MODREC: COMPARING OPTIMIZING TECHNIQUES IN AUTOMATIC MODULATION RECOGNITION USING CNN

Aryan Kargwal[1]
[1]Énergie Matériaux Télécommunications Research Centre, INRS, Place Bonaventure, Montreal, Quebec, Canada
aryan.kargwal@inrs.ca

Abstract: In the following study, we are experimenting with various model optimizing techniques for traditional basic LeNet-5 like CNN architecture for Automatic Modulation Recognition. Through this study, we intend to find the effect of different optimization techniques to achieve faster and better generalization in our model. The inspiration for the work comes from more modular and easily deployable Modulation Recognition models, which can maximize the Signal-to-Noise Ratio in remote deployment setups such as Federated Learning.
Index Terms: Convolutional Neural Networks, Model Optimization, Automatic Modulation Recognition, Signal to Noise Ratio

I. INTRODUCTION

Wireless communication systems have become one of the most integral parts of modern human life, not only due to their applications in bringing people closer but also modern-day applications such as Internet of Things. With these systems, modulation recognition plays a vital role in deciphering transmitted signals, allowing efficient data transmission across distances and enhancing reception strength among the masses covered by such systems.

However, the performance of CNN-aided AMRs has effectively mapped out essential characteristics such as a diverse range of modulation types, SNR levels, and Channel Impairments to facilitate more robust systems.

CNNs have been used for years for tasks such as Image Classification [1], Object Detection [2], and more. However, they have been proven useful in dealing with multi-variate problems, given their ability to deal with multi-dimensional tensors.

With the rise of CNN architectures, we have also witnessed a substantial increase in optimizing techniques ranging from traditional methods such as Gradient Descent optimization to more advanced optimizations such as Adaptive Learning Rate [3], Model Weight Decay [4], and Model Pruning [5].

This research aims to answer the following questions through a series of benchmarks among two optimized LeNet-5 [6] like CNN architectures:

1. How do different optimization techniques affect the accuracy of CNN-based AMR schemes and SNRs?
2. What is the computational overhead associated with such systems, and can they be deployed on remote edge

computing or federated learning pipelines?

To facilitate the comprehensive evaluation, we will utilize the RML2016.10a [7] dataset, which encompasses many Modulation Types and SNR levels in real-life situations. The findings presented herein advance the state-of-the-art narrative of modern deep-learning telecommunication infrastructures and are built upon the works of Zhang et al. [8].

II. MODULES

Moving forward, we will mention the two models trained as "BAMR," or Basic AMR, and "OAMR," or Optimized AMR. Let us look at a comprehensive breakdown of modules and optimizations we have used in the experiments, highlighting the importance and need of adding such optimizations.

A. Dropout Layer

Both BAMR and OAMR have been coded to have a dropout layer after every convolution layer. Dropout [9] is a regularization technique mainly used in CNNs to prevent overfitting. Overfitting is when the model becomes too used to the training dataset and cannot perform well on the validation set. This condition makes the model unable to generalize to unseen data and impractical for real-life deployment.

Dropout addresses the issue of overfitting by randomly "dropping out" or temporarily removing a portion of the architecture's neurons, forcing the model to train with random, lesser neurons, become more robust, and present generalized inferences.

B. Leaky ReLU

Leaky ReLU [10] or Leaky Rectified Linear Unit is an activation function used in CNNs. It is a variation of the traditional ReLU function, which introduces a slight slope for negative input values instead of setting them to zero. This method helps over the traditional ReLU by avoiding the dying ReLU problem in which, during training, few weights stop updating and halting learning for those neurons due to the gradient of ReLU setting the weight to zero when faced with a negative input.

$$\phi(x) = \begin{cases} \alpha x & x \leq 0 \\ x & x > 0 \end{cases}$$

Both BAMR and OAMR come with this activation function for each convolutional layer.

C. Categorical Crossentropy

Categorical Crossentropy [11] is a loss function used mainly in multi-class classification. The module can measure the dissimilarity between the proper distribution of the class labels and the predicted distribution output by the model.

$$L_{\mathrm{CE}} = -\sum_{i=1}^{n} t_i \log(p_i),$$

In practice, the loss function is used with a Softmax layer to ensure the predicted probabilities sum up to 1, forming a valid and confident probability distribution.

D. Batch Normalization

This normalization technique is often used in CNNs to normalize the activations of each layer by adjusting and scaling the inputs to have a mean of zero and a standard deviation of one across the batches during training.

This normalization is applied to intermediate layers of the network, usually just after the convolution layers.

E. L2 Regularization

L2 Regularization [12] or weight decay is commonly used in CNN Architectures to avoid overfitting. It does so by penalizing large

weights and encouraging the neurons to hold low but non-zero weights. The module helps the model to learn more straightforward and smoother weight vectors by minimizing the sum of the squared magnitudes of the weights.

L2 Regularization

$$\text{Cost} = \underbrace{\sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2}_{\text{Loss function}} + \underbrace{\lambda \sum_{j=0}^{M} W_j^2}_{\text{Regularization Term}}$$

Mathematically, the model penalizes the said large weights by adding a term proportional to the squared magnitude of each weight to the loss function and encouraging smaller and more balanced weight values.

III. PROPOSED WORKFLOW

For the study, we propose the following workflow to replicate CNN-based AMR for the RML2016.10a [7] dataset. The method focuses on implementing Modulation Recognition with two similar CNN architectures, distinguished by the traditional CNN optimization techniques, and comparing the two across various parameters. The study aimed to estimate performance increase across parameters such as Model Size, Training Time, Validation Loss, and Validation accuracy.

As shown in Figure 1, the process is divided into the following steps. The models will be trained on the RML2016.10a dataset, where the sampling dimensions are 2 x 129, consisting of 220,000 data samples for SNR, which ranges from -20 to 18 dB and covers over 11 modulation schemes. We intend to compete with state-of-the-art architectures under this benchmark, including the works from O'Shea et al. [13] and Tekbiyik et al. [14], which achieved an accuracy of 56.4% and 56.6% on the benchmark.

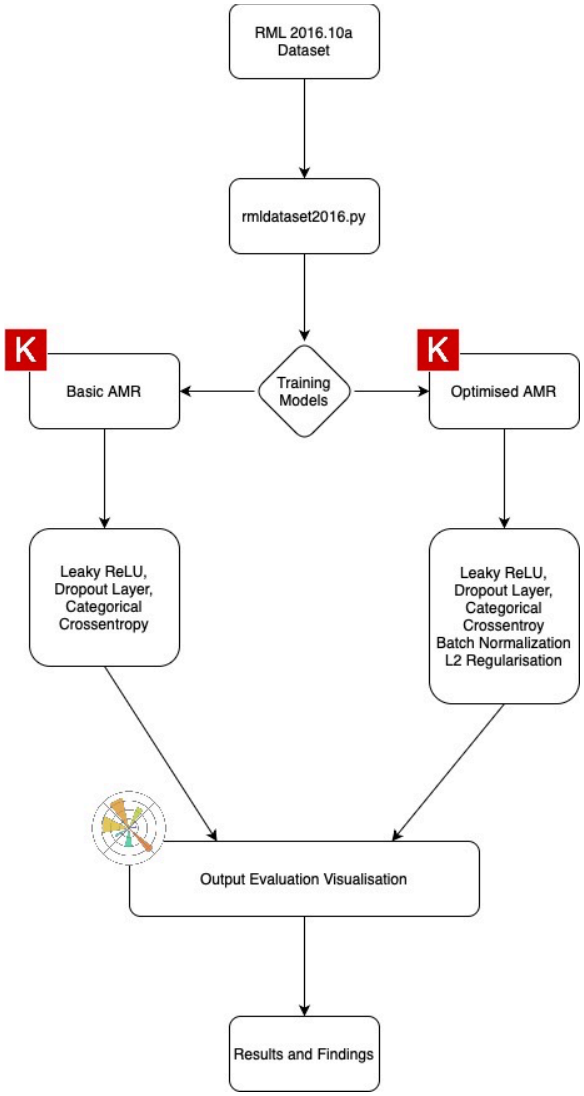

Fig. 1 Proposed Workflow

A. Dataset Preparation

The dataset, as mentioned, comes with 220,000 data samples for SNRs across 11 modulation schemes. Our data loading code, "rmldataset2016.py", uses a Stratified Sampling method for splitting the dataset into training and validation sets. Stratified Sampling is a sampling technique that will ensure that each modulation scheme at each SNR level is represented in both the training and validation sets.

Once the dataset is sampled, the modulations are converted into one-hot encoded vectors for the classification task.

B. Basic AMR Model Development and Training

The Basic AMR Model consists of two convolutional layers, each followed by a dropout layer to avoid overfitting. Regarding the activation functions for each layer, we are using Leaky ReLU for the Convolutional Layers, ReLU for Dropout Layers, and a softmax function for the final prediction layer, which gives us one of the possible outputs.

Layer	**Output Shape**	**Parameters**
Input	(None,2,128,1)	0
Reshape	(None,2,128,1)	0
Conv1	(None,2,128,50)	450
Dropout	(None,2,128,50)	0
Conv2	(None,1,121,50)	40050
Dropout	(None,1,121,50)	0
Flatten	(None, 6050)	0
Dense1	(None, 256)	1549056
Dropout	(None,256)	0
Dense2	(None,11)	2827
Activation	(None,11)	0

Table I Model Summary BAMR

You can access the summary of model parameters in Table I. The model is trained on the following hyperparameters:

- Input Shape: The shape of the input data, defined as [2, 128].
- Number of Convolutional Filters: 50 filters are used in each convolutional layer.
- Dropout Rate: Dropout rate of 60% is applied after each convolutional and dense layer.
- Number of Neurons in Dense Layers: 256 neurons are used in the first dense layer.
- Number of Output Classes: 11 classes are predicted by the final output layer.
- Kernel Size: The convolutional filters have different kernel sizes, [1, 8] and [2, 8], for the two convolutional layers.
- Trainable Parameters: 1592383 (6.07mb)
- Batch Size: 500
- Epochs: 50
- Dropout: 0.7

Going into the actual training process, the model is trained using the ‘fit’ method, specifying the training data, batch size, number of epochs, validation data, and callbacks. Callbacks such as Model Checkpointing, Learning Rate Reduction, and Early Stopping are used to save computational resources and monitor the training process by implementing subtle adjustments to the training process.

C. Optimized AMR Model Development and Training

The development of the model for the Optimized AMR is almost the same as the Basic AMR architecture, barring the introduction of Batch Normalization between every linear and non-linear layer, i.e. between Conv1 and Dropout1, Conv2 and Dropout2, and Dense1 and Dropout3. We also increase the number of epochs and change the dropout rate.

Apart from this, we introduce L2 Regularization or Ridge Regularization in the Convolution Layers. You can access the summary of the model parameters in Table II. This model also includes similar hyperparameters as Basic AMR barring some extra trainable and non-trainable

Parameters because of the Batch Normalization Layers:

- Trainable Parameters: 1593095
- Non-Trainable Parameters: 712
- Total Parameters: 1593807
- Epoch: 100
- Batch Size: 500
- Dropout: 0.6

Layer	**Output Shape**	**Parameters**
Input	(None,2,128,1)	0
Reshape	(None,2,128,1)	0
Conv1	(None,2,128,50)	450
BN1	(None,2,128,50)	200
Dropout	(None,2,128,50)	0
Conv2	(None,1,121,50)	40050
BN2	(None,1,121,50)	200
Dropout	(None,1,121,50)	0
Flatten	(None, 6050)	0
Dense1	(None, 256)	1549056
BN3	(None,256)	1024
Dropout	(None,256)	0
Dense2	(None,11)	2827
Activation	(None,11)	0

Table II Model Summary OAMR

IV. RESULTS AND FINDINGS

As part of the study, we ran a few evaluations and have plotted some of them to show the difference between the two architectures in terms of

A. Accuracy

As part of Table III, you can see the various accuracy and loss values for both BAMR and OAMR. The comparison shows that OAMR is able to perform better than BAMR considerable in terms Accuracy and Loss.

	BAMR	**OAMR**
Test Acc	50.18%	**54.85%**
Training Acc	51.14%	**56.00%**
Validation Acc	50.22%	**54.95%**
Training Loss	1.25	**1.14**
Validation Loss	1.30	**1.17**

Table III Accuracy Comparison

The accuracy also comes very close to our reference papers, within a fraction of training time and Epochs. Here we see that the inclusion of L2 regularization provides a better accuracy across the board. We see that the delta between the Training and Validation Accuracy also changes as we change the Dropout value, making the model more robust.

B. Classification Accuracy for each Modulation

Classification accuracy in the context of different modulations refers to the ability of a classification model to correctly identify the modulation schemes of signals from a mixed dataset containing various modulation types. In radio signal processing, different modulation schemes encode information onto carrier signals differently.

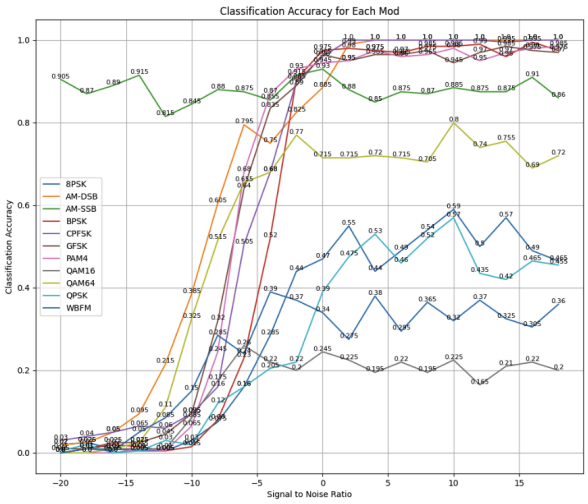

Fig 2 Classification Accuracy for BAMR

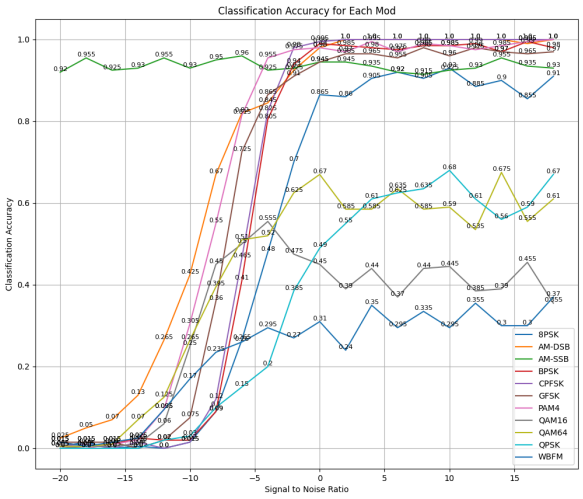

Fig 3. Classification Accuracy for OAMR

As seen from the comparison of the classification accuracy for both BAMR and OAMR in Figures 2 and 3, respectively, we can see that the addition of L2 Regularization has helped improve the Classification accuracy at every SNR level, with the most considerable difference coming in the AM-SSB modulation.

C. Classification Curve for each SNR

The classification curve for each signal-to-noise ratio (SNR) level provides insights into a classification model's performance across different noise levels in the received signals. This metric helps us understand how the model will perform under various noise situations.

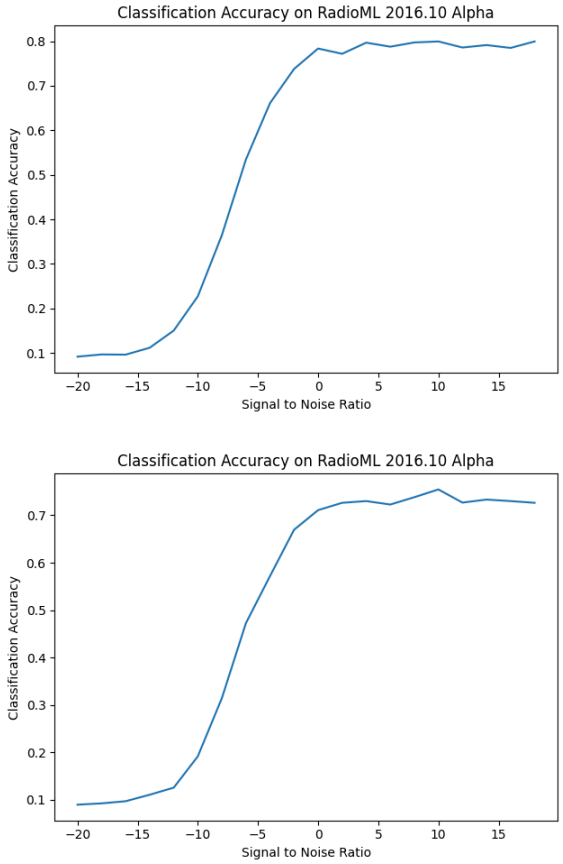

Fig 4. Classification Accuracy on RML2016.10a

Figure 4 shows us that although the models have considerable accuracy differences in terms of pure training and accuracy across various mods, the overall classification accuracy of the model remains almost the same for both models, which makes them act similarly across different noise environment.

D. Final Takeaways

from the metrics and results is that OAMR is the choice of the model when we are looking to have a clear distinction and want to predict the classification of general niche mods. However, given the extra training time and cost, OAMR, when put against BAMR in a general classification benchmark, produces similar results and can cost an avoidable overhead.

From the experiment, we understand that Regularization and Batch Normalization can help a model flourish; however, the model needs to be trained for a longer time for considerable differences. In both cases, we saw that the Learning Rate adapter was triggered, which indicates that a more complex architecture is required to avoid a training plateau.

V. REFERENCES

[1] He, K., Zhang, X., Ren, S., & Sun, J. (2015, December 10). Deep Residual Learning for Image Recognition. ArXiv.org. https://arxiv.org/abs/1512.03385

[2] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015, June 8). You Only Look Once: Unified, Real-Time Object Detection. ArXiv.org. https://arxiv.org/abs/1506.02640

[3] Singh, B., De, S., Zhang, Y., Goldstein, T., & Taylor, G. (n.d.). Layer-Specific Adaptive Learning Rates for Deep Networks. Retrieved April 19, 2024, from https://arxiv.org/pdf/1510.04609.pdf

[4] Krogh·, A., & Hertz, J. (n.d.). A Simple Weight Decay Can Improve Generalization. https://proceedings.neurips.cc/paper/1991/file/8efcfdf5990e441f0fb6f3fad709e21-Paper.pdf

[5] Pasandi, M., Hajabdollahi, M., Karimi, N., & Samavi, S. (n.d.). Modeling of Pruning Techniques for Deep Neural Networks Simplification. https://arxiv.org/pdf/2001.04062.pdf

[6] Lecun, Y., L Eon Bottou, Bengio, Y., & Patrick Haaner Abstract|. (1998). Gradient-Based Learning Applied to Document Recognition. PROC. OF the IEEE. http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf

[7] Datasets. (n.d.). DeepSig. Retrieved April 19, 2024, from https://www.deepsig.ai/datasets/

[8] Zhang, F., Luo, C., Xu, J., Luo, Y., & Zheng, F. (2022, July 20). Deep Learning Based Automatic Modulation Recognition: Models, Datasets, and Challenges. ArXiv.org. https://doi.org/10.48550/arXiv.2207.09647

[9] Srivastava, N., Hinton, G., Krizhevsky, A., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research, 15, 1929–1958. https://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf

[10] Xu, B., Wang, N., Chen, T., & Li, M. (2015). Empirical Evaluation of Rectified Activations in Convolutional Network. ArXiv.org. https://arxiv.org/abs/1505.00853

[11] Zhang, Z., & Sabuncu, M. R. (2018). Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels.

ArXiv:1805.07836 [Cs, Stat]. https://arxiv.org/abs/1805.07836

[12] Ng, A. (n.d.). Feature selection, L 1 vs. L 2 regularization, and rotational invariance. https://icml.cc/Conferences/2004/proceedings/papers/354.pdf

[13] O'Shea, T. J., Corgan, J., & Clancy, T. C. (2016). Convolutional Radio Modulation Recognition Networks. ArXiv:1602.04105 [Cs]. https://arxiv.org/abs/1602.04105

[14] Tekbıyık, Kürşat & Ekti, Ali & Gorcin, Ali & Karabulut Kurt, Gunes & Kececi, Cihat. (2020). Robust and Fast Automatic Modulation Classification with CNN under Multipath Fading Channels. 1-6. 10.1109/VTC2020-Spring48590.2020.9128408.