Implemented the ResNet model. Created and ran training and testing code for CNN with tanh/relu and resnet.

Rui Shang: Implemented the basic CNN network model in pytorch with Jahya. Implemented LSTM but not get it working yet.

Richard Bell: Generated test and training data. Tested frame misalignment performance using our NN implementation. Searched for and reviewed most references for final paper.

References

[1] W. Peterson, T. Birdsall, and W. Fox, "The theory of signal detectability," *Transactions of the IRE Professional Group on Information Theory*, vol. 4, no. 4, pp. 171–212, Sep. 1954.

[2] R. C. Davis, "The detectability of random signals in the presence of noise," *Transactions of the IRE Professional Group on Information Theory*, vol. 3, no. 3, pp. 52–62, March 1954.

[3] S.-Z. Hsue, "Automatic modulation classification using zero crossing," *IEE Proceedings F (Radar and Signal Processing)*, vol. 137, pp. 459–464(5), December 1990. [Online]. Available: https://digital-library.theiet.org/content/journals/10.1049/ip-f-2.1990.0066

[4] K. Kim and A. Polydoros, "Digital modulation classification: the bpsk versus qpsk case," in *MILCOM 88, 21st Century Military Communications - What's Possible?'. Conference record. Military Communications Conference*, Oct 1988, pp. 431–436 vol.2.

[5] J. E. Hipp, "Modulation classification based on statistical moments," in *MILCOM 1986 - IEEE Military Communications Conference: Communications-Computers: Teamed for the 90's*, vol. 2, Oct 1986, pp. 20.2.1–20.2.6.

[6] J. E. Whelchel, D. L. McNeill, R. D. Hughes, and M. M. Loos, "Signal understanding: an artificial intelligence approach to modulation classification," in *[Proceedings 1989] IEEE International Workshop on Tools for Artificial Intelligence*, Oct 1989, pp. 231–236.

[7] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. Savannah, GA: USENIX Association, 2016, pp. 265–283. [Online]. Available: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi

[8] A. Paszke, S. Gross, S. Chintala, and G. Chanan, "Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration," *PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration*, vol. 6, 2017.

[9] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[10] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[11] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[12] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, Y. W. Teh and M. Titterington, Eds., vol. 9. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256. [Online]. Available: http://proceedings.mlr.press/v9/glorot10a.html

[13] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *CoRR*, vol. abs/1505.00853, 2015. [Online]. Available: http://arxiv.org/abs/1505.00853

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

[15] B. Kim, J. Kim, H. Chae, D. Yoon, and J. W. Choi, "Deep neural network-based automatic modulation classification technique," in *2016 International Conference on Information and Communication Technology Convergence (ICTC)*, Oct 2016, pp. 579–582.

[16] T. J. O'Shea and J. Hoydis, "An introduction to machine learning communications systems," *CoRR*, vol. abs/1702.00832, 2017. [Online]. Available: http://arxiv.org/abs/1702.00832

[17] U. Satija, M. S. Manikandan, and B. Ramkumar, "Performance study of cyclostationary based digital modulation classification schemes," in *2014 9th International Conference on Industrial and Information Systems (ICIIS)*, Dec 2014, pp. 1–5.

[18] K. Kim, I. A. Akbar, K. K. Bae, J. sun Urn, C. M. Spooner, and J. H. Reed, "Cyclostationary approaches to signal detection and classification in cognitive radio," in *Proceedings of the 2007 2Nd IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 212–215. [Online]. Available: http://dx.doi.org/10.1109/DYSPAN.2007.35

[19] U. Satija, M. Mohanty, and B. Ramkumar, "Cyclostationary features based modulation classification in presence of non gaussian noise using sparse signal decomposition," *Wireless Personal Communications*, vol. 96, no. 4, pp. 5723–5741, Oct 2017. [Online]. Available: https://doi.org/10.1007/s11277-017-4444-4

[20] MATHWORKS, *Modulation Classification with Deep Learning*, 2019 (accessed June 12, 2019). [Online]. Available: https://www.mathworks.com/help/deeplearning/examples/modulation-classification-with-deep-learning.html

[21] T. J. OShea, T. Roy, and T. C. Clancy, "Over-the-air deep learning based radio signal classification," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 168–179, Feb 2018.

[22] O. A. Dobre, A. Abdi, Y. Bar-Ness, and W. Su, "Survey of automatic modulation classification techniques: classical approaches and new trends," *IET Communications*, vol. 1, no. 2, pp. 137–156, April 2007.

[23] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[24] DeepSigIO, *Modulation Classification with Deep Learning*, 2019 (accessed June 13, 2019). [Online]. Available: https://www.deepsig.io/datasets