Implementation: End to End Deep Learning Architectures for Automatic Modulation Recognition

Timothy Do, *Student Member, IEEE*

***Abstract*—Automatic Modulation Recognition (AMR) is the process of blindly identifying the type of modulation scheme of a transmitted signal from the receiver end. This work employs the task of AMR by developing convolutional neural networks (CNNs) for the RadioML 2016.10A, RadioML 2018.01A, and HisarMod 2019.1 datasets. Data augmentation by temporal reversal provides an improvement for all AMR architectures by making the models robust in I/Q sample order. Across all models, our best (OSheaCNN2018) model with data augmentation achieves a classification accuracy at 20 dB SNR of 93.2%.**

***Index Terms*—Automatic Modulation Recognition, Convolutional Neural Networks, Data Augmentation**

I. Introduction

WIRELESS communication signals go through an end-to-end pipeline for a receiver to reliably receive data from a transmitter. The high-level processes from the input data are channel coding, modulating at some carrier frequency, pulse shaping, transmitting through a channel, then demodulating and decoding the output data. Mathematically, it is abstractly modeled as

$$y(t) = x(t) * h(t) + n(t) \tag{1}$$

where the complex baseband received signal y is the complex baseband pulse-shaped input signal x convolved with the baseband complex channel impulse response h in addition to Additive White Gaussian Noise (AWGN) n. Features of wireless signals (e.g., modulation type, channel path loss/fading realizations, channel codes) can be very insightful to know at the receiver to effectively decode the data. The problem lies when these features aren't known at the receiver, thus making reliable estimation and detection methods necessary for learning them.

Automatic Modulation Recognition (AMR) is the process of blindly identifying the type of modulation scheme of a transmitted signal from the receiver end. Its applications range from dynamic spectrum access in cognitive radio to intercepting malicious communications in the military [1]. A high-level block diagram in Figure 1 incorporates AMR as a typical decision block for the demodulator to correctly recover the transmitted signal.

This implementation project employs the task of AMR by classifying with a variety of deep neural network models from end to end. In the field of computer vision, neural network architectures such as CNN (Convolutional Neural Network) from AlexNet [2] yield high classification accuracy in image classification tasks and time-efficiency for training with GPU (graphics processing unit) accelerators. The input to the neural networks are raw I/Q samples, where for the received signal y can be decomposed as the sample of its in-phase and quadrature components:

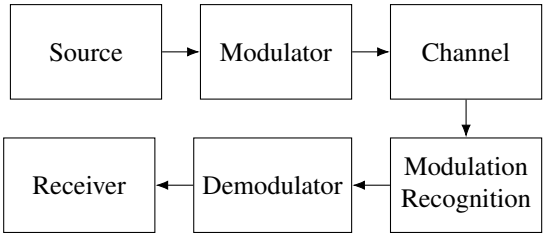

Fig. 1. Typical Wireless Communications Block Diagram with AMR

$$y_i(t) = \Re\{y(t)\} \tag{2}$$

$$y_q(t) = \Im\{y(t)\} \tag{3}$$

This project explores classification of various datasets that have samples of a signal's I/Q components in addition to labels for their modulation class. Additionally, this project explores data augmentation as a means of improving classification accuracy, shown effective in [3] for the analogous task of image classification.

II. Literature Review

A. Radio Deep Learning Architectures

Foundational research efforts have shown the effectiveness of CNNs [4] in the task of modulation recognition. Using the RadioML v2016.10a dataset, the authors achieved an overall 87.4% accuracy rate across all SNRs. A follow-up work by the same group used a hybrid architecture known as CLDNN (Convolutional Long Term Short Memory) [5]. It is composed of a 3-layered CNN with concatenation of its first layer fed as the input to the LSTM (Long Term Short Memory) which outperforms the previous CNN. Finally, they trained and tested the architectures on real over-the air data (i.e., RadioML v2018.01a) and with 10 dB SNR, the overall classification rate for all modulation schemes was 95.6% [6].

Another work investigates the performance of LSTM (Long Term Short Memory) models when performing the task of AMR against CNNs [7]. Using the RadioML v2016.10a dataset, they found that in high (10 dB) SNR regimes the LSTM outperformed the CNN with a classification accuracy of 94%, but falls short on low (-2 dB) SNR regimes.

In the industry, researchers at BAE Systems proposed a hierarchical neural network (HNN) with frequency feature extraction [8]. It utilizes multiple classifiers in series to first segment the modulation into two broad classes (i.e., Analog/Digital), then based on the first classifier utilizes another classifier for more specific types of modulations. With the RadioML v2016.10a dataset, they assessed that the overall model performs well over high (18 dB) SNRs at over 90%. However, they couldn't assess the sub-classifier of modulation order effectively because of the lack of training samples, partitioned only from m-order modulations.

B. Modulation Datasets

The RadioML datasets are a repository for generating synthetic data for various radio machine learning tasks [9]. It was created using GNURadio [10], a program with various models for modulators, channel coders, demodulators, and dynamic channel impairments like fading. Random channel impairments with SNRs ranged from -20 to 20 dB are applied to each vector. For RadioML 2016.10A, there are 11 different modulation classes, which include a mixture of digital and analog modulations. Each I/Q sequence is stored as 128 complex samples (i.e., 2-Wide Columns) with a modulation class and SNR label for each sequence. There are a total of 220,000 sequences in the dataset. A most recent version, RadioML 2018.01A, contains both synthetic and real over-the air data with 24 different modulation classes as 1024 complex samples for each I/Q sequence [6]. There are a total of 2,555,904 sequences in the dataset.

The HisarMod 2019.1 dataset is a repository that aims at being more applicable in real-world scenarios than the RadioML dataset [11]. It was created using MATLAB v2017a with the Communications Toolbox with SNRs ranging from -20 dB to 18 dB aligning with the RadioML dataset. There is also a uniform distribution of channel fading realizations between Rayleigh, Rician (k=3), and Nakagami models applied throughout the dataset. This is a distinction from the over-the-air data in RadioML 2018.01A set, which only has a constant environment for fading. It has 26 different modulation classes, each with I/Q sequences of 1024 complex samples with a modulation class and SNR label for each sequence. There are a total of 780,000 sequences in the dataset.

III. Methodology

A. Dependencies

The datasets explored in the project would be RadioML 2016.10A, RadioML 2018.01A, and HisarMod 2019.1. The codebase for this project is implemented in Python 3.11 and MATLAB R2023B. The project is hosted as a GitHub repository for version control [12].

In terms of hardware requirements, A PC desktop with adequate amounts of RAM ($\geq$ 64 GB), a multicore processor, and an NVIDIA RTX GPU with adequate VRAM ($\geq$ 8 GB) are recommended for fully loading in the datasets and efficient model training. For context, a PC desktop with an Intel Core i7-11700K processor, 128 GB of RAM, and an NVIDIA RTX 3090 GPU was used for this project's development.

B. Model Design

All the models were implemented in PyTorch to leverage the use of GPU accelerators during forward/backward propagation of the model [13]. They are subclasses of the *torch.nn.Module* superclass. Architecture layers are listed in Figure 2–4.

Layer (type:depth-idx)	Output Shape	Param #
├─Conv1d: 1-1	[-1, 64, 128]	448
├─ReLU: 1-2	[-1, 64, 128]	--
├─Dropout: 1-3	[-1, 64, 128]	--
├─Conv1d: 1-4	[-1, 16, 128]	3,088
├─ReLU: 1-5	[-1, 16, 128]	--
├─Dropout: 1-6	[-1, 16, 128]	--
├─Linear: 1-7	[-1, 128]	262,272
├─ReLU: 1-8	[-1, 128]	--
├─Dropout: 1-9	[-1, 128]	--
├─Linear: 1-10	[-1, 11]	1,419
├─Softmax: 1-11	[-1, 11]	--

Fig. 2. OsheaCNN2016 Architecture adopted from [4]

.

Layer (type:depth-idx)	Output Shape	Param #
├─Conv1d: 1-1	[-1, 64, 1024]	448
├─MaxPool1d: 1-2	[-1, 64, 512]	--
├─SELU: 1-3	[-1, 64, 512]	--
├─Conv1d: 1-4	[-1, 64, 512]	12,352
├─MaxPool1d: 1-5	[-1, 64, 256]	--
├─SELU: 1-6	[-1, 64, 256]	--
├─Conv1d: 1-7	[-1, 64, 256]	12,352
├─MaxPool1d: 1-8	[-1, 64, 128]	--
├─SELU: 1-9	[-1, 64, 128]	--
├─Conv1d: 1-10	[-1, 64, 128]	12,352
├─MaxPool1d: 1-11	[-1, 64, 64]	--
├─SELU: 1-12	[-1, 64, 64]	--
├─Conv1d: 1-13	[-1, 64, 64]	12,352
├─MaxPool1d: 1-14	[-1, 64, 32]	--
├─SELU: 1-15	[-1, 64, 32]	--
├─Conv1d: 1-16	[-1, 64, 32]	12,352
├─MaxPool1d: 1-17	[-1, 64, 16]	--
├─SELU: 1-18	[-1, 64, 16]	--
├─Conv1d: 1-19	[-1, 64, 16]	12,352
├─MaxPool1d: 1-20	[-1, 64, 8]	--
├─SELU: 1-21	[-1, 64, 8]	--
├─Linear: 1-22	[-1, 128]	65,664
├─SELU: 1-23	[-1, 128]	--
├─Linear: 1-24	[-1, 128]	16,512
├─SELU: 1-25	[-1, 128]	--
├─Linear: 1-26	[-1, 24]	3,096
├─Softmax: 1-27	[-1, 24]	--

Fig. 3. OsheaCNN2018 Architecture adopted from [6]

.

Layer (type:depth-idx)	Output Shape	Param #
├─Conv1d: 1-1	[-1, 64, 1024]	448
├─MaxPool1d: 1-2	[-1, 64, 512]	--
├─SELU: 1-3	[-1, 64, 512]	--
├─Conv1d: 1-4	[-1, 64, 512]	12,352
├─MaxPool1d: 1-5	[-1, 64, 256]	--
├─SELU: 1-6	[-1, 64, 256]	--
├─Conv1d: 1-7	[-1, 64, 256]	12,352
├─MaxPool1d: 1-8	[-1, 64, 128]	--
├─SELU: 1-9	[-1, 64, 128]	--
├─Conv1d: 1-10	[-1, 64, 128]	12,352
├─MaxPool1d: 1-11	[-1, 64, 64]	--
├─SELU: 1-12	[-1, 64, 64]	--
├─Conv1d: 1-13	[-1, 64, 64]	12,352
├─MaxPool1d: 1-14	[-1, 64, 32]	--
├─SELU: 1-15	[-1, 64, 32]	--
├─Conv1d: 1-16	[-1, 64, 32]	12,352
├─MaxPool1d: 1-17	[-1, 64, 16]	--
├─SELU: 1-18	[-1, 64, 16]	--
├─Conv1d: 1-19	[-1, 64, 16]	12,352
├─MaxPool1d: 1-20	[-1, 64, 8]	--
├─SELU: 1-21	[-1, 64, 8]	--
├─Linear: 1-22	[-1, 128]	65,664
├─SELU: 1-23	[-1, 128]	--
├─Linear: 1-24	[-1, 128]	16,512
├─SELU: 1-25	[-1, 128]	--
├─Linear: 1-26	[-1, 26]	3,354
├─Softmax: 1-27	[-1, 26]	--

Fig. 4. HisarCNN2019 Architecture adopted from [6]

.

OSheaCNN2016, OSheaCNN2018, and HisarCNN2019, are the AMR model architectures used to classify samples in the RadioML 2016.10A, RadioML 2018.01A, and HisarMod 2019.1 datasets respectively. All models are purely CNNs based on the papers where each dataset was introduced, aside from HisarMod. This is because the original architecture requires an external SNR parameter [11], not typically known for an end to end pipeline in the context of this project. Thus, the HisarCNN2019 model is designed to be very similar to the OSheaCNN2018 model aside from the number of class probabilities at the output layer (i.e., from 24 to 26 classes).

In all the AMR models, class probabilities are outputted through the *Softmax* function [14]. Given raw logits at the neural network's output, $\mathbf{z}$, the estimated one-hot encoded vectors $\hat{\mathbf{y}}$ representing N class probabilities is defined [1] as

$$\hat{y}_n = softmax(\mathbf{z})_n = \frac{exp(z_n)}{\sum_{j=0}^{N-1} exp(z_j)}, n = 0, ..., N-1 \quad (4)$$

C. Model Training & Evaluation

The RadioML datasets are split 90-5-5% for training/validation/test subsets, and the HisarMod train subset is split 80-20% for training/validation. The training set is used to learn the model weights, the validation set is used for fine-tuning model hyperparameters, and the test set is used for evaluating classification accuracy. For training each AMR model, the criterion to minimize during backpropagation is the cross-entropy loss. For a one-hot encoded label $\mathbf{y}$ and its model estimate, $\hat{\mathbf{y}}$, the cross-entropy loss is defined as

$$\mathcal{L} = -\sum_{i=0}^{N-1} y_i log(\hat{y}_i) \quad (5)$$

The Adam (Adaptive Moment) optimizer was used for its use of both first and second adaptive moments during model parameter updates [15]. Weight decay is also employed for learning rate annealing to mitigate overfitting. Additionally, early stopping is implemented to choose the model with the lowest validation loss rather than the last epoch, mitigating overfitting [14]. Batch sizes are chosen for computing regularizing minibatch gradients, avoiding shallow local minima in the loss curvature. The optimal training hyperparameters were searched through manual grid search. Table I lists the hyperparameters for each model.

TABLE I
OPTIMAL TRAINING HYPERPARAMETERS FOR THE AMR MODELS.

Model	Epochs	Learning Rate	Weight Decay	Batch Size
OSheaCNN2016	250	1.00E-04	1.00E-05	1024
OSheaCNN2018	100	1.00E-03	1.00E-04	9000
HisarCNN2019	100	1.00E-03	1.00E-04	9000

For evaluating each AMR model, an estimated modulated class label $\hat{i}$ is generated by selecting the maximal element in the estimated one-hot vector. This indicates the highest class probability for the particular I/Q sequence, i.e.,

$$\hat{i} = \underset{i=0,...,N-1}{\operatorname{argmax}} \hat{y}_i \quad (6)$$

The estimated labels for each input are compared with the actual labels for computing model accuracy. The model accuracy is defined as the number of labels that match the correct labels over the total number of evaluated samples.

D. Dataset Preparation

To preprocess RadioML 2016.10A for the classification models, it first has to be unpickled from its pickle file with the Python *pickle* library using *latin1* encoding. The input data came in batches of 1000 I/Q sequences, each shaped (1000,2,128). All batches were combined to a single variable for a shape of (220000,2,128), stored as X. It's already dimensionally compatible with the CNN model (OSheaCNN2016). Each batch came with a unique label tuple, specifying the combination modulation class and SNR (e.g. (BPSK, 20)). The modulation class label must be converted into one-hot encoded vectors to fit into the cross entropy loss (Eq. 5) of the model. To do so, each modulation class is assigned a unique integer label and then converted into a one-hot encoded vector by doing the following:

For a dataset with N classes, each I/Q sequence has a corresponding integer label i in the range 0 - $N-1$. The corresponding N-length one-hot encoded vector $\mathbf{y}$ can be defined by its elements [14] as

$$y_n = \left\{ \begin{array}{ll} 1, & n = i \\ 0, & n \neq i \end{array} \right\}, n = 0, 1, ..., N-1 \quad (7)$$

Doing this conversion and combining all one-hot encoded labels results in a label of shape (220000,11), stored as Y.

To preprocess the RadioML 2018.01A dataset, its HDF5 file is read using the Python library *h5py*, yielding three variables: X, Y, and Z. X is the input I/Q data as an entire variable, shaped (2555904,1024,2). The latter two input dimensions are swapped to be dimensionally compatible with the CNN model (OSheaCNN2018). Y are the one-hot encoded vectors for the modulation classes with shape (2555904,24). Z is the SNRs of each of the samples with shape (2555904,1).

The HisarMod 2019.1 dataset already comes split into training (520,000) and testing (260,000) samples, with data, labels, and SNRs stored as CSVs. However, the data CSV files for each set isn't easily readable by Python by default, since each cell in the CSV file is a complex value string (i.e., *a+bi*). Thus, MATLAB was used to split the I/Q components for each subset and then recombined as a matrix (like RadioML) and stored as a Version 7.3 MAT file to store variables greater than 2 GB. The MAT files are then read into Python using the *mat73* library, resulting in training/testing data (X_train/X_test) shapes[2] of (520000,1024,2) and (260000,1024,2) respectively. The labels are predefined integer labels dependent on their type [16], thus they first must be mapped into the range 0-25 (N = 26), then converted into one-hot encoded vectors (each shaped (1,26)) as described in Eq. 7, stored as Y_train/Y_test.

[1]To align with the integer categorical labels used throughout the project, indices notationally start at 0 for one-hot encoded vectors.

[2]Like RadioML 2018.01A, the latter two axes should later be swapped to conform to its model's input (HisarCNN2019).

E. Dataset Augmentation

For data augmentation, an index reversal on each I/Q sequence is applied (e.g., by horizontal flip), since index reordering doesn't affect whether the I/Q signal is part of the same symbol set for modulation [17]. Overfitting the model is mitigated because of this invariance. Note that data augmentation is only applied to the training set; the validation and test sets remain the same. The labels of the augmented set are the same as those of the original training set. After applying the augmentation, the training set increases by a factor of 2, leading model training to be more data driven for improving classification accuracy.

Figures 5–7 indicate the loss trajectories of each of the AMR models after the training has been complete. The plots on the left of each figure are trained models without augmentation, and on the right with augmentation. For all the AMR models, there is no strong sign of overfitting, since the validation loss decreases linearly along with the training loss. This seems to be independent on whether the training set has been augmented or not. An interesting observation is the augmented loss trajectory for the HisarCNN2019 model has training/validation loss increasing, getting slightly overfit.

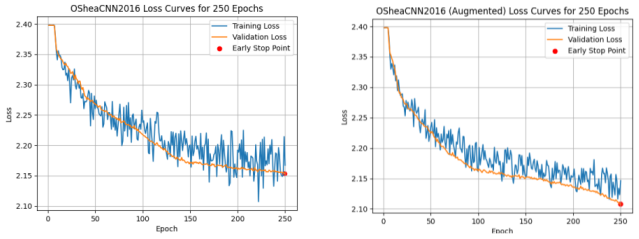

Fig. 5. OSheaCNN2016 Loss Trajectories.

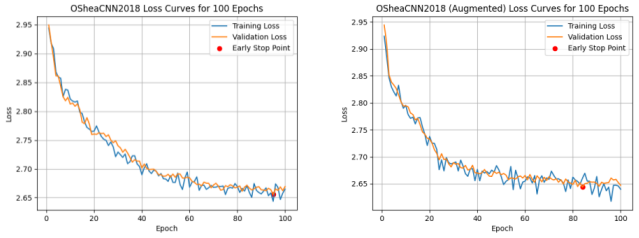

Fig. 6. OSheaCNN2016 Loss Trajectories.

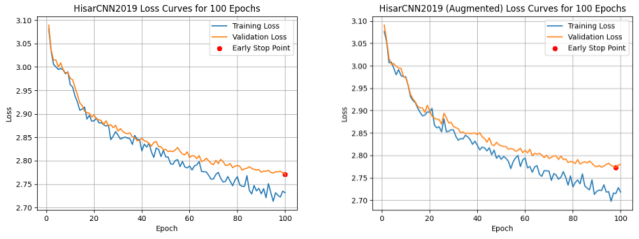

Fig. 7. HisarCNN2019 Loss Trajectories.

IV. Results

Each of the models were trained successfully with the given hyperparameter configuration in Table I using PyTorch. Plotted below for each of the datasets (Figures 9–11) is classification accuracy as a function of SNR threshold. The test data is filtered to an evaluation set where all elements are above a certain SNR threshold, and then model accuracy is calculated. An expected trend in all of these plots is that classification accuracy increases along SNR. A modulation's I/Q diagram[3] becomes more visually distinguishable as its SNR increases (see Figure 8 for an example in RadioML 2018.01A).

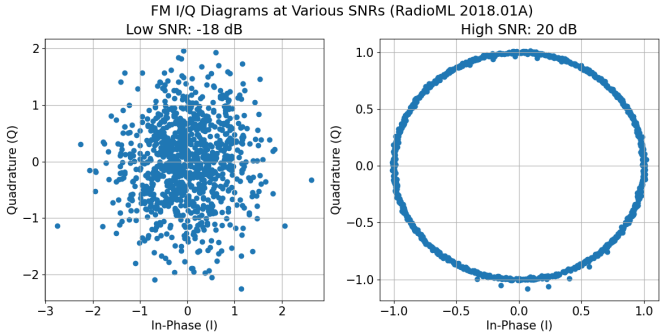

Fig. 8. I/Q Diagrams for a sample FM Modulation sequence at Low/High SNRs (-18/20 dB) in RadioML 2018.01A.

A. RadioML 2016.10A

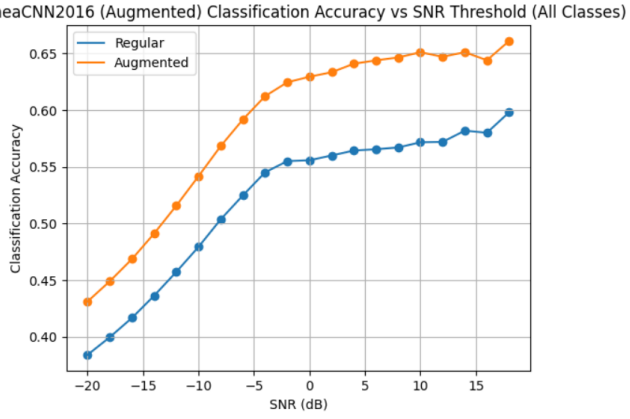

Fig. 9. OSheaCNN2016 Classification Accuracy vs. SNR Threshold

The OSheaCNN2016 models achieved an overall classification accuracy of 38.1/43.4% across all SNRs without/with data augmentation. At the highest SNR (i.e., 18 dB), mitigating the effect of noise, OSheaCNN2016 achieves an accuracy of 59.8/66.1%. In Figure 9, applying data augmentation provides a 6% uplift in classification accuracy across all SNRs. The RadioML 2016.10A dataset has a relatively low number of I/Q sequences compared to RadioML 2018.01A/HisarMod 2019.1 datasets, with each sequence having an eight of I/Q samples (i.e., 128 vs 1024). Thus, applying data augmentation mitigates temporal dependencies, especially since RadioML 2016.10A has few samples per sequence.

[3]For digital modulations, the I/Q diagram is referred to as its constellation.

B. RadioML 2018.01A

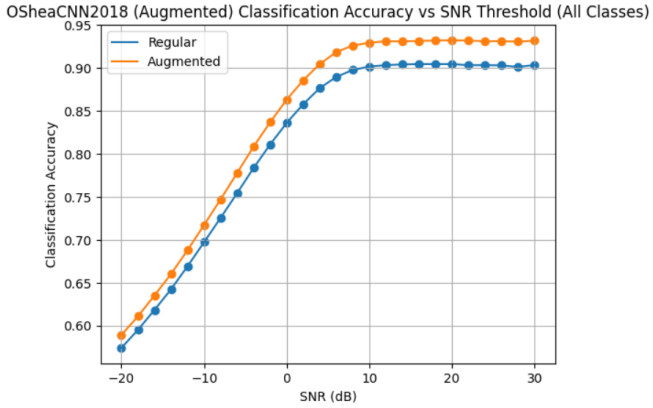

Fig. 10. OSheaCNN2018 Classification Accuracy vs. SNR Threshold

The OSheaCNN2018 models achieved an overall classification accuracy of 57.4/58.9% across all SNRs without/with data augmentation. At the highest SNR (i.e., 20 dB), OSheaCNN2018 achieves an accuracy of 90.5/93.2%. In Figure 10, applying data augmentation provides a 1-3% uplift in classification accuracy across all SNRs. As the RadioML 2018.01A already has lots of relatively high-resolution I/Q sequences, the classification accuracy is much higher than OSheaCNN2016. OSheaCNN2018 is designed to be a deeper model, exhibiting more hidden layers to account for the higher sample resolution with respect to their intended datasets.

C. HisarMod 2019.1

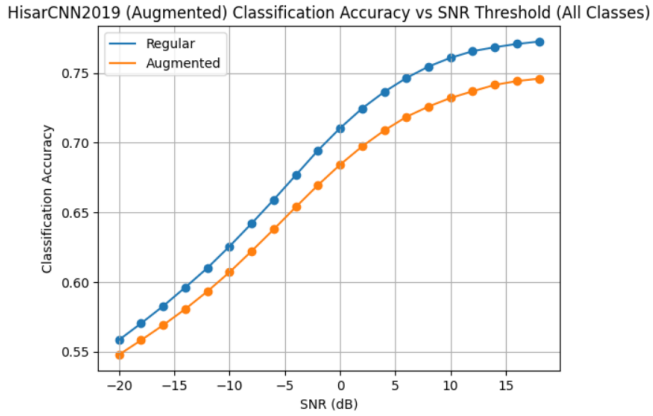

Fig. 11. HisarCNN2019 Classification Accuracy vs. SNR Threshold

The HisarCNN2019 models achieved an overall classification accuracy of 55.8/54.8% across all SNRs without/with data augmentation. At the highest SNR (i.e., 18 dB), OSheaCNN2018 achieves an accuracy of 74.6/77.3%. In Figure 11, applying data augmentation actually decreased overall accuracy by around 1-3% across all SNRs. A plausible explanation for this is that the model becomes slightly overfit at later epochs, foreshadowed earlier in Section III-E.

V. Conclusion

Training an AMR model is most effective when lots of high-resolution I/Q sequences are supplied, such as in the case of the RadioML 2018.01A/HisarMod 2019.1 dataset. Data augmentation by temporal reversal provides an improvement for all AMR architectures by making the models robust in the order of I/Q samples. A challenge of this project is being hardware limited when loading in the large datasets. Loading in RadioML 2018.01A already takes around 32 GB in memory, so further augmenting the data and storing them into tensors requires sophisticated garbage collection.

Future works of this project include training more advanced models that have a recurrent dependence on previous model weights (e.g. a hybrid CNN-LSTM) given less hardware constraints. Additionally, other data augmentations (e.g., block shuffling, I/Q rotation) should be added to make the model more generalizable. Lastly, the project can be applied in practice by collecting pure real-world data with software-defined radios (SDRs), testing on current AMR models, and fine-tuning them on non-simulated environmental factors.

Acknowledgment

The author thanks Professor Ian Roberts for providing the opportunity to work on this implementation project, in addition to providing insights during his Winter Quarter 2024 course "ECE 239AS: Advanced Techniques and Technologies in Modern Wireless Communications" at UCLA. Additionally, the author thanks his ECE239AS peers for providing feedback on throughout the project's progress.

References

[1] V. Iglesias, J. Grajal, and O. Yeste-Ojeda, "Automatic modulation classifier for military applications," in *2011 19th European Signal Processing Conference*, 2011, pp. 1814–1818.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. 25th Int. Conf. Neural Inf. Proc. Syst.*, ser. NIPS'12, vol. 1. Red Hook, NY, USA: Curran Associates Inc., Dec. 2012, pp. 1097—1105.

[3] A. Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," in *Proc. Intl. Interdiscip. PhD Wkshp.*, May 2018, pp. 117–122.

[4] T. J. O'Shea, J. Corgan, and T. C. Clancy, "Convolutional radio modulation recognition networks," in *Engineering Applications of Neural Networks*, C. Jayne and L. Iliadis, Eds. Cham: Springer International Publishing, Aug. 2016, pp. 213–226.

[5] N. E. West and T. O'Shea, "Deep architectures for modulation recognition," in *Proc. IEEE Int. Symp. Dyn. Spectr. Acc. Netw.*, Mar. 2017, pp. 1–6.

[6] T. J. O'Shea, T. Roy, and T. C. Clancy, "Over-the-air deep learning based radio signal classification," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 168–179, Jan. 2018.

[7] S. Rajendran, W. Meert, D. Giustiniano, V. Lenders, and S. Pollin, "Deep learning models for wireless signal classification with distributed low-cost spectrum sensors," *IEEE Trans. on Cogn. Commun. Netw.*, vol. 4, no. 3, pp. 433–445, Sep. 2018.

[8] K. Karra, S. Kuzdeba, and J. Petersen, "Modulation recognition using hierarchical deep neural networks," in *Proc. IEEE Int. Symp. Dyn. Spectr. Acc. Netw.*, Mar. 2017, pp. 1–3.

[9] T. J. O'Shea and N. West, "Radio machine learning dataset generation with gnu radio," *Proc. 6th GNU Radio. Conf.*, Sep. 2016.

[10] E. Blossom, "Gnu radio: tools for exploring the radio frequency spectrum," *Linux J.*, vol. 2004, p. 4, Jan. 2004.

[11] K. Tekbiyik, A. R. Ekti, A. Gorcin, G. K. Kurt, and C. Kececi, "Robust and fast automatic modulation classification with cnn under multipath fading channels," in *Proc. IEEE Veh. Technol. Conf.* IEEE, May 2020. [Online]. Available: http://dx.doi.org/10.1109/VTC2020-Spring48590.2020.9128408

[12] T. Do, "ECE239AS-MWC-AMR." [Online]. Available: https://github.com/dotimothy/ECE239AS-MWC-AMR

[13] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," Dec. 2019.

[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[16] K. Tekbıyık, C. Keçeci, A. R. Ekti, A. Görçin, and G. Karabulut Kurt, "Hisarmod: A new challenging modulated signals dataset," 2019. [Online]. Available: https://dx.doi.org/10.21227/8k12-2g70

[17] L. Huang, W. Pan, Y. Zhang, L. Qian, N. Gao, and Y. Wu, "Data augmentation for deep learning-based radio modulation classification," *IEEE Access*, vol. 8, pp. 1498–1506, 2020.

Timothy Do (Student Member, IEEE) is a first year M.S. student in Electrical and Computer Engineering at the University of California, Los Angeles, specializing in Signals and Systems. He received a B.S. in Electrical Engineering, double specializing in Digital Signal Processing and Communications, from the University of California, Irvine. His research interests include hyperspectral image processing and computer vision. He also has industry experience, interning with Western Digital and Maxeon Solar developing embedded IoT systems.