electronics

MDPI

Article

Adaptive Modem Based on LSTM-AutoEncoder with Vector Quantization

Weijie Gao [1,2], Shijun Xie [2,*], Heng Wang [2], Yufeng Zhang [2] and Yao Ling [1,2]

[1] School of Electronics and Information Engineering, Nanjing University of Information Science and Technology, Nanjing 210044, China
[2] 63rd Research Institute, National University of Defense Technology, Nanjing 210007, China
* Correspondence: xie_sj@nudt.edu.cn

Abstract: Recently, researchers have achieved the goal of using unified architecture to achieve multiple modulation modes under specific conditions. However, existing research still suffers from the problem of large resource and time overhead. This paper proposes an adaptive modem based on a vector quantization (VQ) long short-term memory autoencoder (LSTM-AE), designed to implement modulation and demodulation of signals from the second to thirty-second order in a lightweight way. By leveraging the memory capacity of the LSTM module and the compression capability of the autoencoder, the model is able to support multi-order modulation methods. This study used the Adam optimizer for training and testing on a simple dataset extended by adding AWGN noise only and made modifications based on the MSE loss function to balance the accuracy and training speed of each part of the model. Experiments demonstrate that the proposed method not only achieves comparable modem performance to existing frameworks for second to thirty-second order signals, but also significantly reduces the number of parameters and training time. Experimental results indicate that the proposed methodology not only matches the performance of existing frameworks for signals ranging from the second to the thirty-second order, but also employs merely 79.6% of the average parameter count and a mere 7.4% of the average training duration. This represents a substantial reduction in resource expenditure.

Keywords: adaptive modem; autoencoder; vector quantization

check for updates

Citation: Gao, W.; Xie, S.; Wang, H.; Zhang, Y.; Ling, Y. Adaptive Modem Based on LSTM-AutoEncoder with Vector Quantization. *Electronics* **2024**, *13*, 3124. https://doi.org/10.3390/electronics13163124

Academic Editor: Adão Silva

Received: 7 July 2024
Revised: 1 August 2024
Accepted: 1 August 2024
Published: 7 August 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

As 2019 becomes the first year of commercial 5G mobile network applications, and emerging technologies are progressively assimilated into a single international standard [1], wireless communication research is rapidly progressing in the direction described by the ITU Network 2030 Working Group [2] and is constantly evolving towards AI-driven and intelligent directions [3]. In modern communication systems, modulation and demodulation are crucial links, and efficient and stable modulation methods still maintain high value [4–6]. However, the continuous advancement of communication countermeasure technology has put forward higher and higher requirements for the rapid generation and switching of waveforms. Therefore, it is a very valuable task to study the fast and low resource consumption implementation of adaptive modulation and demodulation methods.

Classic modulation and demodulation methods mainly rely on mathematical models designed by experts, such as QAM (Quadrature Amplitude Modulation) and PSK (Phase Shift Keying). These methods can only be used individually, and there is no unified architecture that can implement multiple modes. Adaptive coding and modulation [7] has played a historic role in solving this problem. By changing parameters to adapt to the channel environment, adaptive coding and modulation implements a variety of modulation and demodulation methods with different rates and orders and seeks to achieve an acceptable balance between communication reliability and effectiveness. However, traditional ACM

has gradually failed to meet the intelligence requirements of communication systems, and algorithms based on neural networks [8–12] have received more attention and research. Deep learning, as a powerful artificial intelligence technology, has achieved remarkable results in many fields, surpassing traditional methods such as image recognition and natural language processing. In recent years, researchers have also begun to explore the application of deep learning to the modulation and demodulation process of communication systems and have achieved the goal of using a unified architecture to implement multiple modulation methods under specific conditions. After testing, various models have shown advantages over traditional methods in some orders, such as achieving similar performance to 2PSK [8,9] and the same or better performance to 4PSK [8,10]. However, the cost of this versatility and adaptability is the consumption of a large amount of software and hardware resources. In practice, the calculation is complex and requires special hardware or DSP software to work.

Finding a balance between system performance and resource consumption is crucial. This paper introduces a new deep learning framework for multi-order modulation and demodulation that seamlessly integrates the LSTM-AE model for anomaly detection with vector quantization technology to innovatively generate learned modulation signals while optimizing cost efficiency. This integrated optimization technology will play an increasingly important role in resource-constrained contexts such as military satellite communications. The main contributions of this paper are as follows:

1. An adaptive modem based on LSTM-AE was developed. LSTM-AE was applied to communication modulation and demodulation and combined with vector quantization technology to achieve the purpose of lightweight adaptive modulation and demodulation. The loss function, optimization algorithm, and training method were given. Compared with other models, it achieved equal or better performance with a lower number of parameters and improved the model's ability to reconstruct representation.
2. Based on Pytorch, VQ-LSTM-AE was trained and tested, and the constellation diagram, symbol mapping relationship, and demodulation performance of second to thirty-second-order modems were given. The computational overhead of the proposed DL model was compared and evaluated, and it was verified that the method proposed in this paper can achieve the same or a lower bit error rate with lower resource overhead compared with baseline model.

We used BER as a performance indicator to measure the modulation and demodulation capabilities of the model under different E_b/N_0 conditions. Currently, there is no unified mathematical relationship that can describe the changes in BER and E_b/N_0 in multi-order modulation and demodulation, but the mathematical relationship between MPSK and MQAM can be used as a reference. For MPSK, the BER is approximately $\frac{1}{log_2(M)}Q(\sqrt{2\frac{E_b}{N_0}sin(\frac{\pi}{M})})$, and for MQAM, the BER is approximately $\frac{4}{log_2(M)}(1-\frac{1}{\sqrt{M}})Q(\sqrt{3\frac{E_b}{N_0}\frac{log_2(M)}{M-1}})$, where $Q(x)$ is the Gaussian Q function, defined as $\frac{1}{\sqrt{2\pi}}\int_x^\infty e^{-\frac{t^2}{2}}dt$.

The rest of this paper is organized as follows: first, the introduction of related literature is given in Section 2; then, the description of the system model is given in Section 3; Section 4 details the structure of the proposed VQ-LSTM-AE; Section 5 gives the simulation results and discussion and analyzes the computational complexity; finally, a summary and future research areas are given.

2. Related Work

In this section, we first give a general introduction to the application of deep learning technology in modulation and demodulation and then introduce the development and application of LSTM-AE and codebook structure.

The autoencoder network is an unsupervised learning technique that applies backpropagation. Generally speaking, the autoencoder is divided into at least two parts: the encoder and the decoder. One part is responsible for compressing the input data into a low-dimensional latent space, and the other part restores the low-dimensional representation

to an output vector close to the input data by decompression. Similar to other networks, the encoder and decoder each have nonlinear activation functions and multiple hidden layers to obtain sufficient mapping capabilities. O'Shea [10] first considered the feasibility of autoencoder networks in deep learning for end-to-end communication systems, discussed the impact of loss functions and different channel regularization layers, and used an attention model specific to the field of wireless communications to deal with channel effects such as phase offset and frequency offset. Autoencoders' excellent adaptability enables them to have the ability to generate modulation and coding schemes close to the Shannon limit. Two years later, the team introduced adversarial networks into the model to synthesize new physical layer modulation and coding schemes for communication systems without the need for analytical models of channel impairments [11]. This work extended previous research on channel autoencoders to consider cases where the random channel response is unknown or difficult to model with closed-form analytical expressions. However, none of these frameworks have demonstrated the ability to uniformly implement multiple modulation schemes in a single framework. Shrestha et al. [12] discussed in detail the application of autoencoders in spatial modulation. To balance energy efficiency and detection performance, three spatial modulation frameworks based on autoencoders were proposed for high antenna correlation scenarios. Lu et al. [8] combined the characteristics of Sparse Autoencoders (SAE) and Denoising Autoencoders (DAE) and proposed a stacked autoencoder to implement M-ary phase position shift keying (MPSK) modulation and demodulation. Experiments have shown that the 2PSK and 4PSK schemes of the stacked autoencoder model can reach or exceed those of traditional methods. Wei et al. [9] implemented a modulation method for two to one hundred twenty-eight orders using a unified architecture that is close to the performance of traditional modulation schemes in most cases and even exceeds the traditional methods in some cases. However, these studies often require more complex models, which require software and hardware to provide precious resources for storage and use.

In the literature cited above, at least eight hidden layers are required to build a complete autoencoder, and complex structures such as stacking or special functional layers are often used to pursue better performance, which makes the number of hidden neurons considerable and becomes a source of resource burden. In addition, the autoencoder models mentioned in the references above also involve the mixed use of multiple different types of neural networks, and there is no simple and efficient unified network architecture to achieve unified modulation of multiple waveforms.

Since its introduction, the LSTM structure [13] has had a profound impact on the field of deep learning and sequence modeling and has become one of the most popular models in processing sequence data, time series predictions, natural language processing, and other tasks. Malhotra [14] combined it with an autoencoder network for multi-sensor anomaly detection. This method learns to reconstruct the behavior of normal time series and uses the reconstruction error to detect anomalies, which is very similar to compression transmission and recovery of signals in communication. Aaron et al. [15] further introduced vector quantization technology into the variational autoencoder, which greatly reduced the storage and computational costs of the model through the discrete vector space, effectively alleviated problems such as gradient vanishing, and improved the generalization ability of the model. This is an effective means to reduce the resource consumption of the autoencoder and is very suitable for performance enhancement in communication modulation.

3. System Model

In a typical communication system, there are at least two main components: the transmitter and the receiver. The transmitter is responsible for sending the original information in the form of a signal after encryption and compression. Conversely, the receiver filters and decompresses the received signal, which is often mixed with interference and noise. From a functional perspective, the autoencoder can be seen as a simplified version of a communication system. Therefore, it is plausible to consider using an autoencoder network

to simulate communication processes to fulfill the intelligence requirements that traditional systems cannot achieve.

To this end, this paper proposes a communication scheme based on deep learning (DL). The system model is shown in Figure 1. In the autoencoder model, the encoder network implements the function of the traditional modulator, and the decoder network implements the function of the traditional demodulator. The symbol string I_S representing the input information is converted into a one-hot code matrix $\boldsymbol{O}_e$, where $I_S \in \{0, 1, 2, \ldots, N-1\}$. The encoder network at the sending end uses the trained mapping function $f_e(\boldsymbol{O}_e, \theta_e)$ to compress and map different types of signals into constellation I/Q values, where θ_e represents the weight parameter set of the encoder part in the autoencoder network. The encoder output x is input into the Additive White Gaussian Noise (AWGN) channel.

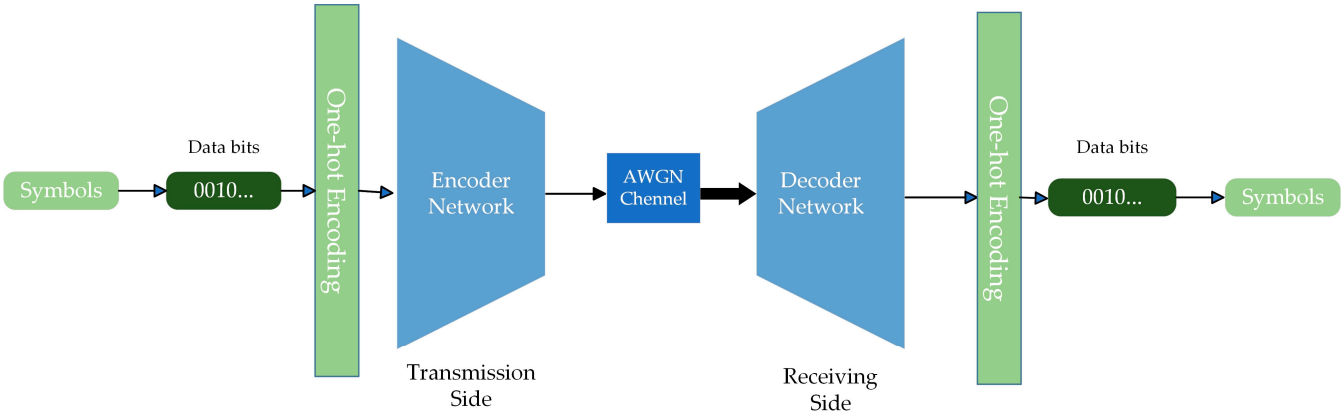


Figure 1. System model.

It is assumed that the signal transmitted by the transmitter passes through an AWGN channel. The noise generated by the channel has a fixed variance $\beta = (E_b/N_0)^{-1}$, where E_b/N_0 represents the ratio of the energy per bit transmitted to the noise power spectral density. The received signal input to the decoder network can be expressed as follows:

$$\boldsymbol{y_R} = \mathbf{H}x + \mathbf{n} \tag{1}$$

In the equation above, **H** represents the channel gain matrix, **n** is additive white noise, and the noise samples have a Gaussian distribution and have the same power density at all frequencies. The channel output is restored by the decoder network, and its mapping function is $O_d = f_d(y_R, \theta_d)$, where the channel output $\boldsymbol{y_R}$ is used as the input of the mapping function, and θ_d is the weight parameter set of the decoder part in the autoencoder network. The output of the receiver is $\boldsymbol{O}_S = \mathrm{L}(O_d)$, and the output of the decoder needs to be converted into a one-hot code and then compiled into a symbol string $\boldsymbol{O}_S$.

4. Modulation and Demodulation Model Based on a Vector Quantization LSTM Autoencoder

Modulation converts the signal into a form suitable for propagation in the channel by mapping it onto a carrier signal; demodulation converts the modulated signal into the original information format suitable for data processing. In the conversion process, modulation and demodulation accomplish the functions of compression and decompression to a certain extent. For example, in the traditional modulation method of MQAM, each constellation point represents one symbol, and the number of bits represented by each symbol is $log_2(M)$ bits. From this perspective, we believe that the essence of modulation is to process the original information into compressed signals that need to be distinguished from each other. The design concept of the autoencoder network aligns with this by performing dimensionality reduction and data re-representation. Existing technologies in this area primarily focus on enhancing system effectiveness, often overlooking the resource con-

sumption of the system itself. In certain scenarios, there is a direct conflict between resource usage and performance. Consequently, it is imperative to explore the use of lightweight models to minimize resource consumption. This paper introduces the integration of vector quantization technology and an LSTM-Autoencoder (LSTM-AE) into the modulation and demodulation processes.

4.1. LSTM-AE-Based Modulation and Demodulation Framework

LSTM-AE plays an important role in the task of time series anomaly detection [12]. The core of this task is the reconstruction of sequence data, which aligns with our objectives to some extent. Therefore, LSTM-AE was employed in the system model depicted in Figure 1 to perform the information compression and recovery processes in the communication system. This results in a modulation and demodulation framework based on LSTM-AE, as illustrated in Figure 2.

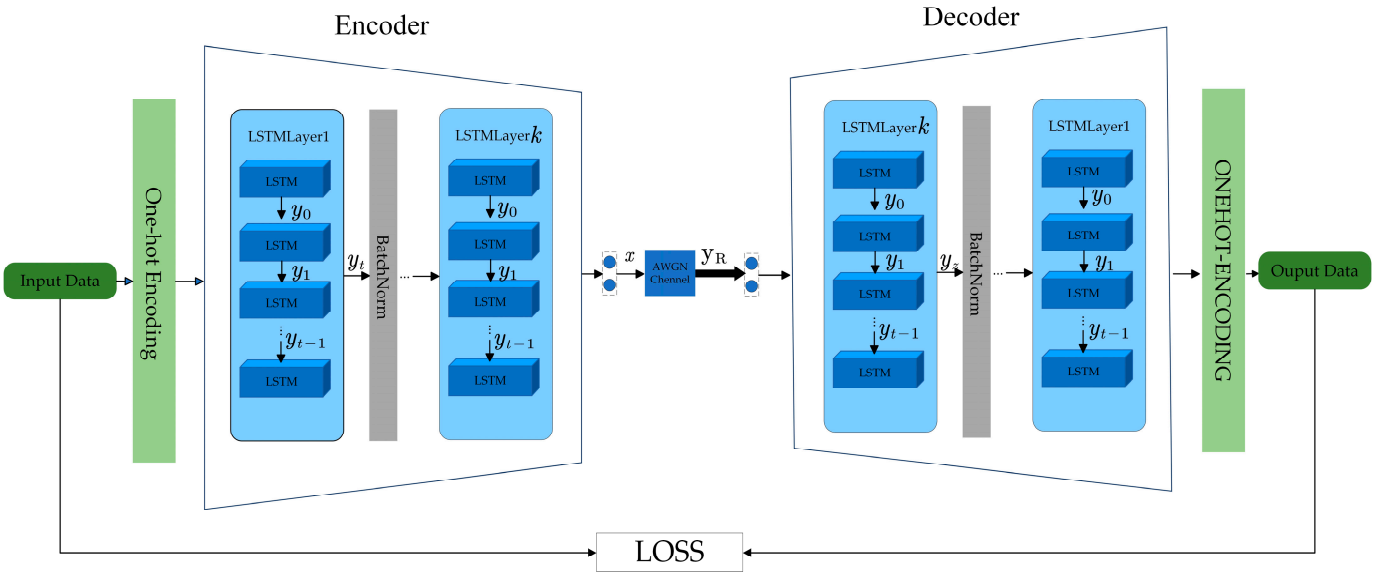


Figure 2. Architecture of LSTM-AE-based modulation.

The general framework of the improved LSTM-AE is shown in Figure 2. This section will describe the composition of the architecture in detail. The input of the network is the one-hot code matrix $\boldsymbol{O}_e$ of the symbol string S. The encoder consists of an LSTM hidden layer, a normalization layer, and an output layer, according to the processing flow, and the output is a two-dimensional real vector sequence $\boldsymbol{O}_{SN}$. The first layer accepts data input, which can be simply expressed as Formula (2), where X represents the input data, y_0 represents the initialized hidden information, c_0 represents the initialized cell information, and Θ_1 is the parameter of this layer.

$$y_1 = \text{LSTM}(X, y_0, c_0; \Theta_1) \tag{2}$$

After the input layer, the data are processed by BatchNorm and compressed by $K-1$ layers of LSTM. The output of the j layer can be expressed as Formula (3), where Y^{j-1} represents the output hidden state information from the previous layer, $\boldsymbol{C}^{j-1}$ comes from the output cell information of the previous layer, Θ_j is the parameter of this layer, and $j \in [1, k-1]$.

$$y_j = \text{LSTM}\left(Y^{j-1}, C^{j-1}; \Theta_j\right) \tag{3}$$

The output layer of the encoder is the Kth LSTM layer. While compressing the information, the potential representation is fixed to the form of a two-dimensional real

vector sequence. Similar to the previous two formulas, the output can be expressed as follows:

$$y_K = \mathrm{LSTM}\left(Y^{K-1}, C^{K-1}; \Theta_K\right) \tag{4}$$

The input of the decoder is the output of the channel, denoted by $\boldsymbol{I}_{RN}$. The decoder is designed symmetrically with the encoder. It consists of an input layer, a normalization layer, and a $K-1$ LSTM hidden layer according to the processing flow. The output is a matrix $\boldsymbol{O}_{RN}$. We used the LSTM block for a single signal rather than a time series, so we set the timestamp to one.

The structure of the LSTM is illustrated in Figure 3. The introduction of LSTM has addressed the issues of gradient vanishing and gradient exploding that were prevalent in earlier recurrent neural networks (RNNs) to a considerable extent. LSTM incorporates long-term memory while blending the output with short-term memory, thereby enhancing the network's ability to capture temporal dependencies. An LSTM unit contains a forget gate, an input gate, and an output gate, which can be expressed by Equations (5)–(10) [15]. Formula (5) gives the processing method of input data x_t and the previous round output y_{t-1}, where W_z and R_z represent the weight matrices about x_t and y_{t-1}, respectively, and b_z is the bias. $g()$ is the activation function.

$$z_t = g(W_z x_t + R_z y_{t-1} + b_z) \tag{5}$$

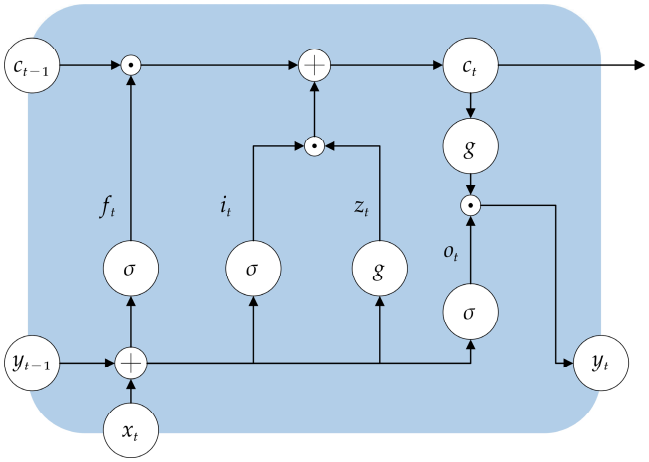


Figure 3. Architecture of a typical LSTM block.

Formula (6) gives the process of updating the input gate, W_i and R_i represent the weight matrices about x_t and y_{t-1}, respectively, b_i is the bias, and $\sigma(x)$ is the activation function.

$$i_t = \sigma(W_i x_t + R_i y_{t-1} + b_i) \tag{6}$$

Formula (7) gives the process of updating the forget gate, which is the key to information transmission and determines the degree of information retention, where W_f and R_f represent the weight matrices of x_t and y_{t-1}, respectively, and b_f is the bias.

$$f_t = \sigma\left(W_f x_t + R_f y_{t-1} + b_f\right) \tag{7}$$

Formula (8) is the calculation method of the cell value. The forget gate selects the cell value of the previous round, and the input gate processes the input data, and the cell value of this round is calculated in combination; $\odot$ is the vector dot product symbol.

$$c_t = z_t \odot i_t + c_{t-1} \odot f_t \tag{8}$$

Formula (9) gives the process of updating the output gate. The output gate considers the current round input, the previous round output, and the cell value together and performs nonlinear transformation through the activation function $\sigma(x)$. Here W_o and R_o represent the weight matrices about x_t and y_{t-1}, respectively, and b_o is the bias.

$$o_t = \sigma(W_o x_t + R_o y_{t-1} + b_o) \tag{9}$$

Finally, we can obtain the output y_t of the LSTM unit in this round from Formula (10), which is calculated by combining the cell value in this round with the output gate result.

$$y_t = g(c_t) \odot o_t \tag{10}$$

Generally speaking, we use the logistic sigmoid function for the activation function $\sigma(x) = \frac{1}{1+e^{1-x}}$ and the $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ function for $g(x)$. LSTM-AE uses a large number of LSTM blocks as hidden layer units, which play the role of compressing information to the latent space in the encoder and decompressing and restoring the signal in the decoder.

In Figure 4, the encoder and decoder can be considered as utilizing the connection and stacking of multiple LSTM layers, with the detailed structure shown in Figure 4. The horizontal transmission is time information; the LSTM layer will be reused many times and finally leave the filtered information. The spatial information is transmitted vertically. The hidden state Y output by the previous layer is used as the input information of the next layer. Increasing the depth can provide stronger fitting ability. After the one-hot code matrix $\boldsymbol{O_e}$ is input to the encoder, it first passes through $k-1$ LSTM layers. Each layer consists of multiple LSTM units. The compression ratio of each layer can be controlled by setting the number of LSTM layer output information units. Due to the LSTM cells' ability to store long-term information, the network can decouple the input data dimension N from the output data dimension of each layer by adjusting the hidden layer size, thus avoiding the need for exponential decay patterns. The last layer of the encoder is the output layer. When the hidden layer dimension is set to two, the purpose of outputting a fixed range of feature two-dimensional real vectors can be achieved. The two dimensions represent the I/Q signals. The output of LSTM is shown in Formula (10), which is obtained by multiplying the activation value of the cell state $c^{(t)}$ and the result of the output gate $o^{(t)}$. The activation function generally uses the tanh function, so it can be ensured that the requirement is met within the interval $(-1, 1)$. Before the final output, a normalization operation is added, which helps to speed up the convergence of the network and prevents the gradient from disappearing.

As mentioned above, the decoder part is the reverse operation of the encoder part that decompresses the received signal containing channel noise into a tensor of the same shape as the input data and then uses the argmax function to restore it to the output data in the one-hot encoding format. The received signal can be represented as $\boldsymbol{I}_{RN} = \boldsymbol{O}_{SN} + \mathbf{n}$, where each element of $\mathbf{n}$ follows a complex Gaussian distribution with zero mean and variance $\sqrt{\frac{P_s}{SNR}}$, P_s is the signal power, and SNR is the signal-to-noise ratio specified during training or testing.

According to the conclusion of [6], during training we set the loss function to the mean square error function. As shown in Formula (11), the training direction of the model is guided by comparing the difference between the input $\boldsymbol{O_e}$ and the output $\boldsymbol{O_d}$ of the encoder network. In the formula, π is the set of all adjustable parameters, and B is the batch size.

$$\min_{\pi}(\text{loss}) = \frac{1}{B}\left(\sum_{\text{n}=1}^{B}\sum_{\text{i}=1}^{\text{N}}(\boldsymbol{O_d}(\text{i}) - \boldsymbol{O_d}(\text{i}))^2\right) \tag{11}$$

The Adam optimizer is widely used as the default optimizer due to its adaptability and stability. We use Adam instead of the AdaDelta optimizer to adapt to the task background with fast convergence.

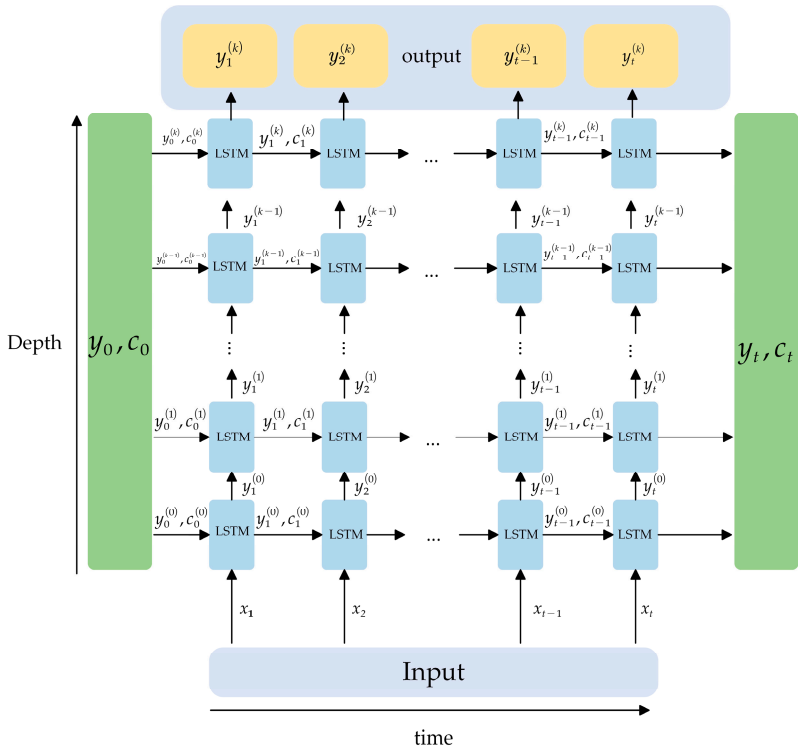


Figure 4. Working structural diagram of the encoder.

4.2. Vector Quantization

In the application of classic autoencoder networks, the input data usually have high-dimensional features, leading to significant training costs in terms of computational resources and substantial storage space for a large number of model parameters. Vector quantization technology effectively mitigates these resource requirements for model training and storage, thereby reducing the overall cost. The DeepMind team [13] successfully integrated the idea of vector quantization with the autoencoder model, proving that discrete variables can also fully represent information. Given the discrete nature of modulated signals, using discrete variables as compressed feature vectors is naturally appropriate. Vector quantization technology can better capture the categorical properties in the data and assist the model in generating clearer outputs.

As shown in Figure 5, a codebook structure was added based on Figure 2, the main purpose of which was to discretize the output of the encoder into the closest vector in the codebook and convert the continuous encoded output into a discrete representation. First, we needed to define a potential embedding space $e \in R^{d \times D}$, that is, the codebook. The codebook contains d vectors as quantization centers, each with a dimension of $D = 2$. The codebook is not fixed. After the initial random initialization, it will gradually approach the distribution after compression of different signals during training. The output of the encoder O_{SN} searches for the vector in the potential embedding space according to the nearest neighbor principle and performs one-hot coding so that only the most similar quantization center index d matched is 1, and the rest are 0. The discrete latent variable index z distribution is shown in Formula (12).

$$Q\left(z = d \mid O_{SN}\right) = \begin{cases} 1, & for\ d = argmin_j \left\| O_{SN} - e_j \right\|_2 \\ 0, & otherwise \end{cases} \tag{12}$$

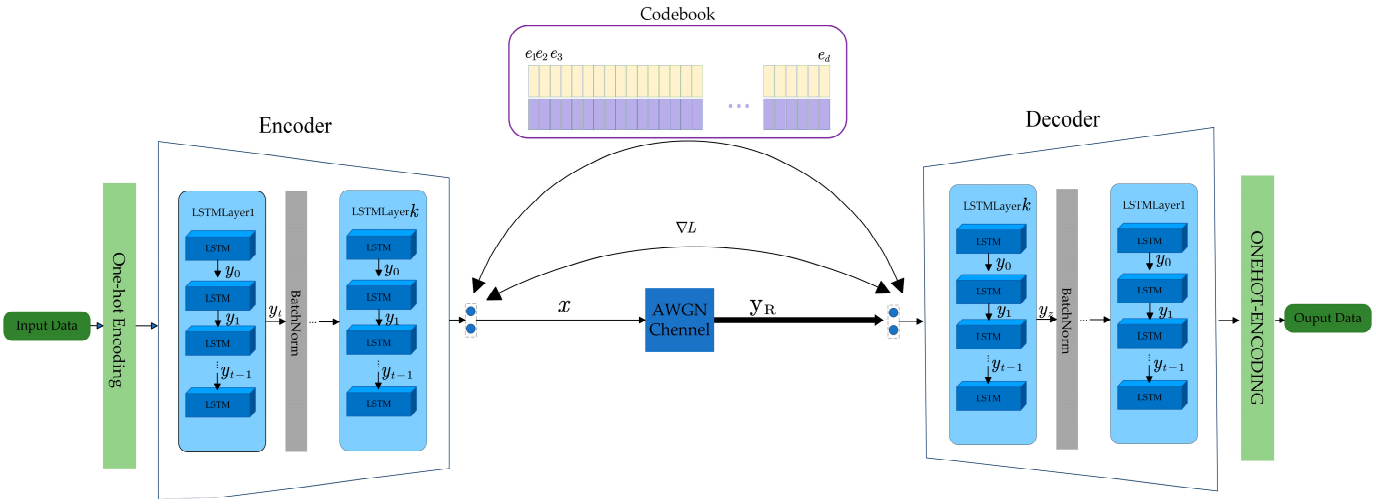


Figure 5. Framework of VQ-LSTM-AE-based modulation.

After completing the index search for the encoder output $\boldsymbol{O}_{SN}$, we passed the corresponding discrete signal into the AWGN channel, and the channel output $\boldsymbol{y}_R$ is the input of the decoder. In this step, the use of codebook quantization interrupts the gradient propagation of the model, but because the expectation of the noise signal in the AWGN channel is 0, it is a random variable unrelated to the transmitted signal, so the additive noise has no effect on the gradient transfer. By directly assigning the gradient of the encoder output to the decoder input, the gradient discontinuity problem can be avoided.

Although the gradient replication method connects the transmission and reception ends, it is still impossible to complete the learning of the codebook. To achieve this goal, it is necessary to modify the loss function of Formula (11) to indicate the iteration direction of the codebook. Specifically, in addition to the reconstruction loss, a part that measures the difference between the compressed potential representation and the predefined codebook and a part that measures the difference between the encoder's fixed output and the codebook quantization center were added. The former constrains the encoder output to be close to the codebook, and the latter constrains the codebook to learn the encoder output. The complete loss function is shown in Formula (13):

$$L = L_{Re} + \| \text{ sg}[\boldsymbol{O}_{SN}] - e \|_2^2 + \beta \| \boldsymbol{O}_{SN} - \text{sg}[e] \|_2^2 \tag{13}$$

In Equation (13), L_{Re} represents the reconstruction loss, which is consistent with Formula (13); e represents the quantization center selected in the codebook; sg means stopping the gradient update and fixing variables; and β is a hyperparameter used to control the weight of this item in the overall loss function—-its value is generally 0.25.

5. Results

5.1. Training Settings

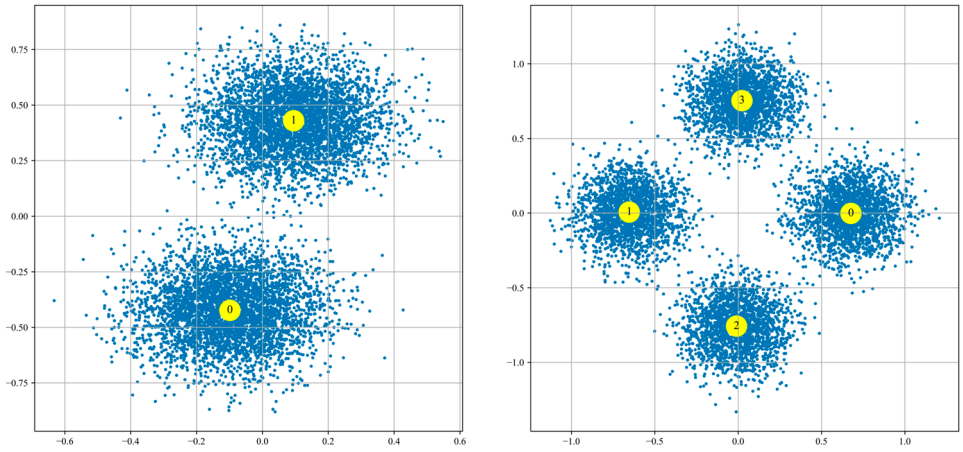
The structure of the VQ-LATM-AE model itself has a lightweight effect, so a large amount of computing resources is not required during training. When performing end-to-end joint training, the training set used is composed of 8000 random integers ranging from 0 to N – 1, and the test set is a random number of 100,000 symbols. The E_b/N_0 during training cannot be too high, so that the model deliberately ignores the impact of noise, nor can it be set too low, to cause slow convergence. The other tuned hyperparameter settings are shown in Table 1. T is the time step length, z is the hidden layer size, and the minimum setting can be four. When it is greater than eight, the effect is relatively stable. There are D quantization centers in the codebook, and β is the weight hyperparameter in the loss function that affects the learning speed of the codebook.

Table 1. Training hyperparameter setting table.

Hyperparameters	VQ-LATM-AE
E_b/N_0 (dB)	7 (N = 2/4/8/16) 12 (N = 32)
Batch Size	256
t	1
z	4
Learning rate	0.01
D	64
β	0.25

5.2. Constellation Learning

The constellation diagram represented by the I/Q signal learned by the LSTM autoencoder is one of the ways to visualize the modulation capability. The constellation diagram of the VQ-LSTM-AE is shown in Figure 6. Figure 6 shows the modulation results of the model for second-, fourth-, eighth, sixteenth, and thirty-second-order signals, respectively. The model converges within 100 epochs for all types of signals. Among them, the constellation diagrams shown in Figure 6a,b are similar in distribution to traditional BPSK and QPSK, so the performance is also similar to the traditional method; Figure 6c shows a constellation distribution that is different from that of traditional 8PSK or 8QAM. The model sets one constellation point at the center surround by the other constellation points. This change increases the Euclidean distance between points, making it easier to distinguish them from each other. The training in Figure 6d,e was carried out under the background of $E_b/N_0 = 12$ dB. Compared with the traditional method, the distribution of the constellation diagram tends to evolve towards a concentric circle structure, which also changes the interval and optimizes the space utilization efficiency.



(**a**) (**b**)

Figure 6. *Cont.*

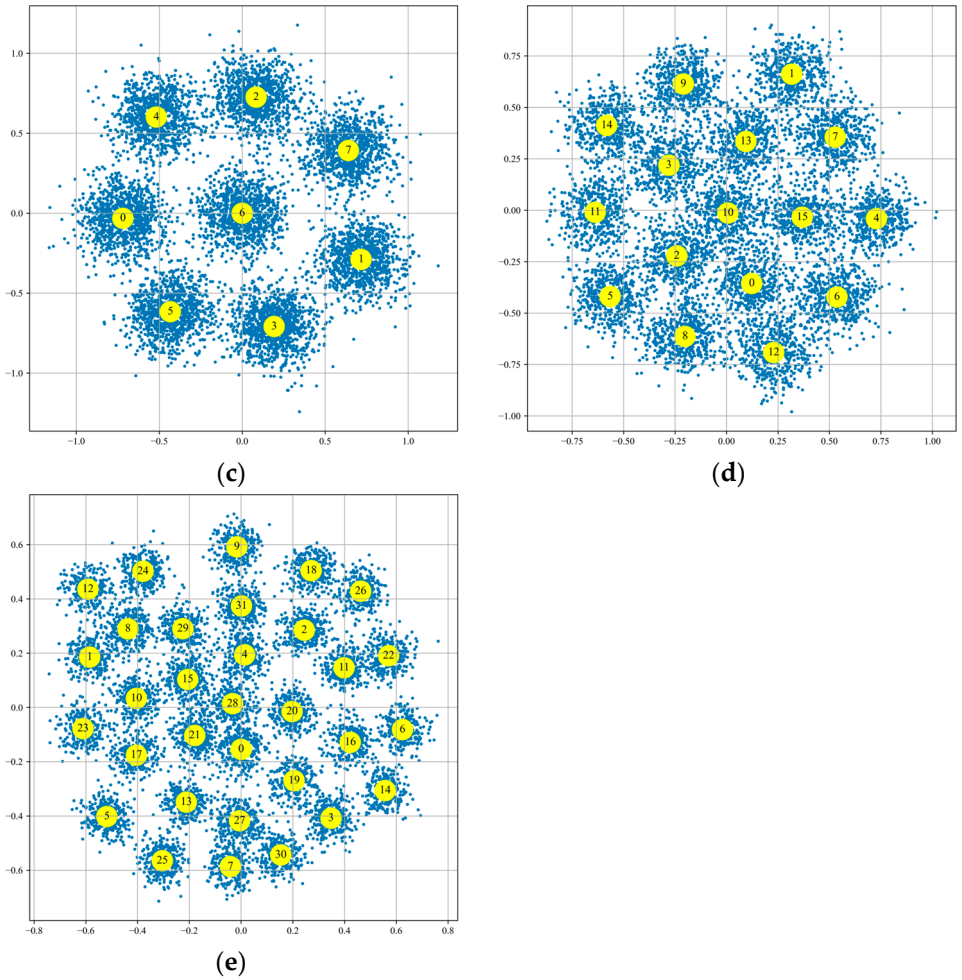


Figure 6. Constellations learned by the proposed model. (**a**) Second-order constellation diagram; (**b**) fourth-order constellation diagram; (**c**) eighth-order constellation diagram; (**d**) sixteenth-order constellation diagram; (**e**) thirty-second-order constellation diagram.

5.3. BER Performance Comparison

The closest model to this study is that described in reference [10], so we compare the proposed VQ-LSTM-AE framework with the DAN-AE model proposed in the reference, and the results are shown in Figure 7. Figure 7a,b compare the performance curves of BER versus E_b/N_0 of second-order and fourth-order LSTM-AE, BPSK, and DAN-AE, respectively. We can see that the three curves are almost identical, and the difference is too small to be considered a reasonable error, which is consistent with the consistency of the constellation distribution. In Figure 7c, VQ-LSTM-AE has almost the same performance as 8QAM before 4 dB, which is better than DAN-AE, while DAN-AE is slightly weaker than 8QAM before 6 dB, but surpasses it afterward; DAN-AE and VQ-LSTM-AE both surpass 8QAM after 4 dB, and VQ-LSTM-AE drops by nearly an order of magnitude compared to DAN-AE after 10 dB. In Figure 7d, both autoencoder-based models are inferior to traditional methods, and the curves of VQ-LSTM-AE and DAN-AE overlap. In Figure 7e, at the 32nd order, both autoencoder-based models do not surpass 32QAM, but VQ-LSTM-AE gradually widens the gap with DAN-AE after 4 dB, and finally drops by nearly an order of magnitude. Considering that LSTM-AE reduces more than 20% of parameters compared to DAN-AE, this balance is acceptable.

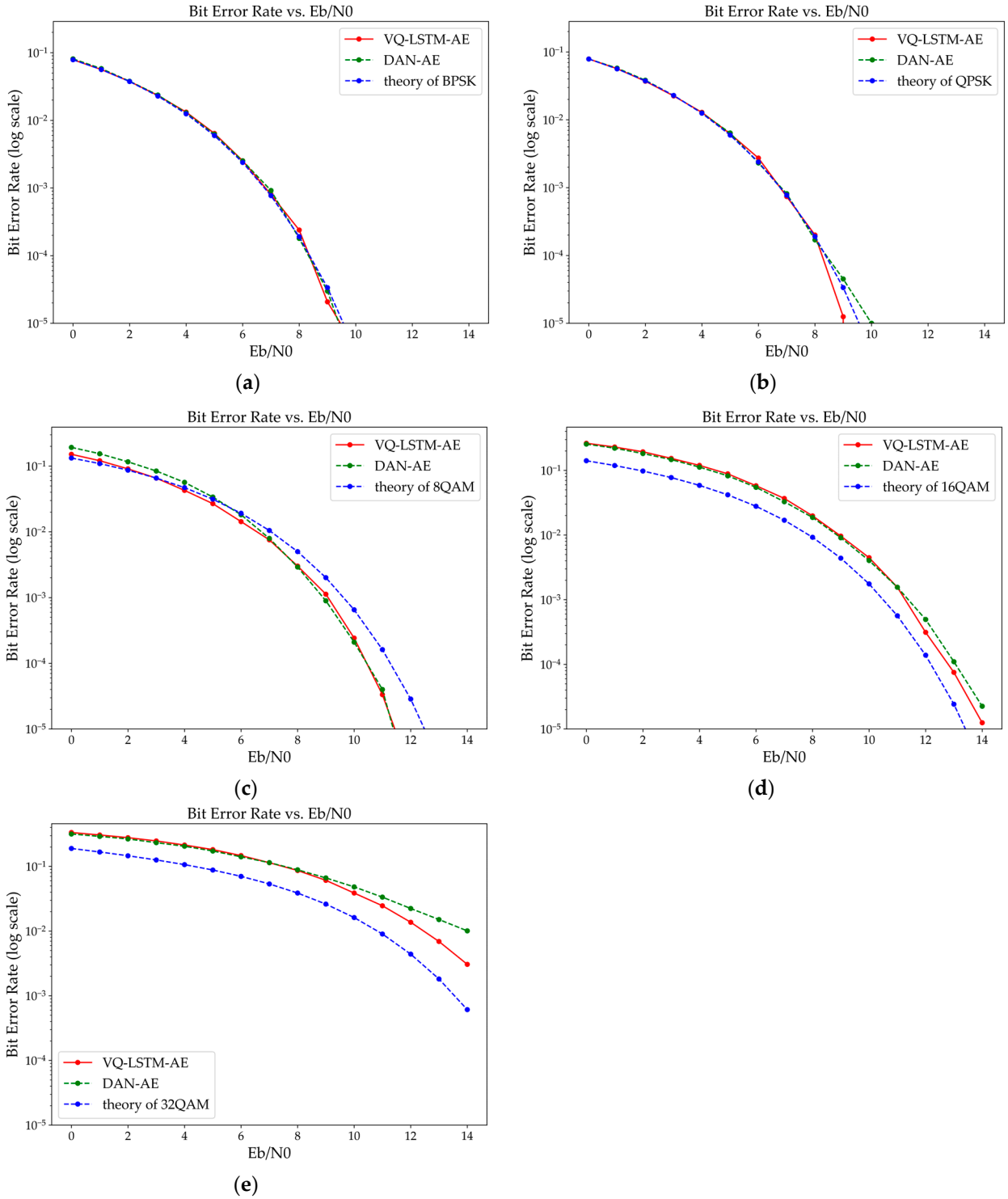


Figure 7. Comparison of the BER performance of VQ-LSTM-AE with other models. (**a**) Second-order BER performance; (**b**) fourth-order BER performance; (**c**) eighth-order BER performance; (**d**) sixteenth-order BER performance; (**e**) thirty-second-order BER performance.

5.4. Effectiveness and Complexity

5.4.1. Effectiveness

First, we repeated experiments to verify the BER performance. Ten experiments were conducted for each order, and significance level analysis was performed on the BER data under different E_b/N_0 conditions. Under small sample conditions, T distribution was used for independent testing, and the null hypothesis (H0) was set, as there was no significant

difference in BER between the old and new models. The alternative hypothesis (H1) is that there is a significant difference in BER between the old and new models. The significance level was set to $\alpha = 0.05$; that is, if the p value is less than 0.05, we reject the null hypothesis and believe that the performance difference between the VQ-LSTM-VAE and DAN-AE models under this E_b/N_0 condition is statistically significant. The statistical results are shown in Table 2.

Table 2. Significance level test results.

Order	E_b/N_0										
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
2	0.9860	0.4804	0.1249	0.8961	0.1589	0.0107	0.0909	0.6328	0.0478	0.1157	0.2613
4	0.9988	0.9459	0.9930	0.8216	0.8027	0.0900	0.0763	0.7385	0.6534	0.9313	0.2785
8	0.0002	0.0010	0.0001	0.0002	0.0002	0.0003	0.0003	0.0001	0.0001	0.0001	0.0022
16	0.1520	0.9662	0.1189	0.6116	0.8844	0.1877	0.0227	0.0960	0.0034	0.0086	0.0014
32	0.3196	0.0080	0.0001	0.0011	0.0001	0.0001	0.0001	0.0000	0.0000	0.0000	0.0000

Most of the significance levels of orders 2, 4, and 16 in the table are greater than 0.05, which verifies the conclusion that the model proposed in this article has the same performance as DAN-AE in these three cases; the values for orders 8 and 32 were mostly less than 0.05, which also shows that the BER performance of VQ-LSTM-AE and DAN-AE is significantly different. From Figure 7, we can know that the performance of VQ-LSTM-AE is better than DAN-AE.

During the training process, perplexity P can be used as an indicator to monitor the training effect of the codebook in real time. Perplexity P uses the information entropy of the category distribution to detect the learning effect of the model, which can be considered as the inverse of the exponential function of information entropy. Information entropy is a measure of the uncertainty of a random variable. Perplexity is positively correlated with information entropy. The larger the perplexity, the more signals can be distinguished. The calculation of perplexity P is shown in Formula (14). $\overline{p}_j$ is the jth element in the average probability distribution of all samples, ε is a very small positive number used to avoid mathematical problems in logarithmic operations, and N is the number of elements in the probability distribution.

$$P = \mathrm{e}^{-\sum_{j=1}^{N} \overline{p}_j log(\overline{p}_j + \varepsilon)} \tag{14}$$

The perplexity monitoring during the training process of each order is shown in Figure 8. Figure 8a–c show the situations of the second, fourth, and eighth orders, respectively. The model can find the optimal solution very quickly and converge near the extreme values of 1.9, 3.9, and 7.9, which means that the required corresponding signal types are successfully distinguished; Figure 8d shows the training situation in the 16th order. Compared with the second to eighth orders, it took more epochs to complete convergence, but the goal of a perplexity of about 16 was still achieved within 30 rounds; Figure 8e shows that the perplexity result converges to around 30, which is slightly away from the target 32, which is one of the reasons why the BER performance did not reach 32QAM.

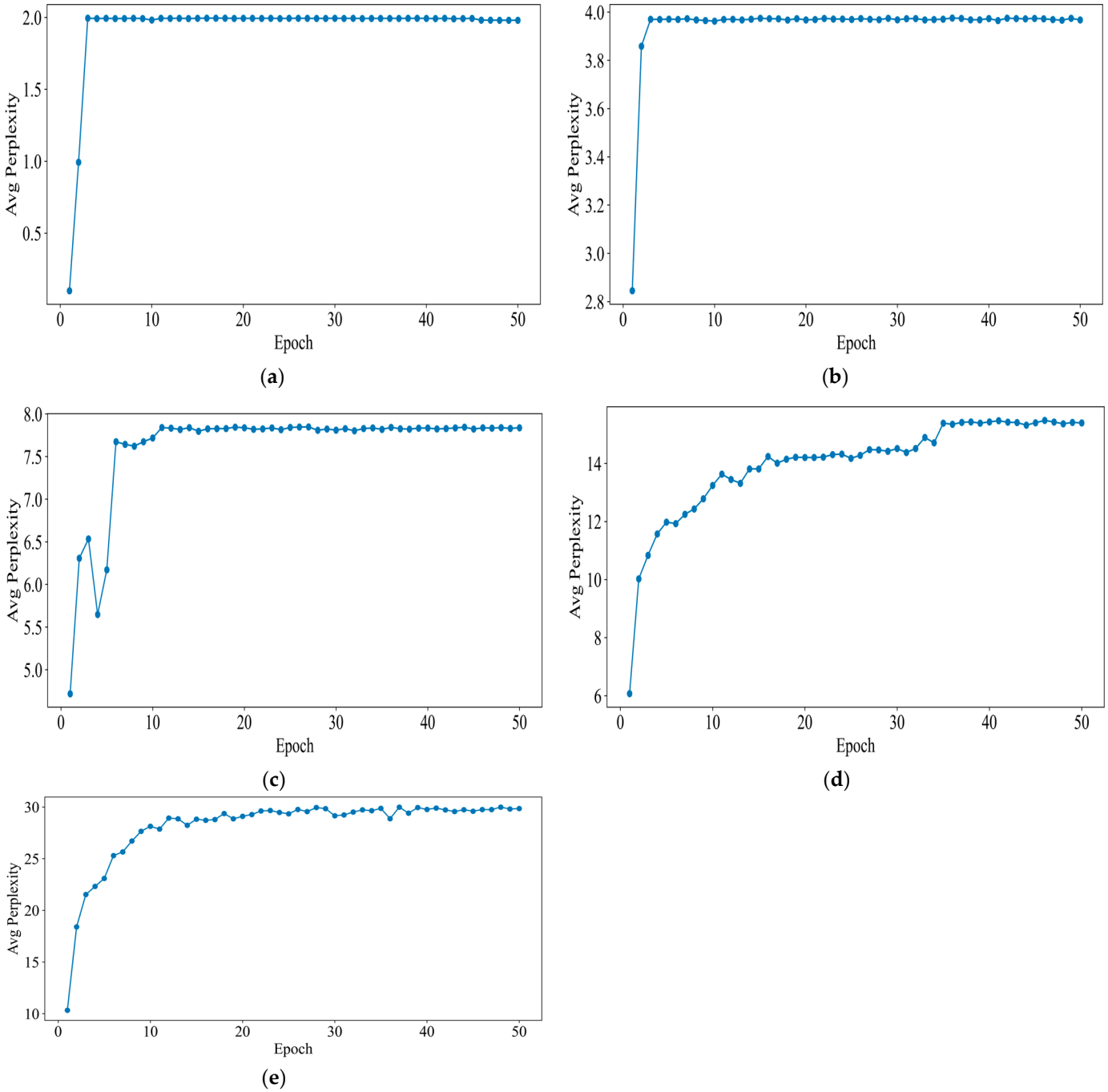


Figure 8. Perplexity of VQ-LSTM-AE during training. (**a**) Second-order perplexity; (**b**) fourth-order perplexity; (**c**) eighth-order perplexity; (**d**) sixteenth-order perplexity; (**e**) thirty-second-order perplexity.

5.4.2. Computational Complexity

In this section, we study the computational complexity of the VQ-LSTM-AE model and compare it with existing techniques. Due to the offline training mode, generation of the vector quantization codebook can be obtained offline, and only the best matching value needs to be queried, so the training complexity can be ignored, and the focus is on the prediction complexity. Specifically, the complexity of the proposed solution is determined by the number of autoencoder layers, the number of neurons in each layer, the dimension of the input data, and the number of codewords in the codebook. Therefore, the computational complexity can be expressed as Formula (15), where L_e and L_d represent the number of

layers of the encoder and decoder, Z_e and Z_d represent the size of the hidden layer of the encoder and decoder, C and D represent the dimension and size of the codebook, and I_e and I_d represent the dimension of the input data of the hidden layer of the encoder and decoder, respectively.

$$C_{VQ-LSTM-AE} = L_e \cdot \left(I_e \cdot Z_e + {Z_e}^2 \right) + C \cdot D + L_d \cdot \left(I_d \cdot Z_d + {Z_d}^2 \right) \tag{15}$$

Formula (15) shows that the complexity of VQ-LSTM-AE can be simplified to $O\left(L_e \cdot \left(I_e \cdot Z_e + {Z_e}^2\right)\right)$. For the existing multi-layer perceptron-based model [6], its complexity depends on the number of hidden layers and the dimensions of input/output. Therefore, the complexity of the existing DAN-AE can be given by Formula (16), where L_e and L_d represent the number of layers of the encoder and decoder, Z_e and Z_d represent the size of the hidden layers of the encoder and decoder, and I_e and I_d represent the dimensions of the input data of the hidden layers of the encoder and decoder, respectively, which can be expressed as $O(L_e \cdot I_e \cdot Z_e)$. The complexity of the two is at the same order of magnitude, but VQ-LATM-AE has an advantage because it has a smaller number of parameters.

$$C_{DAN-AE} = O(L_e \cdot I_e \cdot Z_e) + O(L_d \cdot I_d \cdot Z_d) \tag{16}$$

Table 3 compares the number of parameters of DAN-AE and VQ-LSTM-AE. Regarding two, four, eight, and sixteen orders, VQ-LSTM-AE reduces the number of parameters by 70%, 66%, 58%, and 22%, respectively, compared with DAN-AE, but is greater than DAN-AE at 32 orders. Therefore, the method proposed in this paper is suitable for scenarios such as military satellite communications that pursue stability and a low bit error rate, in which the total amount of resources is limited and it is necessary to improve resource utilization.

Table 3. Parameter comparison table.

Order	VQ-LSTM-AE(s)	DAN-AE
2	528	1812
4	656	1942
8	1008	2442
16	2096	2722
32	5808	3762
Average	2019	2536

We also tested the effectiveness of our proposed method from the perspective of computing time. Computation time is an important indicator for evaluating classifier performance, especially in the era of big data, where real-time classification requires a large amount of data. Table 4 shows the training time of the VQ-LSTM-AE algorithm and DAN-AE. We observed that the VQ-LSTM-AE algorithm consumes significantly less time in training than the DAN-AE method, with an average training time of only 7.4% that of DAN-AE, and this trend becomes more obvious as the order increases. The average number of parameters of VQ-LSTM-AE in five cases is only 79.6% that of DAN-AE. This characteristic of de-linear correlation between computational time and the order of input data will show greater advantages and appeal when it is expanded to 64 or 128 orders in the future.

Table 4. Training time comparison table.

Orders	VQ-LSTM-AE(s)	DAN-AE(s)
2	1.81	13.3
4	2.82	20.19
8	3.57	31.76
16	5.68	39.07
32	5.94	158.42
Average	3.964	52.548

6. Conclusions

This paper used a variant of the autoencoder to solve the problem of rapid generation of unified framework communication waveforms, developed an LSTM autoencoder framework that supports fast switching of multiple waveforms, and introduced vector quantization technology to further reduce resource overhead. Through theoretical analysis with simulation tuning, its optimal structural parameters were obtained. The test results show that the lightweight model based on vector quantization LSTM-AE can achieve the same or better modulation and demodulation performance as the existing model, and the training convergence speed is significantly improved. It can be applied to demodulation systems with limited resources or scenarios with higher performance requirements.

This research reduces the resource consumption of adaptive modulation technology, improves the overall efficiency of the communication system, and brings more possibilities for its application in scenarios such as military satellite communications. The requirements for channel information perception accuracy, fast feedback, and increasing real-time requirements of update mechanisms also bring more challenges. The proposed model will be further studied in the future to be extended to a wider range of application scenarios, including waveform generation with more orders, a joint channel and source coding and modulation unified autoencoder framework, and generation with specific frequency domain constraints (such as continuous phase) modems and other anti-jamming technologies to deal with increasingly intelligent jamming methods.

Author Contributions: Conceptualization, W.G. and S.X.; methodology, W.G.; software, W.G.; validation, H.W. and Y.Z.; writing—original draft preparation, W.G.; writing—review and editing, S.X. and Y.L.; visualization, W.G. and Y.L.; supervision, H.W. and Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are private, from the School of Electronics and Information Engineering, Nanjing University of Information Science and Technology, and 63rd Research Institute, National University of Defense Technology.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Kshirsagar, P.R.; Reddy, D.H.; Dhingra, M. A Review on Comparative study of 4G, 5G and 6G Networks. In Proceedings of the 2022 5th International Conference on Contemporary Computing and Informatics (IC3I), Uttar Pradesh, India, 14–16 December 2022; pp. 1830–1833.
2. FG-NET-2030. Network 2030: A Blueprint of Technology, Applications and Market Drivers towards the Year 2030 and Beyond. Available online: https://www.itu.int/en/ITU-T/focusgroups/net2030/Documents/White_paper.pdf (accessed on 18 July 2024).
3. Fontanesi, G.; Ortíz, F.; Lagunas, E.; Baeza, V.M.; Vázquez, M.Á.; Vásquez-Peralvo, J.A.; Minardi, M.; Vu, H.N.; Honnaiah, P.J.; Lacoste, C.; et al. Artificial Intelligence for Satellite Communication and Non-Terrestrial Networks: A Survey. *arXiv* **2023**, arXiv:2304.13008.
4. Yang, Y.; Dang, S.; Wen, M.; Guizani, M. Millimeter Wave MIMO-OFDM With Index Modulation: A Pareto Paradigm on Spectral-Energy Efficiency Trade-Off. *IEEE Trans. Wirel. Commun.* **2021**, *20*, 6371–6386. [CrossRef]
5. Salh, A.; Audah, L.; Abdullah, Q.; Abdullah, N.; Shah, N.S.M.; Saif, A. Trade-off Energy and Spectral Efficiency With Multi-Objective Optimization Problem in 5G Massive MIMO System. In Proceedings of the 2021 1st International Conference on

Emerging Smart Technologies and Applications (eSmarTA), Sana'a, Yemen, 10–12 August 2021; IEEE: Sana'a, Yemen, 2021; pp. 1–6.
6. Al-Obiedollah, H.M.; Cumanan, K.; Thiyagalingam, J.; Tang, J.; Burr, A.G.; Ding, Z.; Dobre, O.A. Spectral-Energy Efficiency Trade-Off-Based Beamforming Design for MISO Non-Orthogonal Multiple Access Systems. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 6593–6606. [CrossRef]
7. Zhang, J.; Chen, S.; Maunder, R.G.; Zhang, R.; Hanzo, L. Adaptive Coding and Modulation for Large-Scale Antenna Array Based Aeronautical Communications in the Presence of Co-Channel Interference. *IEEE Trans. Wirel. Commun.* **2017**, *17*, 1343–1357. [CrossRef]
8. Lu, C.; Chen, P.; Zhong, H.; Wang, M. M-Ary Phase Position Shift Keying Demodulation Using Stacked Denoising Sparse Autoencoders. *Electronics* **2022**, *11*, 1233. [CrossRef]
9. Wei, P.; Wang, S.; Luo, J. Adaptive Modem and Interference Suppression Based on Deep Learning. *Trans. Emerg. Telecommun. Technol.* **2021**, *32*, e4220. [CrossRef]
10. O'Shea, T.J.; Karra, K.; Clancy, T.C. Learning to Communicate: Channel Auto-Encoders, Domain Specific Regularizers, and Attention. In Proceedings of the 2016 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), Limassol, Cyprus, 12–14 December 2016.
11. Oshea, T.J.; Roy, T.; West, N.; Hilburn, B.C. Physical Layer Communications System Design Over-the-Air Using Adversarial Networks. In Proceedings of the 2018 26th European Signal Processing Conference (EUSIPCO), Rome, Italy, 3–7 September 2018; IEEE: Rome, Italy, 2018; pp. 529–532.
12. Shrestha, S.; Naser, S.; Bariah, L.; Muhaidat, S.; Sofotasios, P.C.; Elgala, H.; Damiani, E. Autoencoder-Based Spatial Modulation for the Next Generation of Wireless Networks. *IEEE Internet Things J.* **2024**, 1. [CrossRef]
13. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
14. Malhotra, P.; Ramakrishnan, A.; Anand, G.; Vig, L.; Agarwal, P.; Shroff, G. LSTM-Based Encoder-Decoder for Multi-Sensor Anomaly Detection. *arXiv* **2016**, arXiv:1607.00148.
15. Van Den Oord, A.; Vinyals, O. Neural discrete representation learning. *arXiv* **2017**, arXiv:1711.00937.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.