Article:
Cai, J., Gan, F., Cao, X. et al. (2022) Signal modulation classification based on the transformer network. IEEE Transactions on Cognitive Communications and Networking, 8 (3). pp. 1348-1357. ISSN: 2372-2045

https://doi.org/10.1109/tccn.2022.3176640

Signal Modulation Classification Based on the Transformer Network

Jingjing Cai, Fengming Gan, Xianghai Cao and Wei Liu, *Senior Member, IEEE*

***Abstract*—In this work, the Transformer Network (TRN) is applied to the automatic modulation classification (AMC) problem for the first time. Different from the other deep networks, the TRN can incorporate the global information of each sample sequence and exploit the information that is semantically relevant for classification. In order to illustrate the performance of the proposed model, it is compared with four other deep models and two traditional methods. Simulation results show that the proposed one has a higher classification accuracy especially at low signal to noise ratios (SNRs), and the number of training parameters of the proposed model is less than those of the other deep models, which makes it more suitable for practical applications.**

***Index Terms*—Automatic modulation classification, transformer network, deep learning.**

I. Introduction

MODULATION classification plays a significant role in wireless spectrum monitoring [1], [2]. In the early stage, it usually relied on experts to provide decisions on modulation types based on the parameters of measured signals. However, this approach is not practical in crowded electromagnetic environments due to its slow response. Nowadays, various automatic modulation classification (AMC) techniques have been proposed, which are more suitable for scenarios encountered in modern warfare for its swift response [3], [4], [5], [6], [7]. There are mainly two categories of approaches for AMC: the likelihood based approaches and feature based ones.

For the first category, it is usually based on the hypothesis testing theory and constructs a judgment criterion by analyzing statistical characteristics of signals [8]. The ALRT (Average Likelihood Rate Test) algorithm was proposed in [9], which can distinguish BPSK (Binary Phase Shift Keying) and QPSK (Quadrature Phase Shift Keying) signals by employing the decision-theoretic approach based on Bayesian theory. The same approach was adopted in [10] to identify varieties of analog and digital signals, and a classifier based on qLLR (quasi-Log-Likelihood Ratio) rule was proposed in [11]. As the computational cost increases substantially in a complex signal environment due to an increasing number of parameters, a classifier based on Bayesian theory was developed for fast modulation classification in [12]. Prior knowledge about the signal model is essential for the likelihood based approaches, which limits their practical applications.

For the second category, feature based approaches mainly rely on discriminative features extracted from the raw signal data. These features mainly contain the following types, such as spectrum features [13], cumulants and moments [14], [15], [16], [17], [18], [19], and zero crossing features [3]. However, the above methods may not work well at low SNR since the derived features may not provide enough discriminating information. To reduce adverse effects of noise, wavelet transform was introduced in [20], which requires no prior information of the received signal such as signal sampling rate or carrier frequency. By fully considering the noise factor, further methods have been developed, such as the correntropy coefficient based algorithm [21]. However, the hand-crafting features may make it difficult to apply to new modulation types in non-cooperative scenarios [22].

With the fast development of deep learning techniques [23], [24], [25], [26], [27], the modulation classification problem has reached a new solution, and some algorithms have been proposed in this direction. In [28], the CNN (Convolutional Neural Network) based modulation classification approach has achieved significant performance improvement compared with the traditional methods, especially for low SNR cases. The deep model in [29] can still have a high classification accuracy when the length of the signal is larger than the designed CNN input length. In [30], [31], the RNN (Recurrent Neural Network) was applied to modulation classification with a satisfactory classification performance achieved. As CNNs lack the time sensitivity of RNNs and RNNs lack the lightness of CNNs, some algorithms combining CNNs and RNNs together were proposed [22], [32], which outperform the CNN based or RNN based algorithms. The STN (Spatial Transformer Network) can be incorporated into an existing CNN architecture, explicitly allowing spatial manipulation of data within the network [26], and it was also applied to modulation classification to improve performance [33], [34], [35].

Processing a long sequence will incur a high computational complexity for deep learning based networks. The Transformer Network (TRN) was first proposed in [36] and widely used for natural language processing [37], which is suitable for a long sequence input. The TRN can be seen as an extension of

This work was supported by the National Natural Science Foundation of China (No.61805189 and 62176199), the Fundamental Research Funds for the Central Universities (No.JB210201), the Natural Science Basic Research Plan in Shaanxi Provincial of China (No.2018JQ6068).

Jingjing Cai and Fengming Gan are with the School of Electronic Engineering, Xidian University, Xi'an, 710071, China (e-mail: jjcai@mail.xidian.edu.cn; fm_gan@stu.xidian.edu.cn).

Xianghai Cao *(Corresponding author)* is with the School of Artificial Intelligence, Xidian University, Xi'an, 710071, China (email: caoxh@xidian.edu.cn).

Wei Liu is with the Department of Electronic and Electrical Engineering, University of Sheffield, Sheffield, S1 3JD, United Kingdom (e-mail: w.liu@sheffeld.ac.uk).

the RNN by removing sequential computation with improved performance [38], [39], [40]. Many efficient TRNs have been proposed, which can process long sequences efficiently and take a small storage space [41], [42], [43]. TRNs were also applied to other research areas [44], [45], [46], such as image classification and the performance is comparable to CNNs [47].

In this paper, the TRN is applied to the modulation classification problem for the first time by constructing a TRN-based model with IQ data input. The TRN has the ability to exploit the correlation of IQ sequences and capture powerful features of the signals, which may lead to improved performance. Simulation results show that the TRN-based model has the best performance on classification accuracy compared with the other four deep models, i.e. CNN, LSTM (Long Short-Term Memory), SCRNN (Sequential Convolutional Recurrent Neural Network) and STN, and two traditional methods, i.e. KNN (K-Nearest Neighbor) and SVM (Support Vector Machine), especially at low SNRs, and the number of parameters of the proposed model is less than that of the other deep models.

The paper is organized as follows. In Section II, the TRN-based model is provided, including motivation and detailed construction of the proposed model. Simulation results are presented in Section III, where the proposed model is compared with the other deep models and traditional methods, and impact of various settings for the proposed model is also studied. Conclusions are drawn in Section IV.

II. The Transformer Network Based Model

A. Motivation

The CNN is one of the most popular networks, which was proposed for image classification and then widely used in many other classification problems. The TRN is proposed for language transformation, which is then gradually used in a few other applications, such as image classification. There is a big difference between these two networks: the TRN is based on the attention mechanism, while the CNN is not. It causes some feature differences between these two networks, which are listed as below:

(1) Correlations between elements of the input sequence play an important role in the TRN, while they are not exploited directly in the CNN.

(2) The TRN establishes a long-distance dependence, so more powerful features can be extracted by incorporating the global information, while the CNN concentrates on the local information, and the contextual information of the input cannot be fully exploited for feature capturing.

(3) The global information can be integrated easily by the TRN without stacking any extra layers, and thus fewer parameters are needed. However, the CNN extracts the global information from the local one by continuously stacking convolutional layers, which dramatically increases the number of parameters of the whole network.

The RNN is also widely used in language transformation, but there is a feature difference between it and the TRN. The RNN uses sequential computation which causes the vanishing gradient problem, while the TRN uses parallel computation, so it can be trained more easily and may be more suitable for practical applications.

The STN is a learnable module, which can be incorporated into an existing model, such as CNN [26]. The feature differences between STN and TRN are presented as follows:

(1) The STN is usually embedded into a CNN, leading to a greater number of training parameters and higher computational complexity. This problem does not exist in TRN as it is an independent network.

(2) The TRN concentrates on temporal relations of the input, while the STN focuses on spatial information of the input. From this point of view, it may be more beneficial to employ the TRN for input signal processing than the STN.

As a result, for signal modulation classification with the IQ sequence input, the TRN may be a more suitable choice, as demonstrated by computer simulations later.

B. Overview of the TRN-based model

The architecture of the TRN-based model is shown in Fig. 1. It can be roughly divided into two parts: preprocessing and TRN. There are mainly three parts in TRN, including the Linear Projection Layer, the Transformer Encoder and the Multi-Layer Perception (MLP) Head.

The process of signal modulation classification based on the TRN-based model can be summarized as follows: the IQ signal is transformed into sequences by the preprocessing stage, which are then embedded into linear sequences in the Linear Projection Layer; an additional learnable "classification token" with position embedding is added before being fed into the Transformer Encoder; the output of the Transformer Encoder is then served as the input of the MLP Head, which consists of several fully connected layers and dropout layers; the output of the MLP Head is the final classification result. Details of the above process are provided in the next three subsections.

These parts play different roles in the process of signal modulation classification. In general, the preprocessing stage processes the IQ data before being fed into the TRN, the Linear Projection Layer projects the data linearly to retain the most discriminating features of the signal, the Transformer Encoder extracts several more powerful features of the signal, and then the MLP Head makes the final decision for modulation classification.

C. IQ data preprocessing

The IQ data of the signal can not be directly applied to the TRN, as the TRN requires several sequences as its input, while the original data of the IQ signals are in two sequences. As the in-phase and quadrature data is sampled in a pair, they should be combined together in a patch to serve as one of the input sequences of the TRN, which makes the feature of the signal clearer. So we divide the long IQ signal sequences into multiple shorter sequences with equal lengths, and then a group of shorter IQ signal sequences is obtained as the input of the TRN. The preprocessing steps are shown in Fig. 2 and detailed steps are listed as follows.

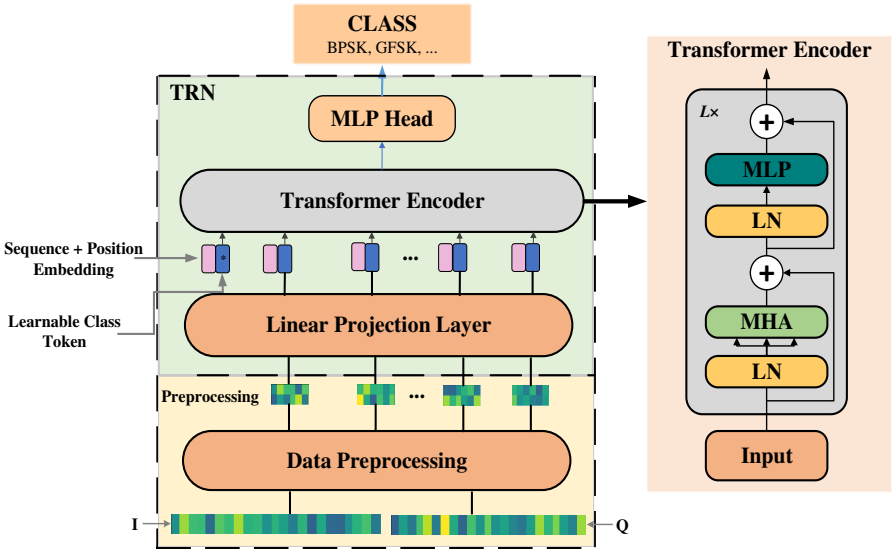


Fig. 1. The architecture of the TRN-based model.

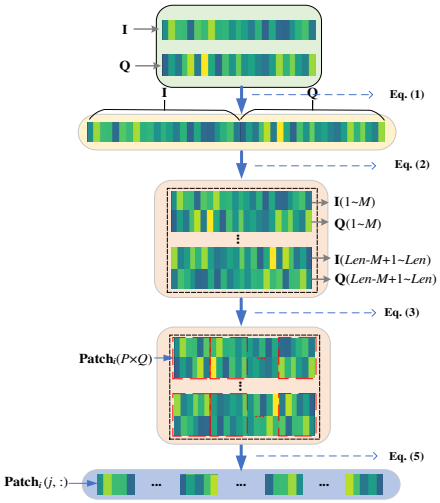


Fig. 2. The process for data preprocessing.

(1) Combine the I and Q signal sequences into one vector $\mathbf{x}$.

The I and Q signal sequences are defined as $\mathbf{x}_I = [\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_{Len}]$ and $\mathbf{x}_Q = [\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{Len}]$, respectively, where Len is the length of the original data, and they are combined as

$$\mathbf{x} = [\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_{Len}, \hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{Len}] \tag{1}$$

(2) Transform $\mathbf{x}$ into an $N \times M$ matrix $\mathbf{R}$.

Suppose $2Len = N \times M$, and the vector $\mathbf{x}$ is restructured into an $N \times M$ matrix $\mathbf{R}$

$$\begin{aligned}\mathbf{R} &= \begin{bmatrix} \bar{x}_1 & \cdots & \bar{x}_M \\ \hat{x}_1 & \cdots & \hat{x}_M \\ \vdots & \ddots & \vdots \\ \bar{x}_{Len-M+1} & \cdots & \bar{x}_{Len} \\ \hat{x}_{Len-M+1} & \cdots & \hat{x}_{Len} \end{bmatrix} \\ &= \begin{bmatrix} r_{1,1} & \cdots & r_{1,M} \\ r_{2,1} & \cdots & r_{2,M} \\ \vdots & \ddots & \vdots \\ r_{N,1} & \cdots & r_{N,M} \end{bmatrix}\end{aligned} \tag{2}$$

(3) Divide $\mathbf{R}$ into Z $P \times Q$ patches $\mathbf{R}_{z_1,z_2}$.

Suppose $Z = NM/PQ$, $Z_1 = N/P$ and $Z_2 = M/Q$ are all integers.

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_{1,1} & \cdots & \mathbf{R}_{1,Z_2} \\ \vdots & \ddots & \vdots \\ \mathbf{R}_{Z_1,1} & \cdots & \mathbf{R}_{Z_1,Z_2} \end{bmatrix} \tag{3}$$

with

$$\begin{aligned}\mathbf{R}_{z_1,z_2} &= \begin{bmatrix} r_{(z_1-1)P+1,(z_2-1)Q+1} & \cdots & r_{(z_1-1)P+1,z_2Q} \\ r_{(z_1-1)P+2,(z_2-1)Q+1} & \cdots & r_{(z_1-1)P+2,z_2Q} \\ \vdots & \ddots & \vdots \\ r_{z_1P,(z_2-1)Q+1} & \cdots & r_{z_1P,z_2Q} \end{bmatrix} \\ z_1 &= 1, \ldots, Z_1, z_2 = 1, \ldots, Z_2\end{aligned} \tag{4}$$

(4) Transform the patches $\mathbf{R}_{z_1,z_2}$ into $1 \times PQ$ sequences $\hat{\mathbf{r}}_{z_1,z_2}$.

$$\hat{\mathbf{r}}_{z_1,z_2} = \left[\mathbf{R}_{z_1,z_2}(1,:), \mathbf{R}_{z_1,z_2}(2,:), \ldots, \mathbf{R}_{z_1,z_2}(P,:) \right] \tag{5}$$

where $\mathbf{R}_{z_1,z_2}(i,:)$ is the i-th row of the patch matrix.

The sequences $\hat{\mathbf{r}}_{z_1,z_2}$ are then used as the input of TRN.

D. The Linear Projection Layer

The Linear Projection Layer is the first part of the TRN and it is the link between the preprocessed data and the Transformer Encoder. The processing steps are listed as follows [47].

(1) Linearly project the sequences $\hat{\mathbf{r}}_{z_1,z_2}$ and construct the matrix $\dot{\mathbf{R}}$.

Suppose $\mathbf{E}$ is an embedding projection matrix with dimension $PQ \times D$, where D is the size of the vector used in the Transformer Encoder. The $1 \times D$ vector $\bar{\mathbf{r}}_{z_1,z_2}$ is defined as

$$\bar{\mathbf{r}}_{z_1,z_2} = \hat{\mathbf{r}}_{z_1,z_2}\mathbf{E} \tag{6}$$

Then, the matrix $\dot{\mathbf{R}}$ is constructed by combining all the sequences $\bar{\mathbf{r}}_{z_1,z_2}$ as

$$\begin{aligned}\dot{\mathbf{R}} &= [\bar{\mathbf{r}}_{1,1}; \ldots; \bar{\mathbf{r}}_{1,Z_2}; \ldots; \bar{\mathbf{r}}_{Z_1,1} \ldots; \bar{\mathbf{r}}_{Z_1,Z_2}] \\ &= [\dot{\mathbf{r}}_1; \ldots; \dot{\mathbf{r}}_{(z_1-1)Z_2+z_2}; \ldots; \dot{\mathbf{r}}_{Z_1Z_2}]\end{aligned} \tag{7}$$

(2) Construct $\hat{\mathbf{R}}$ by adding the learnable class token vector $\mathbf{c}$ to $\dot{\mathbf{R}}$.

The vector $\mathbf{c} = [c_1, c_2, \ldots, c_D]$ is firstly initialized randomly and then updated during the training process. The matrix $\hat{\mathbf{R}}$ with dimension $(Z_1Z_2 + 1) \times D$ is constructed as

$$\hat{\mathbf{R}} = [\mathbf{c}; \dot{\mathbf{R}}] \tag{8}$$

(3) Construct the matrix $\breve{\mathbf{R}}$ by adding a learned position encoding matrix $\mathbf{E}_{pos}$ to the sequence $\hat{\mathbf{R}}$.

Suppose $\mathbf{E}_{pos}$ is a $(Z_1Z_2 + 1) \times D$ matrix and then $\breve{\mathbf{R}}$ is given by

$$\breve{\mathbf{R}} = \hat{\mathbf{R}} + \mathbf{E}_{pos} \tag{9}$$

$\breve{\mathbf{R}}$ is the input of the Transformer Encoder.

E. The Transformer Encoder

The Transformer Encoder consists of a stack of L same layers, and each layer is mainly composed of an MLP block and a Multihead Attention (MHA). The MLP is a simple fully connected feed-forward network, while the MHA is more complicated. The Layernorm (LN) is applied to residual connections after every block, which can mitigate the vanishing gradient problem due to a deeper depth of the neural network.

The structure of the MHA is shown in Fig. 3. The Scaled Dot-Production attention, called attention for short in the following, is the most important part of MHA. The attention can be described as mapping a query and a set of key-value pairs to an output and linking different positions of a single sequence, which integrates information across the entire input data.

Denote the number of times of performing the attention operation in the MHA as H_o, and the process for attention operation can be described as follows [36].

(1) Embed the input $\breve{\mathbf{R}}$ of the Transformer Encoder by the $D \times d_{model}$ embedding matrix $\mathbf{W}$.

$$\bar{\mathbf{Q}} = \bar{\mathbf{K}} = \bar{\mathbf{V}} = \breve{\mathbf{R}}\mathbf{W} \tag{10}$$

where $\bar{\mathbf{Q}}$, $\bar{\mathbf{K}}$ and $\bar{\mathbf{V}}$ are the resultant matrices.

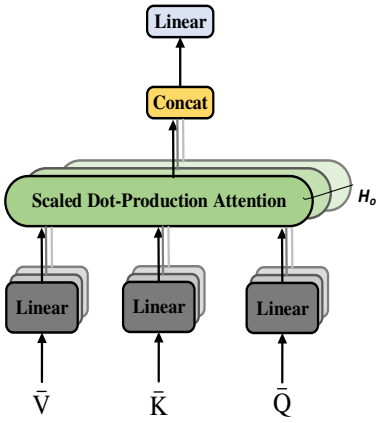


Fig. 3. Structure of the Multihead Attention.

(2) Calculate the query matrix $\mathbf{Q}_i$, the key matrix $\mathbf{K}_i$ and the value matrix $\mathbf{V}_i$, separately.

$$\mathbf{Q}_i = \bar{\mathbf{Q}}\mathbf{W}_i^Q,\ \mathbf{K}_i = \bar{\mathbf{K}}\mathbf{W}_i^K,\ \mathbf{V}_i = \bar{\mathbf{V}}\mathbf{W}_i^V \tag{11}$$

where $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$, and $\mathbf{W}_i^V$, $i = 1, 2, 3, \ldots, H_o$ are the matrices with dimensions $d_{model} \times d_k$, $d_{model} \times d_k$ and $d_{model} \times d_v$, separately.

(3) Calculate $\mathbf{head}_i$ by the operation $\mathrm{Attention}(\cdot)$ as follows

$$\mathbf{head}_\mathrm{i} = \mathrm{Attention}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i) \tag{12}$$

with

$$\mathrm{Attention}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i) = \mathrm{softmax}\left(\frac{\mathbf{Q}_i\mathbf{K}_i^\mathrm{T}}{\sqrt{d_k}}\right)\mathbf{V}_i \tag{13}$$

(4) Construct the matrix $\mathbf{Head}$ by multiplying the vector $\mathbf{head}_i$ with the matrix $\mathbf{W}^0$

Suppose $\mathbf{W}^0$ is an $H_o d_v \times d_{model}$ matrix. we have

$$\mathbf{Head} = [\mathbf{head}_1, \ldots, \mathbf{head}_{H_o}]\mathbf{W}^0\ . \tag{14}$$

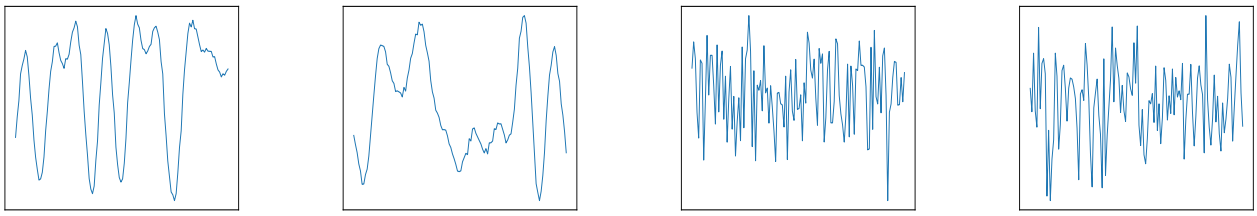
(5) Obtain the output $\bar{\mathbf{Z}}$ of the MHA by adding the matrix $\breve{\mathbf{R}}$ with $\mathbf{Head}$.

$$\bar{\mathbf{Z}} = \mathbf{Head} + \breve{\mathbf{R}} \tag{15}$$

The output of the Transformer Encoder is obtained by inputting $\bar{\mathbf{Z}}$ into the MLP block. If there are more than one layers in the Transformer Encoder, consider the output of the former layer as the input of the next and follow steps described above. Then, choose the first row of the final output of the Transformer Encoder as the input of the MLP Head, and the output of the MLP Head is the final classification result.

III. Simulations and Analyses

In this section, the classification performance of the TRN-based model is compared with the other four deep models and two traditional methods, which are CNN-based model, LSTM-based model, SCRNN-based model, STN-based model, KNN and SVM, respectively. Three benchmark datasets used for the simulations are introduced firstly, and the impact of different settings of the parameters for the TRN-based



(a) QPSK (b) 8PSK (c) WBFM (d) AM-DSB

Fig. 4. The in-phase signal samples of the four modulation types at 10dB SNR.

model is studied to find the best set of values. Then, some implementation details and the classification results of the deep models and traditional methods on the three benchmark datasets are presented.

A. Datasets

Three datasets are used in our simulations, RML2016.04C and RML2016.10 and RML2018.01a, which are generated by GNU Radio [48], [49]. RML2016.10a is an upgraded version of RML2016.04C, which considers more effects of the real electromagnetic environment. RML2018.01a is a more robust dataset with a larger amount of data than the other two datasets.

The parameters of these three datasets are listed in Table I and in-phase signal samples of the four modulation types are presented in Fig. 4. It can be seen that the modulations QPSK and 8PSK are easy to be confused, while WBFM and AM-DSB also look similar to each other.

TABLE I
PARAMETERS OF THE BENCHMARK DATASETS.

Dataset	RML2016.04C, RML2016.10a, RML2018.01a
Modulations	8 Digital Modulations: BPSK, QPSK, 8PSK, 16 QAM, 64 QAM, GFSK, CPFSK, and PAM4 3 Analog Modulations: WBFM, AM-SSB, and AM-DSB (RML2016.04C, RML2016.10a) 19 Digital Modulations: OOK, 4ASK, 8ASK, BPSK, QPSK, 8PSK, 16PSK, 32PSK, 16APSK, 32APSK, 64APSK, 128APSK, 16QAM, 32QAM, 64QAM, 128QAM, 256QAM, OQPSK, GMSK 5 Analog Modulations: AM-SSB-WC, FM AM-SSB-SC, AM-DSB-WC, AM-DSB-SC (RML2018.01a)
Signal format	In-phase and quadrature (IQ)
Signal dimension	2×128 per sample (RML2016.04C, RML2016.10a) 2×1024 per sample (RML2018.01a)
SNR range	-20dB:2dB:18dB (RML2016.04C, RML2016.10a) -20dB:2dB:30dB (RML2018.01a)
Total number of samples	162060(RML2016.04C) 220000(RML2016.10a) 2555904(RML2018.01a)
Extra added effects for RML2016.10a and RML2018.01a	Selective fading Sample rate offset (SRO) Center frequency offset (CFO) Noise (AWGN)

The categorical cross entropy is adopted as the loss function, which can be written as:

$$F_{loss} = -\frac{1}{N_{batch}} \sum_{i=1}^{N_{batch}} \mathbf{y_i} \cdot \log\left(\hat{\mathbf{y}}_\mathbf{i}\right). \tag{16}$$

where $\mathbf{y_i}$ represents the ground truth in the form of one-hot encoding, $\hat{\mathbf{y}}_\mathbf{i}$ is the prediction and N_{batch} is the training batch size set as 32.

At each SNR, there are two sample sets, the training set (90%) and the test set (10%). The Adam optimizer with a learning rate of 0.01 is utilized. All trainings and predictions are implemented in Tensorflow [50].

B. Comparisons with deep models and traditional methods

Four deep models and two traditional methods are used to compare with the TRN-based model for signal modulation classification, and their settings are introduced briefly one by one in the following. For all the deep models, the softmax activation function is used in the last layer, and the Adam optimizer and sparse entropy loss function are applied in the deep models.

The first one is the CNN-based model [28]. It contains four layers, where two layers are convolutional layers and the other two are dense layers. Each hidden layer utilizes the Rectified Linear Unit (ReLU) activation function.

The second one is the LSTM-based model [31]. Two parts are included in this model: the first part contains two 128-unit LSTM layers while the last part is a 11-unit dense layer. The first LSTM layer returns the full sequences while the second one returns the last state.

The third one is the SCRNN-based model [22]. It combines the speed and lightness of the CNN and temporal sensitivity of RNN, and can be divided into three parts: the first part contains two convolutional layers with 128 filters and ReLU activation functions, the second part are two 128-unit LSTM layers with ReLU activation functions, and the last part is a dense layer.

The fourth one is the STN-based model [34], which contains two parts. The first one is STN, which is composed of Localisation Network, Grid Generator and Sampler. The last one is CNN which is composed of two convolutional neural layers, each layer followed by a max pooling layer, and two dense layers at the final stage.

The fifth one is the SVM [51]. It maps the input data into a high-dimensional feature space, where a linear decision

surface is constructed. Different types of classes are located on different sides of the decision surface.

The last one is the KNN [52], which follows the nearest decision rule, and assigns an unclassified sample point to the nearest classified set.

C. Parameter analyses

As the classification accuracy varies with the parameter settings, it is necessary to fine-tune the TRN-based model. The impact of the parameters of the TRN-based model on the classification performance is investigated below, including patch size, the MLP size, the layer number, the attention number, the batch size, the training epoch, the test set size and the optimizer.

The patch size related result is shown in Fig. 5. It can be seen that the classification accuracy is the highest when the patch size is 4. The classification performance of the TRN is sensitive to the length of the sequence, and an inappropriate length will cause position embedding and position information confusion, which may lead to unsatisfactory classification performance. For the MLP size, as shown in Fig. 6, the classification accuracy is the highest when the MLP size is 256. When the MLP size is smaller than 256, the generalization ability of the TRN is poorer, which leads to worse classification performance. However, with the MLP size greater than 256, the classification accuracy drops due to the overfitting problem.

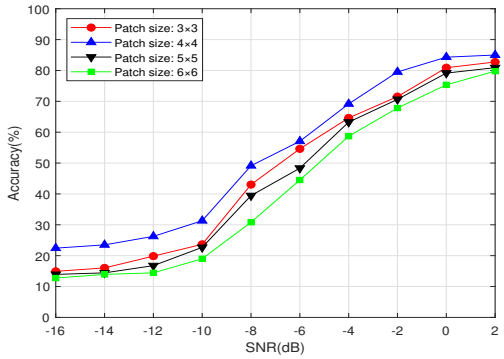


Fig. 5. Classification performance of the TRN-based model with a varying patch size.

The layer number and attention number related results are given in Fig. 7a and Fig. 7b. It can be seen that the best result is achieved when the layer number is 5 and the attention number is 16. The depth of the proposed model increases as the number of layers and attentions increases, and the attention distance increases as the depth of the proposed model increases. The attention distance of the TRN is similar to the size of the receptive field of the CNN, and the larger the attention distance the stronger its ability to extract features. However, if these parameters are too large, the classification accuracy may drop due to the overfitting problem.

The batch size result in Fig. 8 indicates that the classification performance is the best when the batch size is 32. Normally, the greater the batch size the better the performance, but

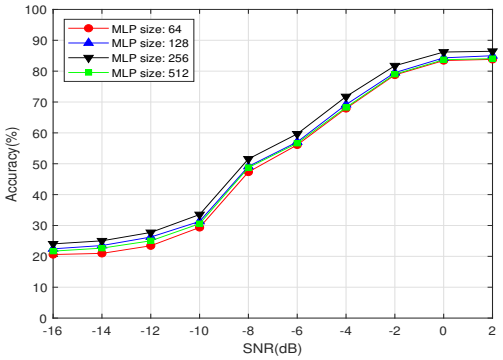
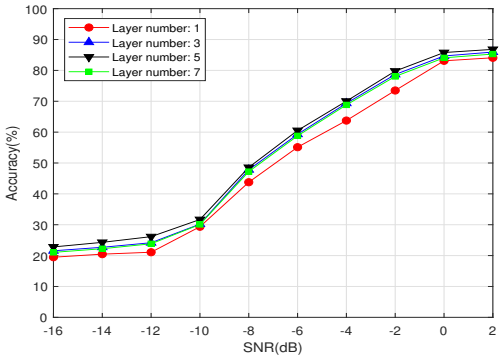
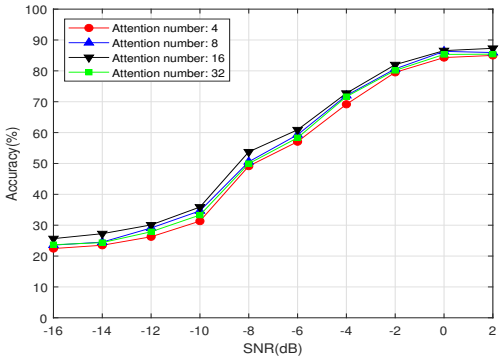


Fig. 6. Classification performance of the TRN-based model with a varying MLP size.



(a)



(b)

Fig. 7. Classification performance of the TRN-based model with a varying (a) layer number, (b) attention number.

the performance will degrade when the batch size exceeds a threshold value [53].

The impact of the training epoch is demonstrated in Fig. 9, where it can be seen that the epoch 15 provides the best performance. As the loss is reduced after each epoch, if the epoch is too small, it may lead to an unsatisfactory classification performance. Furthermore, the model will converge if the epoch exceeds a threshold value.

Splitting the dataset is also an important step before training. The test set size directly affects the classification accuracy of the model and a proper test set size promotes the model training. The impact of the test set size is shown in Fig. 10

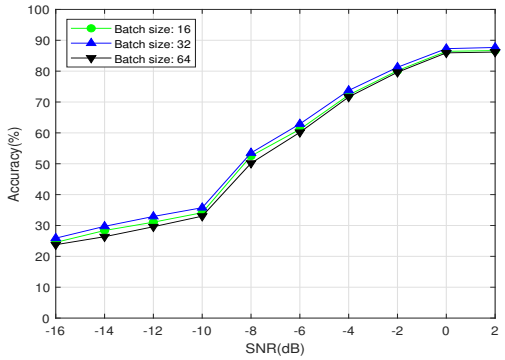


Fig. 8. Classification performance of the TRN-based model with a varying batch size.

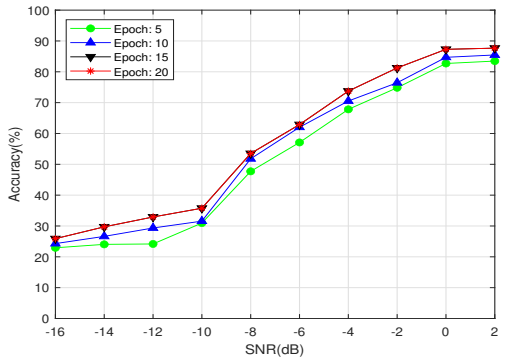


Fig. 9. Classification performance of the TRN-based model with a varying training epoch.

and it can be seen that the best performance corresponds to a test set size of 10%. If the test set size is too small and the training set size is too big, the model may suffer from the overfitting problem; otherwise, it may lead to the underfitting problem.

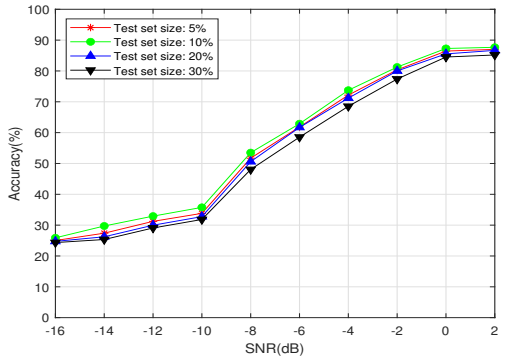


Fig. 10. Classification performance of the TRN-based model with a varying test set size.

The optimizer plays an important role in the training process. An appropriate optimizer makes the network converge quickly. There are some popular optimizers, such as Adam, SGD, AdaGrad and so on. The classification accuracies with different optimizers are provided in Fig. 11. It can be seen that the Adam optimizer is the best choice.

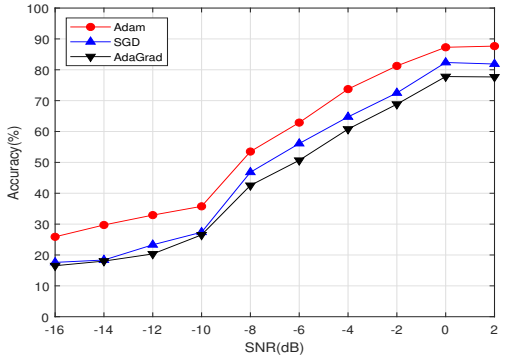


Fig. 11. Classification performance of the TRN-based model with a varying optimizers.

D. Performance comparisons

The data with SNR ranging from −16dB to 10dB is chosen from the datasets RML2016.04C RML2016.10a and RML2018.01a, separately. The settings of the TRN-based model are provided in Table. II. The parameters of the four compared deep models and the two traditional methods are all set for their best performance. The classification results are presented in Figs. 12, 13 and 14.

TABLE II
PARAMETERS FOR THE TRN-BASED MODEL.

Parameters	Value
Patch size	4×4
MLP size	256
Layer number	5
Attention number	16
Batch size	32
Training epoch	15
Test set size	10%
Optimizer	Adam

The classification performance of the deep models and traditional methods based on the first dataset is presented in Fig. 12. It can be seen that the proposed model performs the best in the whole SNR range. Compared with the CNN-based and STN-based models, the proposed one outperforms them by nearly 15% and 10% at lower SNRs, from −16dB to −12dB, respectively. The STN-based model outperforms the CNN-based one, which implies that by making the model spatially-invariant, robustness of the model is enhanced against various adverse effects. Furthermore, the proposed model has better performance than those of LSTM-based and SCRNN-based models, implying that more powerful features can be extracted by the TRN-based model. Besides, The classification performance of the STN-based and CNN-based models are close to that of the TRN-based one with SNR ranging from 0dB to 10dB, but there is a gap of about 5% and 6% between the SCRNN-based and LSTM-based models and the TRN-based model, respectively. Compared with the traditional

method SVM, the proposed model outperforms it by nearly 10% at lower SNRs and 5% at higher SNRs, while KNN has the worst performance.

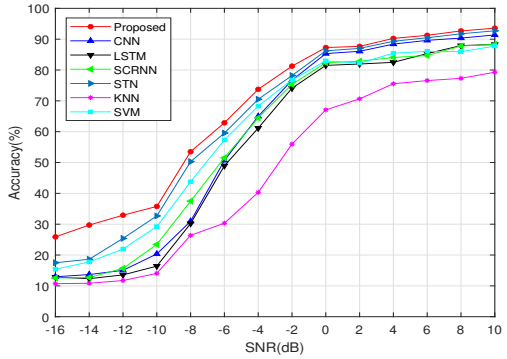


Fig. 12. Classification performance of the deep models and the traditional methods based on RML2016.04C.

The classification performance based on the second dataset is presented in Fig. 13. It can be seen that the proposed model still performs the best. The TRN-based model outperforms the STN-based and CNN-based ones by nearly 2% and 4% in the whole SNR range, respectively. The STN-based model still has a better classification performance than that of the CNN-based model. Furthermore, the proposed model outperforms the LSTM-based and SCRNN-based models by nearly 2% at lower SNRs and 10% at higher SNRs, and the performance of the TRN-based model is better than that of the SVM by nearly 5% at lower SNRs and 15% at higher SNRs. The KNN again has the worst performance in the whole SNR range. Generally, the classification performance based on RML2016.10a drops a lot compared to that based on RML2016.04C. As the signals of the dataset RML2016.10a are affected by several different electromagnetic environments, while the signals of the RML2016.04C are generated in a single electromagnetic environment, the training based on RML2016.10a is harder than that of RML2016.04C, leading to performance deterioration in the former case [54].

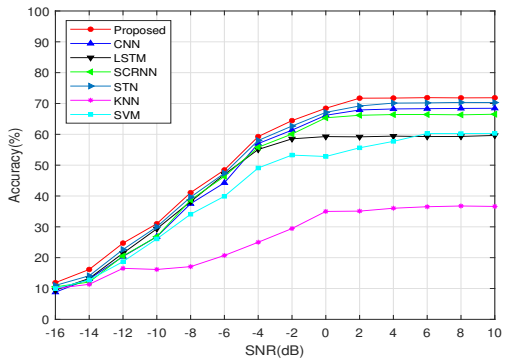


Fig. 13. Classification performance of the deep models and the traditional methods based on RML2016.10a.

The classification performance based on the last dataset is shown in Fig. 14. It can be seen that only TRN-based, SCRNN-based and STN-based models maintain a good performance, while the performance of the other two deep models and two traditional methods drops a lot compared to that based on the first and second datasets. The TRN-based model still performs the best in the whole SNR range, which outperforms the SCRNN-based and STN-based models by nearly 1% at lower SNRs and performs almost the same at higher SNRs. For the CNN-based model, it tends to be overfitting on the training set for its simple network structure and the huge number of samples of the new dataset. For the SVM, both the total number of samples and the dimension of the signal increase, leading to a huge amount of computation in training, and its performance gets worse.

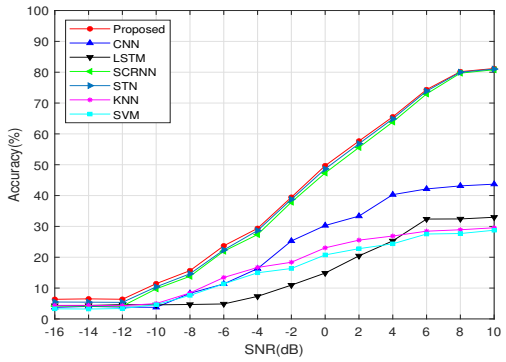


Fig. 14. Classification performance of the deep models and the traditional methods based on RML2018.01a.

It can be seen from Figs. 12, 13 and 14 that the performance of the proposed TRN-based model is similar to that of the STN-based one in most cases under the relevant datasets, and the TRN-based model is superior to the STN-based one for the range of SNR equal to or lower than -10dB in Fig. 12.

Moreover, as shown in Table III, the training parameter number of the TRN-based model is smaller than that of the other deep models. There are fewer training parameters in every layer of the LSTM-based model, so the total number of training parameters is small. For the TRN-based model, only a few layers are needed to extract the features due to the existence of attention mechanism, so the number of training parameters of the proposed model is also small. Although the LSTM-based and TRN-based models have a low number of training parameters, the performance of the TRN-based model outperforms that of the LSTM-based model.

TABLE III
COMPARISON OF TRAINING PARAMETERS BETWEEN FOUR DEEP MODELS AND THE PROPOSED MODEL.

Deep Models	CNN	LSTM	SCRNN	STN	TRN
Number of parameters	3813467	271755	429195	1197033	264587

In deep learning, the confusion matrix is a visual tool to compare the predicted results with the true values. All the classification results for all the classes are displayed in a confusion matrix, where each column represents the predicted modulation class and each row represents the real modulation class. The numerical value on each grid denotes the prediction

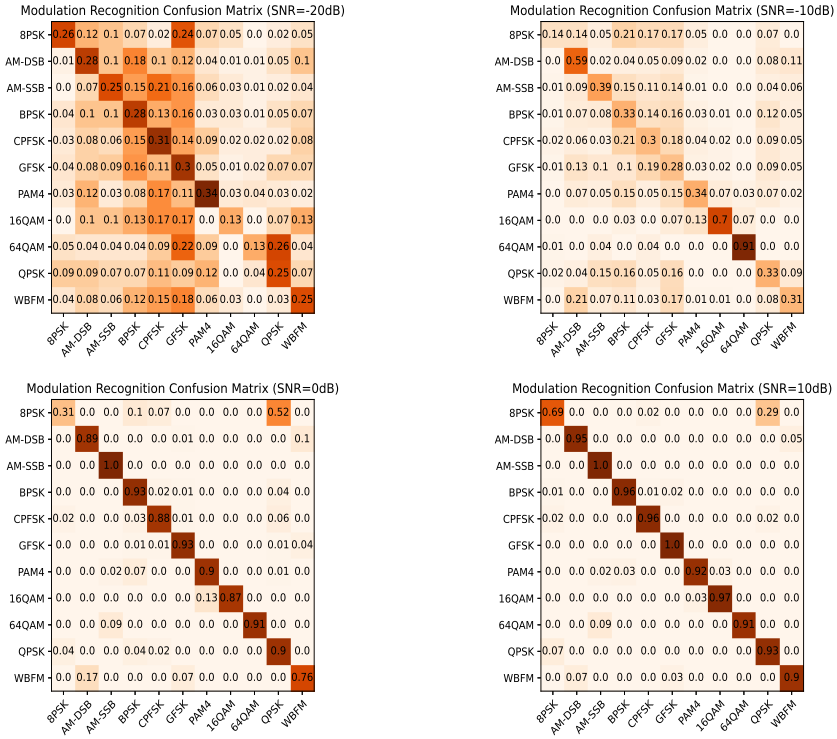


Fig. 15. Confusion matrices of the optimized TRN-based model on the dataset RML2016.04C at different SNRs.

probability of the corresponding modulation class. Confusion matrices of the optimal TRN-based model at various SNRs are presented in Fig. 15. It can be seen that the diagonals become sharper with an increasing SNR, which illustrates that the higher the SNR the better the classification accuracy. However, the confusion between 8PSK and QPSK always exists even at high SNRs. The phases of QPSK is the subset of those of 8PSK, so the curves of their signals are sometimes identical. When the signals are under unideal channel conditions, it becomes harder to distinguish the modulation features of 8PSK and QPSK.

IV. Conclusion

In this paper, the TRN-based model has been constructed and applied to the automatic modulation classification problem successfully for the first time. As demonstrated by simulation results, the proposed model outperforms the other three deep models (LSTM, CNN and SCRNN) and the two traditional methods (KNN and SVM) in terms of classification accuracy, and the number of training parameters of the proposed model is less than the other deep models. In comparison with STN, the proposed model performs at least as well as STN under most relevant datasets, but it is characterized with a smaller number of training parameters and a lower overall computation complexity. The impact on performance of different parameter settings was also studied to find the best configuration for the proposed model. Possible future work can focus on applying various improved TRNs to modulation classification, which may further improve the performance. The signals used in the simulations now are under a single type of channel condition, so it is essential to do more simulations under different types of channel conditions to have a better training result. Furthermore, the robustness of the model may degrade due to lack of essential labels, and thus semi-supervised or unsupervised methods could be further investigated.

References

[1] T. Yucek and H. Arslan, "A survey of spectrum sensing algorithms for cognitive radio applications," *IEEE communications surveys & tutorials*, vol. 11, no. 1, pp. 116–130, 2009.

[2] M. Höyhtyä, A. Mämmelä, M. Eskola, M. Matinmikko, J. Kalliovaara, J. Ojaniemi, J. Suutala, R. Ekman, R. Bacchus, and D. Roberson, "Spectrum occupancy measurements: A survey and use of interference maps," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2386–2414, 2016.

[3] S.-Z. Hsue and S. S. Soliman, "Automatic modulation recognition of digitally modulated signals," in *IEEE Military Communications Conference,'Bridging the Gap. Interoperability, Survivability, Security'*. IEEE, 1989, pp. 645–649.

[4] C. Clancy, J. Hecker, E. Stuntebeck, and T. O'Shea, "Applications of machine learning to cognitive radio networks," *IEEE Wireless Communications*, vol. 14, no. 4, pp. 47–52, 2007.

[5] C. Weber, M. Peter, and T. Felhauer, "Automatic modulation classification technique for radio monitoring," *Electronics Letters*, vol. 51, no. 10, pp. 794–796, 2015.

[6] P. Qi, X. Zhou, S. Zheng, and Z. Li, "Automatic modulation classification based on deep residual networks with multimodal information," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 1, pp. 21–33, 2020.

[7] L. Huang, Y. Zhang, W. Pan, J. Chen, L. P. Qian, and Y. Wu, "Visualizing deep learning-based radio modulation classifier," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 1, pp. 47–58, 2020.

[8] W. Wei and J. M. Mendel, "Maximum-likelihood classification for digital amplitude-phase modulations," *IEEE transactions on Communications*, vol. 48, no. 2, pp. 189–193, 2000.

[9] 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*. IEEE, 1988, pp. 431–436.

[10] A. K. Nandi and E. E. Azzouz, "Algorithms for automatic modulation recognition of communication signals," *IEEE Transactions on communications*, vol. 46, no. 4, pp. 431–436, 1998.

[11] A. Polydoros and K. Kim, "On the detection and classification of quadrature digital modulations in broad-band noise," *IEEE Transactions on Communications*, vol. 38, no. 8, pp. 1199–1211, 1990.

[12] M. D. Wong, S. K. Ting, and A. K. Nandi, "Naïve Bayes classification of adaptive broadband wireless modulation schemes with higher order cumulants," in *2008 2nd International Conference on Signal Processing and Communication Systems*. IEEE, 2008, pp. 1–5.

[13] M. P. DeSimio and G. E. Prescott, "Adaptive generation of decision functions for classification of digitally modulated signals," in *Proceedings of the IEEE 1988 National Aerospace and Electronics Conference*. IEEE, 1988, pp. 1010–1014.

[14] H.-C. Wu, M. Saquib, and Z. Yun, "Novel automatic modulation classification using cumulant features for communications via multipath channels," *IEEE Transactions on Wireless Communications*, vol. 7, no. 8, pp. 3098–3105, 2008.

[15] J. Lopatka and M. Pedzisz, "Automatic modulation classification using statistical moments and a fuzzy classifier," in *WCC 2000-ICSP 2000. 2000 5th International Conference on Signal Processing Proceedings. 16th World Computer Congress 2000*, vol. 3. IEEE, 2000, pp. 1500–1506.

[16] A. Swami and B. M. Sadler, "Hierarchical digital modulation classification using cumulants," *IEEE Transactions on communications*, vol. 48, no. 3, pp. 416–429, 2000.

[17] M. W. Aslam, Z. Zhu, and A. K. Nandi, "Automatic modulation classification using combination of genetic programming and KNN," *IEEE Transactions on wireless communications*, vol. 11, no. 8, pp. 2742–2750, 2012.

[18] O. A. Dobre, Y. Bar-Ness, and W. Su, "Higher-order cyclic cumulants for high order modulation classification," in *IEEE Military Communications Conference, 2003. MILCOM 2003.*, vol. 1. IEEE, 2003, pp. 112–117.

[19] V. D. Orlic and M. L. Dukic, "Automatic modulation classification algorithm using higher-order cumulants under real-world channel conditions," *IEEE Communications Letters*, vol. 13, no. 12, pp. 917–919, 2009.

[20] K. Maliatsos, S. Vassaki, and P. Constantinou, "Interclass and intraclass modulation recognition using the wavelet transform," in *2007 IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications*. IEEE, 2007, pp. 1–5.

[21] A. I. Fontes, L. A. Pasa, V. A. de Sousa Jr, F. M. Abinader Jr, J. A. Costa, and L. F. Silveira, "Automatic modulation classification using information theoretic similarity measures," in *2012 IEEE vehicular technology conference (VTC fall)*. IEEE, 2012, pp. 1–5.

[22] K. Liao, Y. Zhao, J. Gu, Y. Zhang, and Y. Zhong, "Sequential convolutional recurrent neural networks for fast automatic modulation classification," *IEEE Access*, vol. 9, pp. 27 182–27 188, 2021.

[23] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[25] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[26] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, "Spatial transformer networks," *Advances in neural information processing systems*, vol. 28, pp. 2017–2025, 2015.

[27] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *International conference on machine learning*. PMLR, 2016, pp. 173–182.

[28] T. J. OShea, J. Corgan, and T. C. Clancy, "Convolutional radio modulation recognition networks," in *International conference on engineering applications of neural networks*. Springer, 2016, pp. 213–226.

[29] S. Zheng, P. Qi, S. Chen, and X. Yang, "Fusion methods for CNN-based automatic modulation classification," *IEEE Access*, vol. 7, pp. 66 496–66 504, 2019.

[30] D. Hong, Z. Zhang, and X. Xu, "Automatic modulation classification using recurrent neural networks," in *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*. IEEE, 2017, pp. 695–700.

[31] 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 Transactions on Cognitive Communications and Networking*, vol. 4, no. 3, pp. 433–445, 2018.

[32] N. E. West and T. O'Shea, "Deep architectures for modulation recognition," in *2017 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*. IEEE, 2017, pp. 1–6.

[33] T. J. O'Shea, L. Pemula, D. Batra, and T. C. Clancy, "Radio transformer networks: Attention models for learning to synchronize in wireless systems," in *2016 50th Asilomar Conference on Signals, Systems and Computers*. IEEE, 2016, pp. 662–666.

[34] M. Mirmohammadsadeghi, S. S. Hanna, and D. Cabric, "Modulation classification using convolutional neural networks and spatial transformer networks," in *2017 51st Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2017, pp. 936–939.

[35] M. Li, O. Li, G. Liu, and C. Zhang, "An automatic modulation recognition method with low parameter estimation dependence based on spatial transformer networks," *Applied Sciences*, vol. 9, no. 5, p. 1010, 2019.

[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[37] K. Ahmed, N. S. Keskar, and R. Socher, "Weighted transformer network for machine translation," *arXiv preprint arXiv:1711.02132*, 2017.

[38] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model." in *Interspeech*, vol. 2, no. 3. Makuhari, 2010, pp. 1045–1048.

[39] T. Mikolov, S. Kombrink, L. Burget, J. Černockỳ, and S. Khudanpur, "Extensions of recurrent neural network language model," in *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2011, pp. 5528–5531.

[40] I. Sutskever, J. Martens, and G. E. Hinton, "Generating text with recurrent neural networks," in *ICML*, 2011.

[41] N. Kitaev, Ł. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," *arXiv preprint arXiv:2001.04451*, 2020.

[42] J. W. Rae, A. Potapenko, S. M. Jayakumar, and T. P. Lillicrap, "Compressive transformers for long-range sequence modelling," *arXiv preprint arXiv:1911.05507*, 2019.

[43] Z. Wang, Y. Ma, Z. Liu, and J. Tang, "R-transformer: Recurrent neural network enhanced transformer," *arXiv preprint arXiv:1907.05572*, 2019.

[44] X. Chu, Z. Tian, Y. Wang, B. Zhang, H. Ren, X. Wei, H. Xia, and C. Shen, "Twins: Revisiting the design of spatial attention in vision transformers," *arXiv preprint arXiv:2104.13840*, vol. 1, no. 2, p. 3, 2021.

[45] Y. Sha, Y. Zhang, X. Ji, and L. Hu, "Transformer-Unet: Raw Image Processing with Unet," *arXiv preprint arXiv:2109.08417*, 2021.

[46] Y. Wang, X. Zhang, T. Yang, and J. Sun, "Anchor DETR: Query Design for Transformer-Based Detector," *arXiv preprint arXiv:2109.07107*, 2021.

[47] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[48] T. J. O'shea and N. West, "Radio machine learning dataset generation with gnu radio," in *Proceedings of the GNU Radio Conference*, vol. 1, no. 1, 2016.

[49] 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, 2018.

[50] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283.

[51] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[52] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.

[53] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," *arXiv preprint arXiv:1609.04836*, 2016.

[54] E. Perenda, S. Rajendran, G. Bovet, S. Pollin, and M. Zheleva, "Learning the unknown: Improving modulation classification performance in unseen scenarios," in *IEEE Infocom*. Institute of Electrical and Electronics Engineers, 2020.