Robin Lin
ECE4200 – Introduction to Machine Learning
NetID: zl755
Word Count: 793

Pre-processing and Data Augmentation

When I first received the assignment, I decided to visualize the training dataset to understand the different modulation schemes and their associated constellation diagrams. Figures 1, 2, 3, and 4 show the constellation diagrams of four different modulation schemes at various Signal-to-Noise ratios.

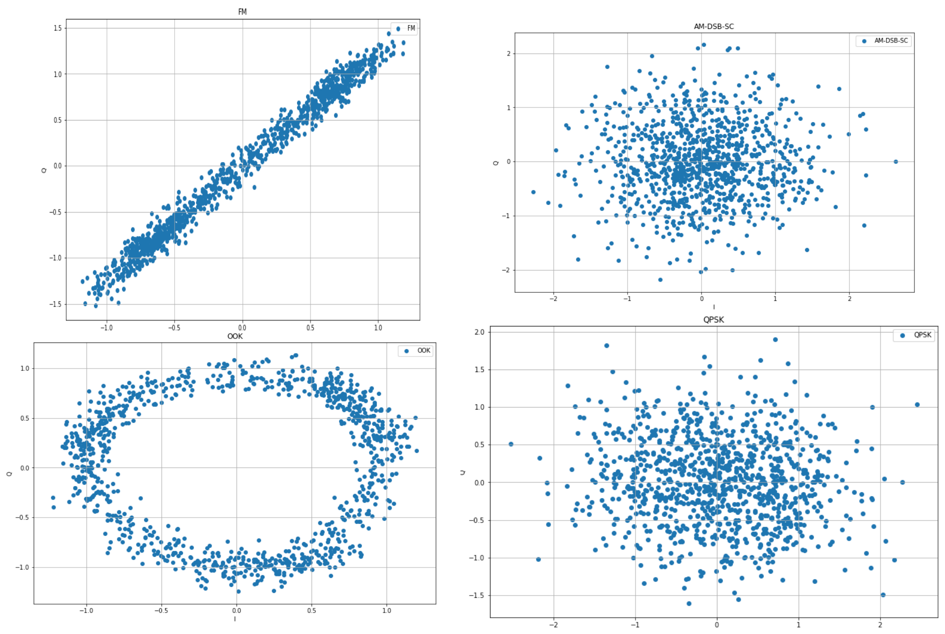


Figure 1, 2, 3 and 4: FM, AM-DSB-SC, OOK, QPSK Constellation Diagrams

The I/Q constellation diagrams for FM and OOK above are quite distinguishable, while the diagrams for QPSK and AM-DSB-SC are almost identical. Thus, my first intuition was to figure out a method for reducing the noise and increase the SNR. Initially, I tried to conduct a literature review on methods for SNR improvements. However, most of the techniques for noise reduction were reasonably complex. Then, I decided to down-sample the I/Q signals down to 512 x 2. This proved to be not much of an improvement. Finally, the last method I tried was data augmentation based on "Data Augmentation for Deep Learning-based Radio Modulation Classification" [1], which includes techniques such as rotation, flipping, and injecting gaussian noise. To further increase the reliability and usefulness of the dataset, the training set can be artificially expanded by adding different variations of a single training sample.

Rotation Matrix

$$\begin{bmatrix} I' \\ Q' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} I \\ Q \end{bmatrix} \quad (1)$$

According to the paper, rotation yielded the highest improvements in accuracy by around 3% at higher SNR ranges. Thus, I decided to use this rotation augmentation method, with rotation angles of 90° and 180° for the first 4000 training samples. This augmentation method proved to be useful, improving performance by around 5%. The "numpy" library was used for the transformation.

Classification Model

After conducting some literature review of the existing models for classifying modulation schemes, I came across several methods, including LSTM, SVM, and Convolutional Neural Networks (CNN) [2]. Overall, CNN was the most popular classifier. For the CNN model, different research publications used various input signals (FFT vs. I/Q) as well as different sizes (128x2 vs. 1024x2). For the competition, I ended up with two different CNN architectures, as shown in Figures 5 and 6 below.

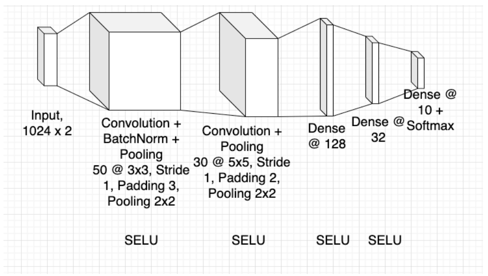

Input, 1024 x 2
Convolution + BatchNorm + Pooling 50 @ 3x3, Stride 1, Padding 3, Pooling 2x2
Convolution + Pooling 30 @ 5x5, Stride 1, Padding 2, Pooling 2x2
Dense @ 128
Dense @ 32
Dense @ 10 + Softmax
SELU
SELU
SELU
SELU

Figure 5. CNN Configuration 1

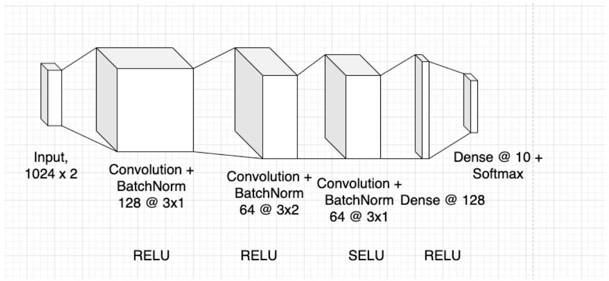

Input, 1024 x 2
Convolution + BatchNorm 128 @ 3x1
Convolution + BatchNorm 64 @ 3x2
Convolution + BatchNorm 64 @ 3x1
Dense @ 128
Dense @ 10 + Softmax
RELU
RELU
SELU
RELU

Figure 6. CNN Configuration 2
(Used for final rankings)

Robin Lin
ECE4200 – Introduction to Machine Learning
NetID: zl755
Word Count: 793

Before deciding on these two configurations, I experimented with various hyperparameters of the network, such as the number of convolution layers, the number of filters, the number of neurons in the dense layers, etc. By using larger filters, the model was more prone to underfitting, as the larger filters allow the model to learn coarse features and, thus, cannot converge to a desirable training accuracy. On the other hand, using smaller filters and a more significant number of neurons/dense layers leads to overfitting, where the training accuracy is significantly higher than the validation accuracy. Furthermore, I experimented with different activation functions, such as RELU, SELU, and LogSigmoid. I found the RELU and SELU activations to be quite good, leading to rapid convergence and shorter training time.

One of the biggest challenges for this classification problem was the fact that my models were very prone to overfitting. Both models can quickly converge to a training accuracy of 90%. However, the validation accuracies can reach at most 35% and 45% for Configuration 1 and 2. I spent a lot of time trying to tune different types of regularization to decrease the overfitting. One method I tried that led to excellent results was batch normalization. Batch normalization normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation, increasing the stability of training. Thus, the covariate shift between layers is reduced. Another method was using dropout, where specific neurons in the fully connected layers are "turned-off." This method did not improve the performance by too much. Thus, in the end, I decided to use batch normalization over dropout.

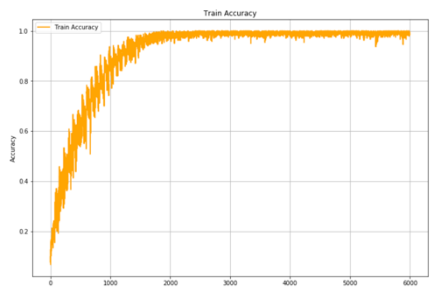
Training the Network

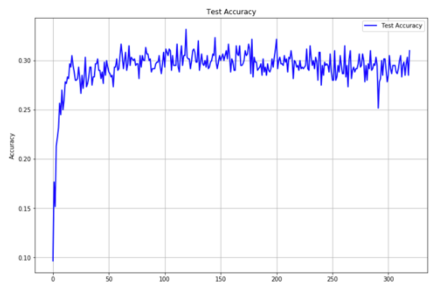
For training the network, the following configurations were used for Configuration 1 and 2.

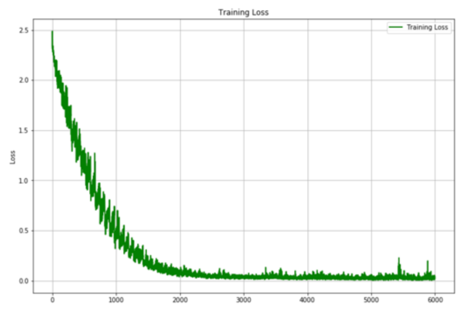
	Configuration 1	Configuration 2
Learning Rate	0.0002	0.0002
Optimizer	Adam	Adagrad
Mini-batch Size	240	240
L2 Regularization Parameter	N/A	0.00007

One significant consideration was whether to use batch training or mini-batch training. I decided to use mini-batch training because it intrinsically adds a regularization effect, as the model is iteration over different batches with different samples each time. The learning rate was tuned by scanning across a range of values. Also, an L2-regularizer was added for Configuration 2, leading to slight improvements in correcting overfitting. The training accuracy, validation accuracy, and training loss are shown below. Both models lead to high training accuracy but converged to significantly lower validation accuracies. In the end, Configuration 2 was chosen for the leaderboard, as it had an average converged validation accuracy of 42%, compared to 30% for Configuration 1.

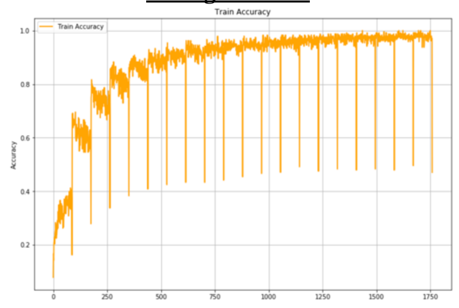
Configuration 1

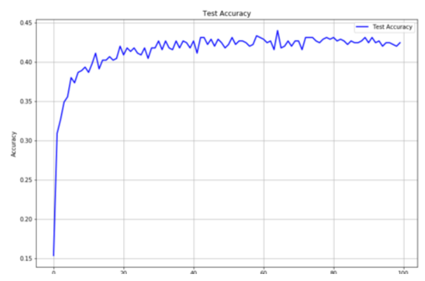


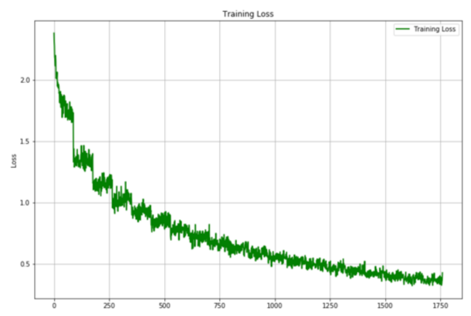




Configuration 2







Robin Lin
ECE4200 – Introduction to Machine Learning
NetID: zl755
Word Count: 793

Reflection

In the end, the project was very rewarding. I learned a lot about the challenges of balancing the model complexity with the potential for underfitting/overfitting, as well as how to tune and implement the Convolutional Neural Network architecture, a topic that was discussed very briefly in class. It was fulfilling to apply my knowledge to a technical project.

Bibliography

[1] Huang, Liang, et al. "Data Augmentation for Deep Learning-based Radio Modulation Classification." IEEE Access (2019).

[2] O'Shea, Timothy James, Tamoghna Roy, and T. Charles Clancy. "Over-the-air deep learning based radio signal classification." IEEE Journal of Selected Topics in Signal Processing 12.1 (2018): 168-179.