Deep and Ensemble Learning to Win the Army RCO AI Signal Classification Challenge

Andres Vila‡*, Donna Branchevsky‡, Kyle Logue‡, Sebastian Olsen‡, Esteban Valles‡, Darren Semmen‡, Alex Utter‡, Eugene Grayver‡

Abstract—Automatic modulation classification is a challenging problem with multiple applications including cognitive radio and signals intelligence. Most of the existing efforts to solve this problem are only applicable when the signal to noise ratio (SNR) is high and/or long observations of the signal are available. Recent work has focused on applying shallow and deep machine learning (ML) to this problem. In this paper, we present an exploration of such deep learning and ensemble learning techniques that was used to win the Army Rapid Capability Office (RCO) 2018 Signal Classification Challenge. An expert feature extraction and shallow learning approach is discussed in a simultaneous publication. We evaluated multiple state-of-the-art deep learning network architectures and adapted them to work in the RF signal domain instead of the image/computer-vision domain. The best deep learning methods were merged with the best expert feature extraction and shallow learning methods using ensemble learning. Finally, the ensemble classifier was calibrated to obtain marginal gains. The methods discussed are capable of correctly classifying waveforms at -10 dB SNR with over 63% accuracy and signals at +10 dB SNR with over 95% accuracy from an Army RCO provided training set.

Index Terms—modulation classification, neural networks, deep learning, machine learning, ensemble learning, wireless communications, signals intelligence, probability calibration

Introduction

All conventional communications systems are designed with the assumption that the transmitter and receiver are cooperative and have full knowledge of the waveform being exchanged. However, there are scenarios where the receiver does not know what waveform (i.e. modulation, coding, etc.) has been transmitted. Classical examples include cognitive radio network (i.e. a new terminal enters a network and needs to figure out what waveform is being used), and signals intelligence (i.e. interception of an adversary's communications). The problem of waveform classifications, or more narrowly, modulation recognition has been studied for decades [Aisbett]. Given the implication of SIGINT[1] applications before cognitive radio, much of the work had not been published. Key early work is done by Azzouz & Nandi [Nandi1], [Nandi2], [Azz1], [Azz2].

The fundamental approach taken by most authors has been to find data reduction functions that accentuate the differences between different waveforms. These functions are applied to input samples and a decision is made by comparing the values against a set of multi-dimensional thresholds. Determining the threshold values by hand becomes impractical as the number of clusters and/or functions grows. The idea to apply neural networks to help make these decisions has been around for decades [Azz2]. However, it is only recently that our understanding of machine learning combined with enormous increase in computational resources has enabled us to use ML techniques to solve this problem.

* *Corresponding author: andres.i.vilacasado@aero.org*
‡ *The Aerospace Corporation*

Challenge Description

The Army Rapid Capability Office is seeking innovative approaches to leverage artificial intelligence (AI) to conduct blind radio frequency signal analysis. To this end, they published a labeled modulation classification dataset and created a competition [Army] to properly classify a pair of unlabeled test sets. This paper details the efforts of The Aerospace Corporation's Team Platypus, the authors of this paper, to build a modulation classification system via deep learning and ensemble learning. In this context, deep refers to the fact that the ML classifier will use the raw IQ[2] data, instead of expertly engineered features. The winning submission from Team Platypus utilized a combination of these deep classifiers and shallow learning classifiers built on expert features which are described in a simultaneous companion publication.

The training dataset [Mitre] consists of 4.32 million signals each of which contain 1024 complex (IQ) points and a label indicating the modulation type and SNR. Modulation type is selected from one of 24 digital and analog modulations (including a noise class), with AWGN at six different signal-to-noise ratios (-10, -6, -2, +2, +6, or +10 dB). The complete dataset included 30,000 rows for each modulation and SNR configuration. Sample rate is selected from a set (200, 500, 1000, or 2000 ksps), and symbol rate is selected from a set (4, 8, 16, or 32 samples per symbol). Neither of the rate parameters is included in the label.

The competition consisted of assigning a likelihood score to each of the 24 possible modulation classes for each of the 100,000 rows in a pair of unlabeled test sets.

Classifier performance is evaluated via a pre-defined equation based on the well-known log loss metric. The traditional log loss equations:

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log p_{ij} \quad (1)$$

1. Signals Intelligence
2. In-Phase & Quadrature

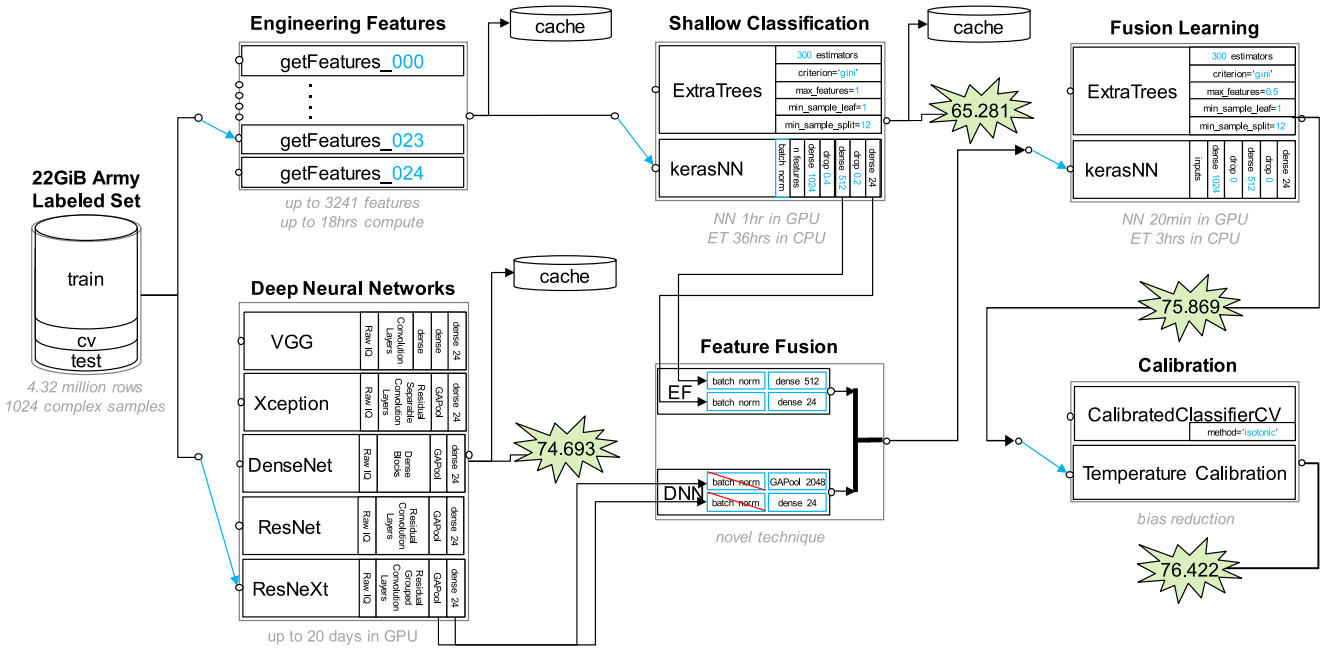

Fig. 1: *Data flow through the classification pipeline. The many variable parameters available are denoted in light blue.*

Where N is the number of instances in the test set, M is the number of modulation class labels (24), y_{ij} is 1 if test instance i belongs to class j and 0 otherwise, p_{ij} is the predicted probability that observation i belongs in class j. The competition score, which we will refer to as simply the score in the remainder of this paper, was defined per [Mitre] as follows:

$$score = \frac{100}{1 + logloss} \tag{2}$$

Notes:

- A uniform probability estimate would yield a score of 23.935, not zero.
- To get a perfect 100 score participants would need to be both 100% correct and 100% confident of every estimation.

We will also use a more standard F_1 metric for each modulation is used. This is an excellent measurement of classifier performance since it uses both recall r and precision p, which better account for false negatives and false positives:

$$r = \frac{\sum true\ positive}{\sum false\ negative + \sum true\ positive} \tag{3}$$

$$p = \frac{\sum true\ positive}{\sum false\ positive + \sum true\ positive} \tag{4}$$

$$F_1 = \frac{2}{\frac{1}{r} + \frac{1}{p}} \tag{5}$$

Approach

Team Platypus' approach to solve this modulation classification problem is to combine deep neural networks and shallow learning classifiers leveraging custom engineering features. Both of these are supervised machine learning systems.

Figure 1 shows the general flow of data through our winning system. The labeled training data is split into training, cross-validation, and testing using a 70%-15%-15% split. When using neural networks, the cross-validation set is used to prevent classifier overfitting. Using the Army RCO score metric, the final version of this system scored 76.422. This equates to a cross-validation log loss of 0.308. The output of each step is written to large cache files to enable quick evaluation of new ideas and integration into the next processing pipeline.

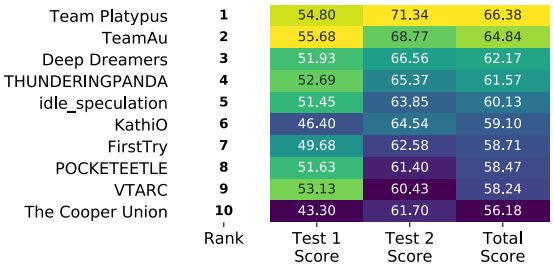

Fig. 2: *Final Army RCO AI Signal Classification leaderboard.*

Classification Strategy & Scores

There were two unlabeled sets released to competitors. Estimates generated for the first set using our deep neural network estimator resulted in very low and inconsistent scores. It was apparent that the data was very unlike the training data initially provided. Team Platypus estimates that only half of the first unlabeled set was like the training set. Our solutions for these datasets relied exclusively on expert engineering feature extraction and shallow classification techniques. Only one of the competitors achieved a higher score (0.8 points) for this set.

The challenge administrators disclosed that the second set contained data 95% like the training set. As such, a combination of a deep learning and shallow learning techniques as described in the rest of this paper was used to generate the submissions for this dataset. Team Platypus held the highest submission score for the duration of the challenge.

Network Type	Best Scores
Simple Convolutional	45.47
VGG	58.38
Modified ResNet	66.21
ResNet-34	72.39
ResNet-50	72.80
ResNeXt-50	74.69
Xception	70.74
DenseNet	65.98

TABLE 1: *Deep Learning Results.*

Deep Learning Modulation Classification

Architecture Search

We implemented multiple Neural Network architectures in Keras using the TensorFlow backend. We begun by testing variations of the networks proposed in [OShea1]. These networks consisted of 2 or 3 convolutional layers followed by 2 or 3 dense layers. We will call these networks "Simple Convolutional". These networks produced scores of around 45 points. We proceeded to test 2 networks proposed in [OShea2], a VGG network and a "Modified ResNet" network. The VGG network produces results around 55 points and the "Modified ResNet" resulted in a score of 59 points.

Our search strategy changed at this point. We conjectured that using the state-of-the-art methods currently applied to image classification would yield good results. Hence, we implemented multiple algorithms by reading their papers and adapting their ideas from 2-dimensional (images) to single dimensional (complex time-series signals). We could not rely on previously built Keras application models since they were all built for the 2-dimensional images classification problem.

We implemented multiple ResNets [ResNet1], [ResNet2], ResNeXts [ResNeXt], DenseNets [DenseNet] and Xception networks [Xception]. Their respective papers provided the number of layers, the number of channels per layer and multiple other details that we never modified in order reduce the number of parameters to tune.

Tuning, Testing and Results

We tested these architectures with different regularization parameters, location of pooling layers and convolution window sizes. The best performance for the different architectures can be found in Table 1. The best performance we obtained during the competition was from a ResNeXt-50 network with a log loss of 0.339. Due to the constraints of the competition, the sub-optimal results of Xception and DenseNet networks may be due to lack of expert tuning time and not an inherent deficiency of these architectures for this problem.

The convolution window size turned out to influence performance dramatically. We found early on that increasing the window size would increase the complexity of the models as well as the score. Our winning ResNeXt-50 network uses window size 64 to obtain its 74.69 score. After the competition we trained the same network with a convolutional window size of 3 and obtained a score of 64.2 which would not have won the challenge.

Merging and Probability Calibration

Merging

As shown in Figure 1, we merged the best Engineering Features (EF) network with the best Deep Learning (DL) network. We merged by taking metrics from both the EF and DL networks as features to go into a new dense neural network. The metrics that worked best were the logit outputs of the last layer of both EF and DL networks as well as the outputs of the penultimate layer of both networks. We believe this to be a novel idea for merging diverse neural networks. We tested using outputs of earlier layers on both networks and didn't obtain a better performance.

The classifier that produced the best results for these new features was a dense neural network. At the input of the merging neural network we use a batch normalization layer [Ioffe] for the features that come from the EF network only. We then concatenate both sets of features and connect them to a dense network that has 2 hidden layers of size 1024 and 512 respectively. The output layer has size 24 which corresponds to the original number of modulations in the challenge.

For reference the code to instantiate the best neural net merging classifier is:

```
from keras.layers import Input,
                         BatchNormalization,
                         Concatenate,
                         Dense,
                         Activation
from keras.models import Model

#Deep Neural Net inputs
main_input1 = Input(shape=(2048,))
main_input2 = Input(shape=(24,))
#Engineering Features Neural Net inputs
auxiliary_input1 = Input(shape=(512,))
auxiliary_input2 = Input(shape=(24,))
#Batch normalizing Engineering Feature layers
x1 = BatchNormalization()(auxiliary_input1)
x2 = BatchNormalization()(auxiliary_input2)
#Concatenate Layers
x = Concatenate([main_input1,main_input2,x1, x2])
#Put through Dense Network
x=Dense(1024, activation='relu', init='he_normal')(x)
x=Dense(512, activation='relu', init='he_normal')(x)
x=Dense(24, init='he_normal')(x)
output=Activation('softmax')(x)
model = Model(inputs=[main_input1,
                      main_input2,
                      auxiliary_input1,
                      auxiliary_input2],
              outputs=output)
```

We tested other types of classifiers that we obtained by using AutoML. The AutoML package we used is TPOT [TPOT1], [TPOT2] which is built on top of scikit-learn. TPOT proposed to use a combination of Linear Support Vector Classification (sklearn.svm.LinearSVC), Naive Bayes for multivariate Bernoulli models (sklearn.naive_bayes.BernoulliNB) and Logistic Regression (sklearn.linear_model.LogisticRegression).

The code to instantiate the best AutoML generated merging classifier is:

```
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import LogisticRegression
from tpot.builtins import StackingEstimator
import sklearn.feature_selection as sklfs

model = make_pipeline(
 sklfs.VarianceThreshold(threshold=0.1),
 StackingEstimator(
 estimator=BernoulliNB(alpha=100.0)),
```

Classifier(s)	Calibration	Pre-cal score	Post-cal score	Accuracy
Neural Network	Temperature	75.55	75.68	86.94
BernoulliNB and LogisticRegression	isotonic	74.75	74.8	87.2
BernoulliNB and LinearSVC	isotonic	73.9	74.74	87.2
LogisticRegression	isotonic	73.49	74.33	86.93
LinearSVC	isotonic	74.23	74.99	87.22

TABLE 2: *Sub-sampled merging and calibration results.*

Classifier(s)	Calibration	Pre-cal score	Post-cal score	Accuracy
Neural Network	Temperature	75.87	76.42	87.47
BernoulliNB and LogisticRegression	isotonic	74.97	75.14	87.2

TABLE 3: *Complete dataset merging and calibration results.*

```
 LogisticRegression(C=0.01, dual=False, penalty="l1",
                    tol=0.001)
)
```

Probability Calibration

The final step in the pipeline presented in Figure 1 is calibration. Probability calibration consists on modifying the final probabilities without changing the class that corresponds to the highest probability. It uses the 15% cross-validation data to shape the output probabilities to increase the score.

In order to calibrate our merging neural network we used a modification of the temperature scaling approach proposed in [Guo]. The temperature scaling approach finds the optimal temperature scalar to divide the output logits by, that minimizes the log loss on the cross-validation dataset. We extended this method by finding the separate optimal temperature scalars for each predicted modulation type using the cross-validation data. Temperature scaling consistently increased the score of neural nets from 0.3 to 0.6 points.

Calibration of the scikit-learn merging classifiers consisted on using the CalibrateClassifierCV class in scikit-learn [SKCal]. This class implements two different approaches for performing calibration: a parametric approach based on Platt's sigmoid model and a non-parametric approach based on isotonic regression. Our best results were achieved with the isotonic approach which were always between 0.1 to 0.9 points better than the uncalibrated score.

Merging and Calibration Results

The best merging and calibration results are presented in Table 2. These results were obtained by training on the same random subsample of the training dataset of size 144000. Table 3 shows the best merging and calibration results for both neural nets classifiers and scikit-learn classifiers when trained on the full training dataset.

Overall Performance

The accuracy of estimation can be visualized as a confusion matrix, shown in Figures 7 and 8 for the deep learning classifier and the final calibrated and merged classifier respectively.

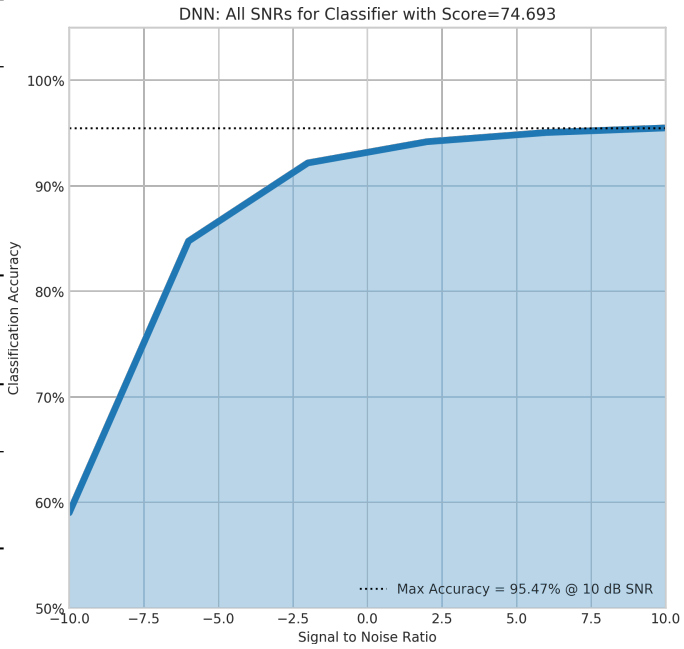

Fig. 3: *Classifier Accuracy vs SNR for deep learning network.*

Each row represents the true waveform, while each column is the estimated probability. The diagonal values correspond to the 'correct' estimate. Brighter colors indicate higher confidence (e.g. the top left square indicates almost 100% correct identification of the BPSK modulation). This view allows us to quickly identify waveforms that are challenging and to see where merging the deep learning classifier with the engineering features classifier helps. Calibration does not improve the confusion matrix since the winning class per sample doesn't change.

The F_1 score (see Challenge Description) provides another view of the same data. Figures 5 and 6 show the performances for the deep learning classifier and the final calibrated and merged classifier respectively. The overall classifier accuracy versus SNR is shown in Figures 3 and 4. Note that we achieve about 63% accuracy even at -10 dB SNR, which is significantly better than previously published results.

Conclusion

This paper showed the variety of ways machine learning techniques in python can be used to dramatically increase the performance of modulation classification algorithms. We presented a performance overview of different deep learning architectures when applied to the one-dimensional RF modulation-classification problem as presented in [Army] and [Mitre]. While the best performing architectures were ResNet and ResNeXt, we would caution against deducing that there is something inherent in those architectures that makes them more suited to the modulation-classification problem. Those algorithms produced the most promising results earlier on and thus, more time was spent running variations of them instead of trying to improve the performance of Xception or DenseNet networks.

This paper also presented a new merging method to fuse different neural networks. The novelty resides in what is being used as the input features of the merging classifiers. We used as inputs not only the results of the final layers of the original networks but the outputs of the last few layers of each of the initial neural networks.

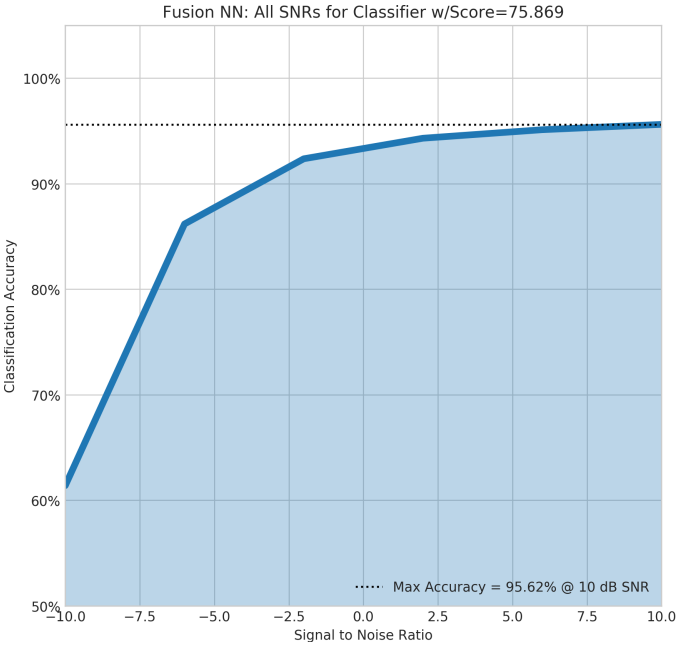

Fig. 4: *Classifier Accuracy vs SNR for final merging network.*

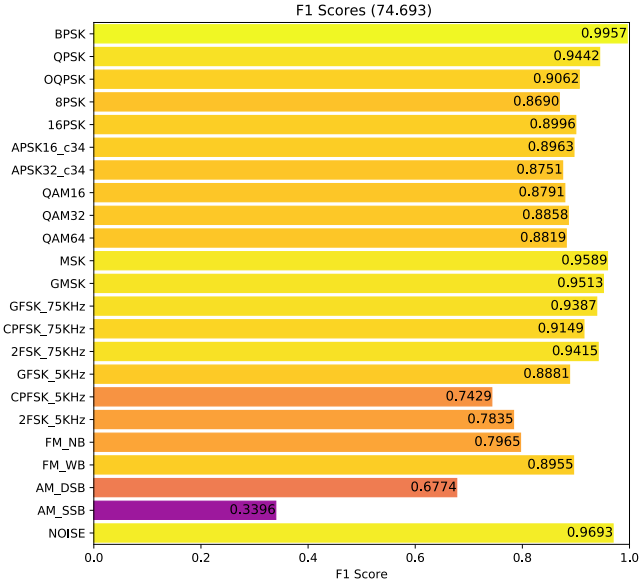

Fig. 5: *F_1 scores for all test data for deep learning network.*

Finally, we showed that calibration techniques can improve the log loss of diverse classifiers. However, it is important to note that the test cases offered by the Challenge are somewhat unrealistic. Real-world scenarios would include non-idealities like those described in [OShea2].

Acknowledgements

The authors would like to thank the Army RCO for creating this interesting challenge as well as our competitors who motivated us to stay up late and reconsider our assumptions.

REFERENCES

[Army] ARMY RCO AI Signal Classification Challenge. (2018). Retrieved from https://www.challenge.gov/challenge/army-signal-classification-challenge/

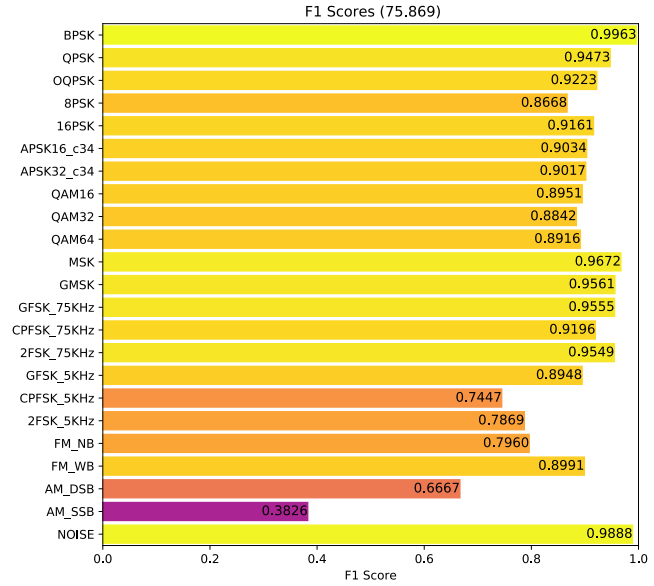

Fig. 6: *F_1 scores for all test data for final merged network.*

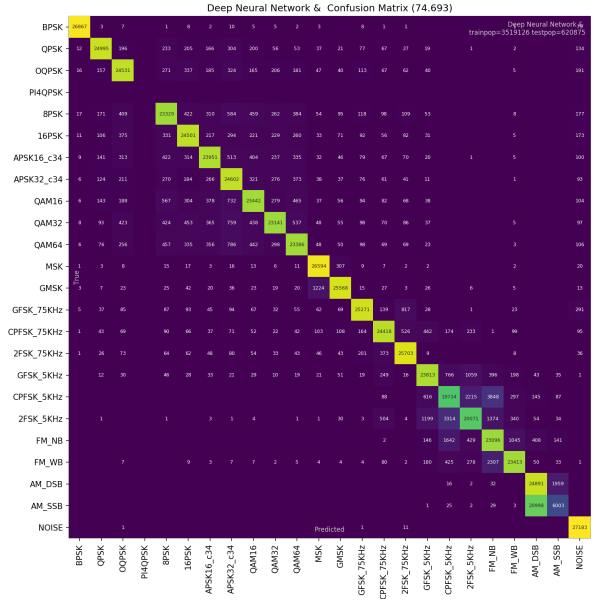

Fig. 7: *Confusion matrix for all test data for deep learning network.*

[Mitre] MITRE Challenge. (2018). Retrieved from https://sites.mitre.org/armychallenge/

[Nandi1] Nandi, Asoke K., and Elsayed Elsayed Azzouz. "Algorithms for automatic modulation recognition of communication signals." IEEE Transactions on communications 46.4 (1998): 431-436. doi:10.1109/26.664294.

[Nandi2] Nandi, A. K., and Elsayed Elsayed Azzouz. "Automatic analogue modulation recognition." Signal processing 46.2 (1995): 211-222. doi:10.1016/0165-1684(95)00083-p.

[Azz1] Azzouz, Elsayed, and Asoke Kumar Nandi. Automatic modulation recognition of communication signals. Springer Science & Business Media, 2013. doi:10.1007/978-1-4757-2469-1.

[Azz2] Azzouz, Elsayed Elsayed, and Asoke Kumar Nandi. "Modulation recognition using artificial neural networks." Automatic Modulation Recognition of Communication Signals. Springer, Boston, MA, 1996. 132-176. doi:10.1007/978-1-4757-2469-1_5.

[OShea1] T. J. O'Shea, J. Corgan, "Convolutional radio modulation recognition networks", CoRR abs/1602.04105, 2016. doi:10.1007/978-3-319-44188-7_16.

[OShea2] T. J. O'Shea, T. Roy and T. C. Clancy. "Over-the-Air Deep Learning Based Radio Signal Classification," in IEEE Journal of

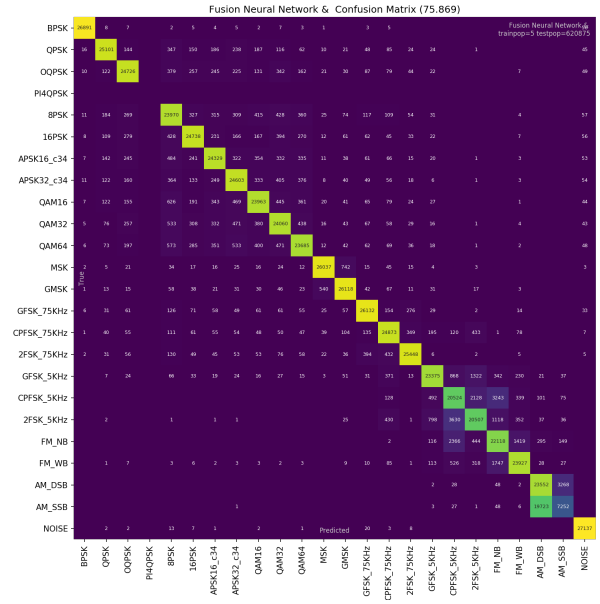

Fig. 8: *Confusion matrix all test data for final merged network.*

Selected Topics in Signal Processing, vol. 12, no. 1, pp. 168-179, Feb. 2018. doi:10.1109/JSTSP.2018.2797022.

[ResNet1] He, K., Zhang, X., Ren, S., Sun, J. "Deep residual learning for image recognition", CVPR arXiv:1512.03385. 2016.

[ResNet2] He, K., Zhang, X., Ren, S., Sun, J. "Identity Mappings in Deep Residual Networks", CVPR arXiv:1603.05027. 2016.

[ResNeXt] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, Kaiming He. "Aggregated Residual Transformations for Deep Neural Networks", CVPR arXiv:1611.05431. 2017.

[Xception] François Chollet. "Xception: Deep Learning with Depthwise Separable Convolutions", CVPR arXiv:1610.02357. 2016.

[DenseNet] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. "Deep Residual Learning for Image Recognition", CVPR arXiv:1512.03385. 2015.

[Guo] Chuan Guo, Geoff Pleiss, Yu Sun, Kilian Q. Weinberger. "On Calibration of Modern Neural Networks", ML arXiv:1706.04599. ICML 2017.

[SKCal] Probability calibration. Retreived from https://scikit-learn.org/stable/modules/calibration.html

[TPOT1] Randal S. Olson, Ryan J. Urbanowicz, Peter C. Andrews, Nicole A. Lavender, La Creis Kidd, and Jason H. Moore (2016). Automating biomedical data science through tree-based pipeline optimization. Applications of Evolutionary Computation, pages 123-137. 2016. doi:10.1007/978-3-319-31204-0_9.

[TPOT2] TPOT, a Python Automated Machine Learning tool. Retrieved from https://epistasislab.github.io/tpot/

[Aisbett] Aisbett, Janet. "Automatic modulation recognition using time domain parameters." Signal Processing 13.3 (1987): 323-328. doi:10.1016/0165-1684(87)90130-7.

[Ioffe] Sergey Ioffe, Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". ML arXiv:1502.03167. 2015.