Deep Learning Decoding Model Based on Group Codes

Yi He

Abstract

This paper studies a special class of encoding functions based on the fundamental theory of group codes and attempts to decode these functions using deep learning models. Experimental results show that the proposed method provides new ideas and directions for group code decoding, demonstrating good decoding performance and potential applications.

Keywords: deep learning, group codes, decoding model

目录

1 Introduction

With the rapid development of communication and information processing technologies, coding theory plays a crucial role in ensuring the reliability and efficiency of data transmission. As an important type of code, group codes are widely used in modern communication systems due to their strong algebraic structure and superior error-correcting capabilities. However, the decoding process of group codes is often computationally complex, and developing simple decoding methods for group codes remains an important research topic.

Based on the fundamental theory of group codes, this paper investigates a special class of encoding functions and attempts to decode these functions by constructing deep learning models. We first introduce the special encoding functions and the proposed deep learning decoding model; then improve the model and analyze the training effects; finally, summarize the research results and outline future research directions.

2 Fundamental Theory of Group Codes

2.1 Introduction to Codes

In modern life, communication is an essential basic need. The essence of communication is the transfer of information between two points. For example, an information source (point A) needs to transmit information to another point (point B), the receiver, forming a basic communication system.

Converting information from the source into digital form for transmission to B is called digital communication.We focus on a special type of digital communication: binary digital information, represented by finite-length binary n-tuples$(c_1, c_2, \ldots, c_n)$, (where $c_i = 0$ or 1).These can be viewed as elements ofZ_2 ,$(c_1, c_2, \ldots, c_n) \in Z_2^n$, where we typically represent information in a set using n-tuples of the same dimension.

In practice, digital information transmission is often disturbed, causing B to receive possibly corrupted data. A fundamental solution is error-resistant encoding: encoding the original information before transmission so that B can use known rules to translate the received data back to the original information, as shown in Fig. 1.

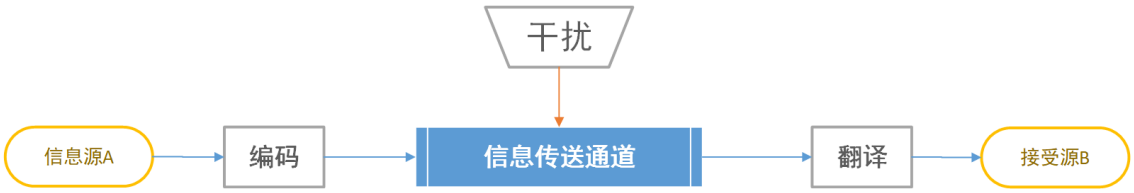


图 1: Information transmission steps

Generally, let the set of original digital information be Z_2^k, (k is a positive integer). Let n be a positive integer greater than k, and define an injective function

$$E : Z_2^k \to Z_2^n$$

$$(c_1, c_2, \ldots, c_k) \to (c_1, c_2, \ldots, c_n)$$

We call E an encoding function, ImE the code, elements of ImE codewords, n the code length, and all elements of Z_2^n words. Words not in ImE are called erroneous codewords, representing data corrupted by noise (i.e., $\mathbf{v} \in Z_2^n$ but $\mathbf{v} \notin$ ImE).

2.2 Group Codes

Clearly, Z_2^n is an n-dimensional linear space over Z_2. For an encoding function $E : Z_2^k \to Z_2^n$, if ImE is a subspace of Z_2^n, it is called a group code.

Let $C =$ ImE be a group code of length n, a k-dimensional subspace over Z_2, with basis $v_1, v_2, \ldots, v_k$. Any codeword $u = (u_1, u_2, \ldots, u_n) \in C$ can be uniquely expressed as a linear combination:

$$u = \alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_k v_k, \quad \alpha_i \in Z_2.$$

3 A Special Class of Encoding Functions and Deep Learning Decoding Model Construction

3.1 Special Encoding Functions

Define the matrix

$$G = \begin{pmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_k^T \end{pmatrix} = \begin{pmatrix} v_{11} & v_{12} & \dots & v_{1n} \\ v_{21} & v_{22} & \dots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{k1} & v_{k2} & \dots & v_{kn} \end{pmatrix}$$

where G is a $k \times n$ matrix of rank k over Z_2. The codeword is then

$$u = (\alpha_1, \alpha_2, \dots, \alpha_k)G.$$

This defines our special encoding function:

$$E((\alpha_1, \alpha_2, \dots, \alpha_k)) = (\alpha_1, \alpha_2, \dots, \alpha_k)G.$$

In other words,

$$E : Z_2^k \to Z_2^n$$

$$(a_1, a_2, \dots, a_k) \to (a_1, a_2, \dots, a_k)G$$

Let $F = \{E \mid E(x) = xG \text{ for some } G\}$. Each $E \in F$ is determined by its matrix G, so decoding E reduces to recovering G.

3.2 Special Encoding Functions and Neural Networks: Origins of the Deep Learning Decoding Model

Encoding functions in F perform matrix operations $E(x) = xG$, resembling the input-output structure of neural networks. The matrix multiplication is analogous to the forward propagation in deep learning—indeed, it is the simplest form of forward propagation.

This similarity inspires the hypothesis: Can deep learning models approximate matrix G, thereby decoding the encoding function E? Since recovering G decodes E, this proposes a novel decoding approach using neural networks.

3.3 Constructing the Deep Learning Decoding Model

Analyzing the matrix operation:

$$E((\alpha_1, \alpha_2, \dots, \alpha_k)) = (\alpha_1, \alpha_2, \dots, \alpha_k)G = (x_1, x_2, \dots, x_n),$$

we have:

$$\begin{pmatrix} v_{11} & v_{12} & \dots & v_{1n} \\ v_{21} & v_{22} & \dots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{k1} & v_{k2} & \dots & v_{kn} \end{pmatrix}$$

$$= (x_1, x_2, \ldots, x_n)$$

then:

$$\begin{aligned} a_1v_{11} + a_2v_{21} + \cdots + a_kv_{k1} &= x_1 \\ a_1v_{12} + a_2v_{22} + \cdots + a_kv_{k2} &= x_2 \\ &\vdots \\ a_1v_{1n} + a_2v_{2n} + \cdots + a_kv_{kn} &= x_n \end{aligned}$$

Each column vector of G independently affects a component of the codeword. Consider the first equation: $a_1v_{11} + a_2v_{21} + \cdots + a_kv_{k1} = x_1$:

$$(a_1, a_2, \ldots, a_k) \begin{pmatrix} v_{11} \\ v_{21} \\ \vdots \\ v_{k1} \end{pmatrix} = x_1$$

corresponding to a neural network as in Fig. 2,

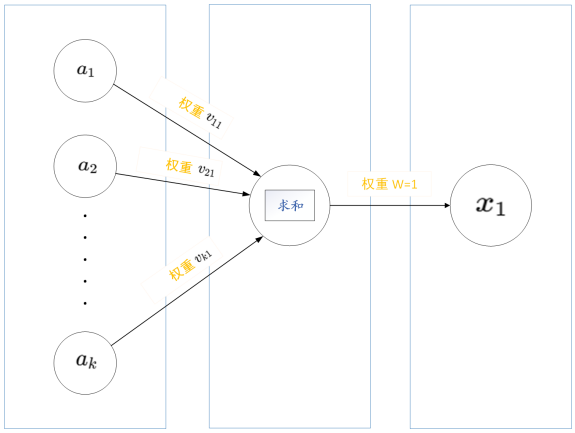


图 2: single neural network

where the weight vector $(v_{11}, v_{21}, \ldots, v_{k1})$ has components in $[0, 1]$ (approximating the binary G, though neural network weights are real numbers in $[0, 1]$, discussed later).

For hidden-to-output layer propagation, the sum may not be 0 or 1, so we use an activation function: $f(x) = x \mod 2$, chosen to retain binary properties, unlike $f(x) = x \mod 1$, with deeper implications for handling real-world errors.

Initially, a cross-entropy loss was considered: $\text{Loss} = -y\log(y_p)$, but it failed for $y = 0$. A modified loss $\text{Loss} = -(y - y_p)^2 \log(y - y_p)^2$ had issues with infinity at $(y - y_p)^2 = 1$. The final choice was:

$$\text{Loss} = (y - y_p)^2 e^{(y-y_p)^2},$$

balancing error representation and training feasibility.

Extending to all columns, we build a neural network as in Fig. 5, where each column of G is approximated by a sub-network.

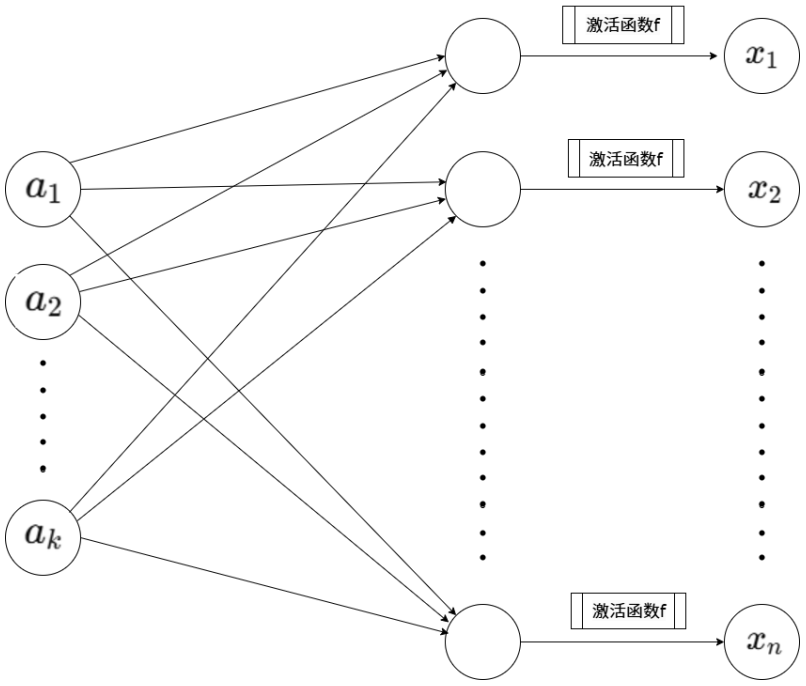


图 3: Neural Network

3.4 Theoretical Explanation and Achievements of the Deep Learning Decoding Model

A key question: With abundant input-output pairs (a, x), why use deep learning instead of direct computation? The answer lies in real-world imperfections:

$$aG = x \quad \text{(ideal computation)},$$
$$a\overline{G} = \overline{x} \quad \text{(real-world transmission)},$$

where $\overline{G}$ and $\overline{x}$ include errors. Training on (a, x) approximates G; on $(a, \overline{x})$, it approximates $\overline{G}$. Let N be the network, n the number of input-output pairs: $N \to G$ as $n \to \infty$ for ideal data, and $N \to \overline{G}$ for real data.

Mathematically, G has binary components, implying determinism, while neural network weights $\overline{v}_i \in [0, 2]$ (relaxed to real numbers) model real-world uncertainties. The activation function $f(x) = x \mod 2$ maps continuous outputs to binary, bridging the deterministic code theory and probabilistic deep learning.

These insights form the foundation for a model that potentially outperforms idealized mathematical decoding by incorporating real-world noise, motivating empirical validation.

4 Decoding Model Construction and Effect Analysis

4.1 Model Improvements and Construction

Initial model performance was poor due to four issues: (1) discontinuous activation function, (2) complex loss function, (3) insufficient data, (4) shallow network. We address these:

Activation Function Analysis and Improvement: The mod 2 function is discontinuous, non-differentiable, and limits learning. We replace it with the Sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}},$$

which is continuous, differentiable, and maps outputs to $[0, 1]$, enabling stable gradient descent and richer feature representation (Fig. 4).

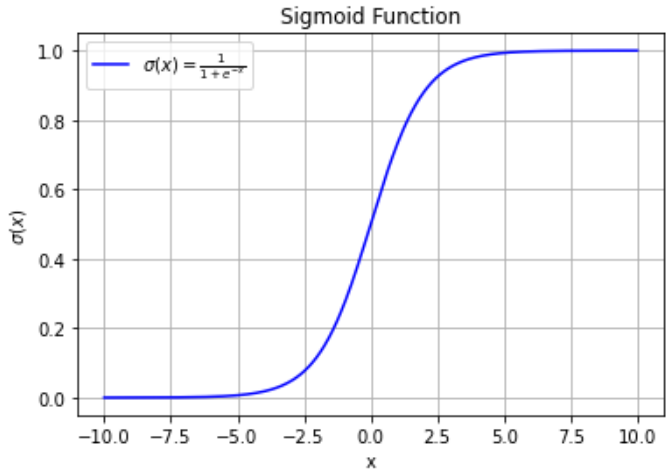


图 4: Sigmoid Function

Analysis and Improvement of Loss Function:

The loss function

$$\textbf{Loss} = (y - y_p)^2 e^{(y-y_p)^2}$$

exhibits exponential growth, which is excessively rapid. This leads to samples with large errors contributing disproportionately to the loss function during training, potentially overshadowing the learning signals from other samples. Additionally, it may cause gradient explosion problems - especially when errors are large, both the loss and gradients can become extremely large, destabilizing the training process. There is also an issue with gradient computation: the derivative of this loss function is

$$\frac{\partial \mathbf{L}}{\partial y_p} = -2(y - y_p)e^{(y-y_p)^2} - 2(y - y_p)^3 e^{(y-y_p)^2}$$

making gradient calculation highly complex. In deep learning models, where we compute the loss function's derivative many times, using this function imposes significant computational pressure on the convergence of the decoding model and slows down convergence, i.e., the following condition does not hold:

$$n \to \infty$$

Thus, we consider the Mean Squared Error (MSE) function, whose mathematical expression is:

$$\text{Loss} = (y_i - y_{p_i})^2$$

We create an MSE function plot with the true value of 0 and, as shown in Figure 5.

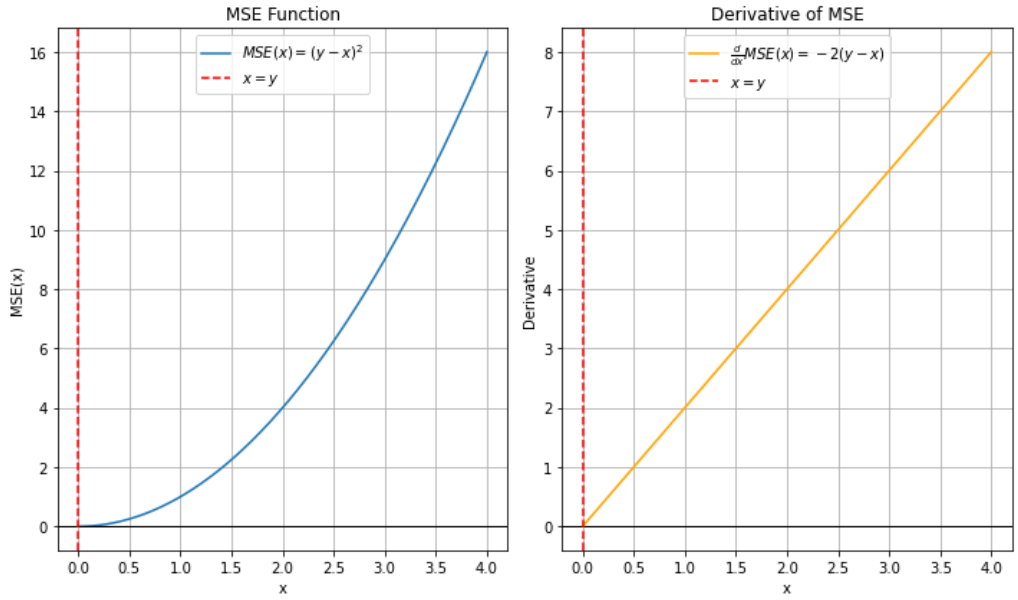


图 5: MSE and its derivative

It has good mathematical properties: The Mean Squared Error (MSE) is a continuous and differentiable function, and its gradient is simple and usually smooth. This is very important for the optimization process. Moreover,

$$\frac{\partial \text{Loss}}{\partial y_{p_i}} = -2(y_i - y_{p_i})$$

Since the derivative of MSE is very simple and easy to calculate, it will not cause problems such as gradient explosion or vanishing like some other complex loss functions. Therefore, using such a continuous, differentiable and relatively simple loss function allows the decoding model to perform calculations more quickly, ensuring that

$$n \to \infty \quad \text{holds}$$

Insufficient Data Provided:

Initially, we used $a \in Z_2^6$ to provide data, which could provide at most $2^6 = 64$ data points. Considering the current situation, this amount is still relatively small. So we expanded it to 10 - dimensional or even 20 - dimensional data. The training results show that the training speed and effectiveness are both excellent when using 10 - dimensional to 20 - dimensional data.

Shallow Neural Network:

The neural network we initially set up was too shallow, resulting in insufficient expressive ability. The expressive ability of a neural network is directly related to its depth (number of layers). A deep neural network with more layers can learn more complex features and mappings. A shallow network lacks sufficient complexity to capture and learn the features in the data. Meanwhile, a too - shallow neural network leads to a weak feature extraction ability of the decoding model, which is prone to underfitting and unable to effectively learn from the data. Additionally, it has insufficient ability to learn complex functions. Due to the small number of layers and poor expressive ability, a shallow neural network cannot fit these

complex patterns. At the same time, our neural network should increase the number of layers to adapt to the modified and more complex activation functions. Therefore, we increased the number of hidden layers of the neural network to 2, with the number of nodes in the first layer increased to 64 and the second layer increased to 32.

After solving these problems, we reconstructed the decoding model, trained it, and made predictions, and obtained a series of very good results.

4.2 Analysis of Decoding Model Effect

Finally, we attempt to decode a class of decoding functions:

$$E : Z_2^{20} \to Z_2^n$$

$$(a_1, a_2, \ldots, a_{20}) \to (a_1, a_2, \ldots, a_{20})G$$

where $a \in Z_2^{20}$. That is, we construct a deep - learning decoding model on Z_2^{20} to decode G.

Based on the above analysis, we know that the specific steps to decode G can be to solve each column vector of G, and finally we obtain G. Let $G = (v_1, v_2, v_3, \ldots, v_n)$. We can use n neural networks $N_1, N_2, \ldots, N_n$ to approximate $(v_1, v_2, v_3, \ldots, v_n)$ respectively.

$$\begin{aligned} \lim_{n\to\infty} N_1 &= v_1 \\ \lim_{n\to\infty} N_2 &= v_2 \\ &\vdots \\ \lim_{n\to\infty} N_n &= v_n \end{aligned}$$

So we have:

$$\lim_{n\to\infty} (N_1, N_2, \ldots, N_n) = (v_1, v_2, \ldots, v_n) = G$$

In this way, by solving each column, we can obtain G, and thus obtain E.

We take the following example

$$v = (1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1)$$

to verify the effectiveness of our decoding model and intuitively feel the excellent approximation effect of the decoding model. The code part is placed in the appendix, and here we only show the change graph of the Loss function, the final value, and the prediction effect.

Loss during Training

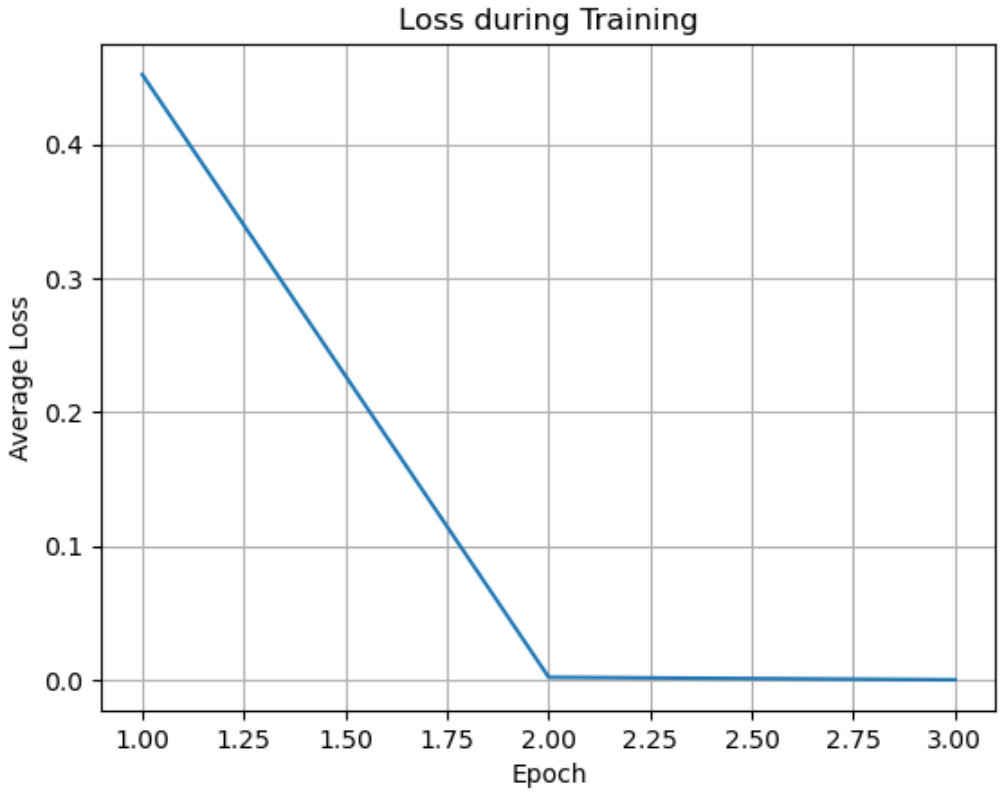


图 6: loss with epoch

From the figure, it is not difficult to observe that during the training process, as the number of epochs increases, the loss function gradually decreases and tends to be stable. Especially when the number of epochs reaches 30, the value of the loss function is already close to zero. This indicates that the training of the model has reached a relatively ideal state at this point. The very low loss value shows that the model can accurately fit the training data in most cases. Specifically, we have:

$$\lim_{\text{epoch}\to\infty} \text{loss} \to 0$$

This means that as the training continues, the loss function approaches zero, and the performance of the model on the training data is gradually optimized, enabling it to better capture the inherent patterns in the data.

5 Final Value of Loss

```
Epoch 1/20
Epoch 1: 100%|██████████| 16384/16384 [04:15<00:00, 64.05it/s, loss=0.0346]
Epoch 1 finished. Average Loss: 0.452231
Epoch 2/20
Epoch 2: 100%|██████████| 16384/16384 [04:28<00:00, 60.99it/s, loss=0.000158]
Epoch 2 finished. Average Loss: 0.002019
Epoch 3/20
Epoch 3: 100%|██████████| 16384/16384 [03:45<00:00, 72.77it/s, loss=0.000163]
Epoch 3 finished. Average Loss: 0.000113
```

图 7: Final Plot of Loss Function (epoch=3)

At epoch=3, the value of the loss function drops to 0.0001113, which is very close to zero. This value indicates that the prediction error of the model is extremely small, and the model's fitting on the training set has approached perfection. At this point, the gap between the predicted values and the true values is minimal, suggesting that the model has fully learned the features of the data and possesses strong generalization ability.

Next, we analyze the specific effects of ten randomly generated prediction sets. The following ten test examples $x_1, x_2, \ldots, x_9, x_{10}$ are obtained by performing operations between four 20-dimensional vectors and a certain column vector $\mathbf{v}$ in the decoding model, where:

$$\mathbf{v} = (1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1)$$

These vectors generate predicted values through inner product operations with a column of the decoding model. The fact that these predicted values are close to the actual expected outputs indicates that the model can successfully learn the mapping relationships in the data and accurately predict new inputs.

6 Prediction Effect

```
Make predictions:
Prediction results:
Sample 1 predicted value: 0, true value: 0
Sample 2 predicted value: 1, true value: 1
Sample 3 predicted value: 0, true value: 0
Sample 4 predicted value: 1, true value: 1
Sample 5 predicted value: 0, true value: 0
Sample 6 predicted value: 1, true value: 1
Sample 7 predicted value: 1, true value: 1
Sample 8 predicted value: 0, true value: 0
Sample 9 predicted value: 0, true value: 0
Sample 10 predicted value: 0, true value: 0
Test accuracy: 100.00%
```

图 8: Prediction Effect

Overall, through effective training, the decoding model has been able to model the relationship between input data and target outputs with high accuracy. The plot and final value of the loss function show that the training of the decoding model is close to convergence, and the prediction effect has been significantly improved, indicating that the model has strong fitting ability and good generalization performance.

7 Summary of Research Results

7.1 Deep Learning Decoding Model for a Special Class of Coding Functions

This paper delves into a special class of coding functions and reveals their striking similarity to the forward propagation process in neural networks and deep learning. Based on this discovery, we put forward a bold hypothesis: Can deep learning models effectively approximate such coding functions, thereby opening up new research directions for the integration of coding theory and deep learning models?

In the construction of the decoding model, we selected a unique and ingenious activation function, successfully addressing the key differences between group code theory and the approximation of deep learning models. The MSE (Mean Squared Error) function was chosen as the loss function for this decoding model. The clever selection of the loss function complements the activation function, resolving the structural discrepancies between coding theory and the construction of deep learning approximation models, while enhancing the model's functionality and adaptability.

参考文献

[1] Y. Pan, C. Lyu, Z. Yang, et al. E-code: Mastering efficient code generation through pretrained models and expert encoder group. *Information and Software Technology*, 178:107602, 2025.

[2] 刘刚，张杲峰，周庆国. *Introduction to artificial intelligence*. 北京邮电大学出版社, 北京, 1nd edition, 2020.

[3] 郭聿琦，冯爱芳，雷鹏. 抽象代数基础（上册）——代数学引论. 科学出版社, 北京, 2nd edition, 2019.

[4] 阮传概，孙伟. 近世代数及其应用. 北京邮电大学出版社, 北京, 1nd edition, 2001.

[5] 魏溪含，涂铭，张修鹏. *Deep Learning and Image Recognition Principle and Practice*. 机械工业出版社, 北京, 1nd edition, 2020.

A Appendix

Conda Environment Configuration Table

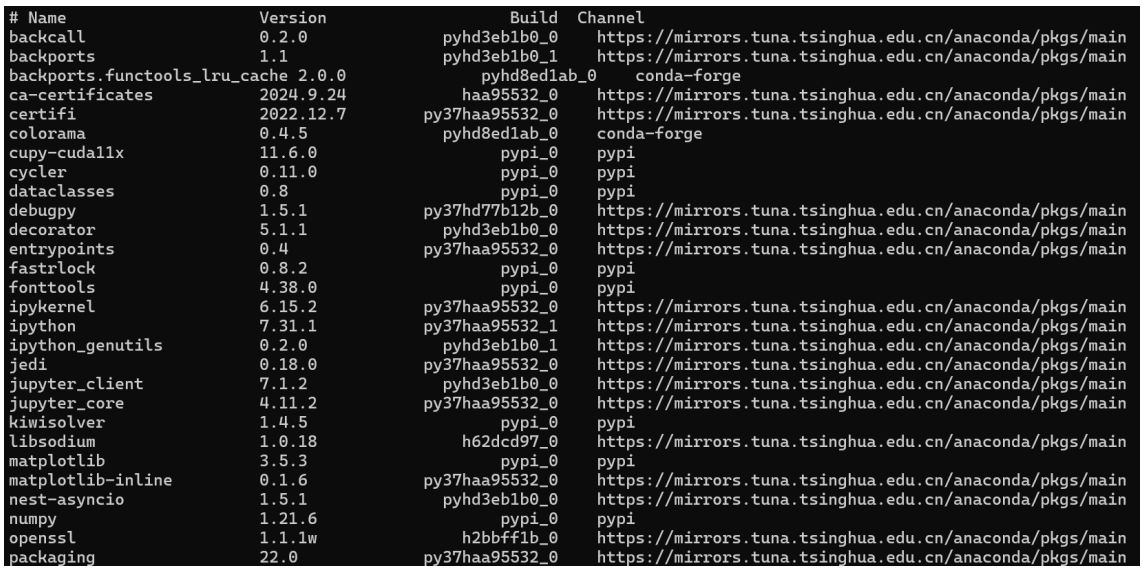
```
     active environment : btlf0
    active env location : C:\Users\86182\anaconda3\envs\btlf0
            shell level : 1
       user config file : C:\Users\86182\.condarc
 populated config files : C:\Users\86182\.condarc
          conda version : 24.9.2
    conda-build version : 24.5.1
         python version : 3.12.4.final.0
                 solver : libmamba (default)
       virtual packages : __archspec=1=skylake
                          __conda=24.9.2=0
                          __cuda=12.7=0
                          __win=0=0
      base environment : C:\Users\86182\anaconda3  (writable)
     conda av data dir : C:\Users\86182\anaconda3\etc\conda
  conda av metadata url : None
          channel URLs : https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/win-64
                         https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/noarch
                         https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/win-64
                         https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/noarch
                         https://repo.anaconda.com/pkgs/main/win-64
                         https://repo.anaconda.com/pkgs/main/noarch
                         https://repo.anaconda.com/pkgs/r/win-64
                         https://repo.anaconda.com/pkgs/r/noarch
                         https://repo.anaconda.com/pkgs/msys2/win-64
                         https://repo.anaconda.com/pkgs/msys2/noarch
```

图 9:

```
                         https://repo.anaconda.com/pkgs/msys2/noarch
         package cache : C:\Users\86182\anaconda3\pkgs
                         C:\Users\86182\.conda\pkgs
                         C:\Users\86182\AppData\Local\conda\conda\pkgs
      envs directories : C:\Users\86182\anaconda3\envs
                         C:\Users\86182\.conda\envs
                         C:\Users\86182\AppData\Local\conda\conda\envs
              platform : win-64
            user-agent : conda/24.9.2 requests/2.32.2 CPython/3.12.4 Windows/11 Windows/10.0.22631 solver/libmamba cond
a-libmamba-solver/24.1.0 libmambapy/1.5.8 aau/0.4.4 c/. s/. e/.
         administrator : False
            netrc file : None
          offline mode : False
```

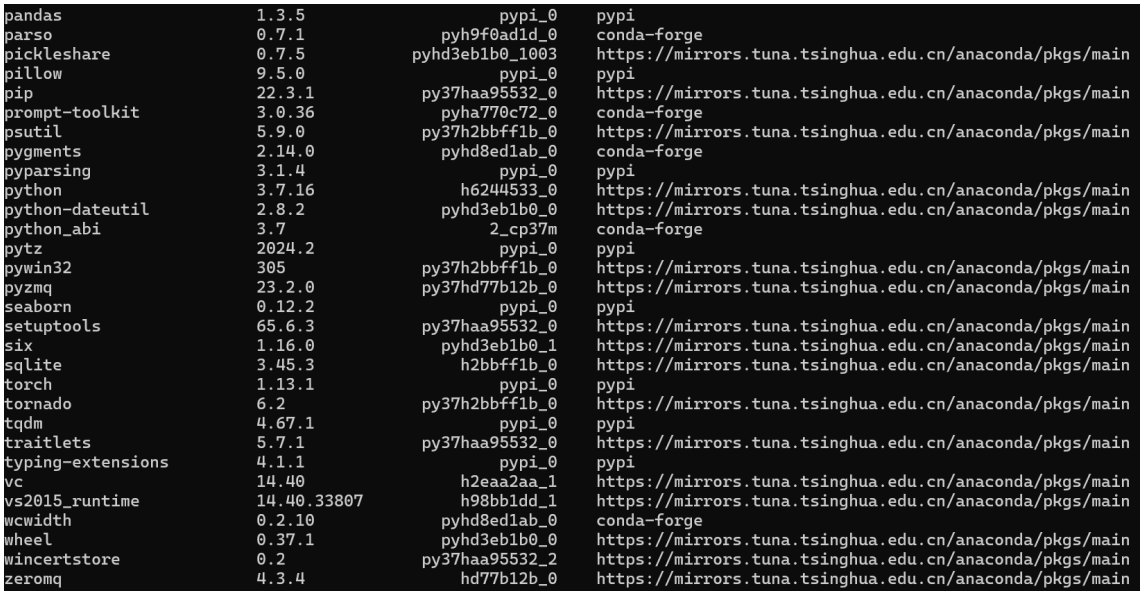
图 10:

Installation Package Configuration Table



```
# Name                    Version                 Build  Channel
backcall                  0.2.0              pyhd3eb1b0_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
backports                 1.1                pyhd3eb1b0_1    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
backports.functools_lru_cache 2.0.0            pyhd8ed1ab_0    conda-forge
ca-certificates           2024.9.24               haa95532_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
certifi                   2022.12.7          py37haa95532_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
colorama                  0.4.5              pyhd8ed1ab_0    conda-forge
cupy-cuda11x              11.6.0                    pypi_0    pypi
cycler                    0.11.0                    pypi_0    pypi
dataclasses               0.8                       pypi_0    pypi
debugpy                   1.5.1              py37hd77b12b_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
decorator                 5.1.1              pyhd3eb1b0_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
entrypoints               0.4                py37haa95532_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
fastrlock                 0.8.2                     pypi_0    pypi
fonttools                 4.38.0                    pypi_0    pypi
ipykernel                 6.15.2             py37haa95532_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
ipython                   7.31.1             py37haa95532_1    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
ipython_genutils          0.2.0              pyhd3eb1b0_1    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
jedi                      0.18.0             py37haa95532_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
jupyter_client            7.1.2              pyhd3eb1b0_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
jupyter_core              4.11.2             py37haa95532_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
kiwisolver                1.4.5                     pypi_0    pypi
libsodium                 1.0.18                  h62dcd97_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
matplotlib                3.5.3                     pypi_0    pypi
matplotlib-inline         0.1.6              py37haa95532_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
nest-asyncio              1.5.1              pyhd3eb1b0_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
numpy                     1.21.6                    pypi_0    pypi
openssl                   1.1.1w                  h2bbff1b_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
packaging                 22.0               py37haa95532_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
```

图 11:



```
pandas                    1.3.5                     pypi_0    pypi
parso                     0.7.1                pyh9f0ad1d_0    conda-forge
pickleshare               0.7.5             pyhd3eb1b0_1003    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
pillow                    9.5.0                     pypi_0    pypi
pip                       22.3.1             py37haa95532_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
prompt-toolkit            3.0.36               pyha770c72_0    conda-forge
psutil                    5.9.0              py37h2bbff1b_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
pygments                  2.14.0             pyhd8ed1ab_0    conda-forge
pyparsing                 3.1.4                     pypi_0    pypi
python                    3.7.16                  h6244533_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
python-dateutil           2.8.2              pyhd3eb1b0_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
python_abi                3.7                    2_cp37m    conda-forge
pytz                      2024.2                    pypi_0    pypi
pywin32                   305                py37h2bbff1b_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
pyzmq                     23.2.0             py37hd77b12b_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
seaborn                   0.12.2                    pypi_0    pypi
setuptools                65.6.3             py37haa95532_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
six                       1.16.0             pyhd3eb1b0_1    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
sqlite                    3.45.3                  h2bbff1b_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
torch                     1.13.1                    pypi_0    pypi
tornado                   6.2                py37h2bbff1b_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
tqdm                      4.67.1                    pypi_0    pypi
traitlets                 5.7.1              py37haa95532_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
typing-extensions         4.1.1                     pypi_0    pypi
vc                        14.40                   h2eaa2aa_1    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
vs2015_runtime            14.40.33807             h98bb1dd_1    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
wcwidth                   0.2.10             pyhd8ed1ab_0    conda-forge
wheel                     0.37.1             pyhd3eb1b0_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
wincertstore              0.2                py37haa95532_2    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
zeromq                    4.3.4                   hd77b12b_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
```

图 12: