Distributed Arithmetic Coding Over Binary Erasure Channel

Xianyu Wang

Xidian University

Cong Li

Academy of Space Electronic Information Technology

Jinlin Tan

Xidian University School of Aerospace Science and Technology

Rui Zhang

Shaanxi Aerospace Technology Application Research Institute Co. Ltd

Zhifeng Liang

Shaanxi Aerospace Technology Application Research Institute Co. Ltd

Hai Wang (✉ 184349256@qq.com)

Xidian University

Research Article

Keywords: Binary erasure channel, Distributed arithmetic coding, Deep residual network

Posted Date: November 30th, 2021

DOI: <https://doi.org/10.21203/rs.3.rs-1090955/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Distributed Arithmetic Coding Over Binary Erasure Channel

Xianyu Wang^{1,2}, Cong Li², Jinlin Tan^{3,4}, Rui Zhang⁴, Zhifeng Liang⁴ and Hai Wang^{3*}

*Correspondence:

wanghai@mail.xidian.edu.cn

³School of Aerospace Science and Technology, Xidian University, 710071 Xi'an, China

Full list of author information is available at the end of the article

Abstract

In this paper, the Binary Erasure Channel (BEC) is researched by Distributed Arithmetic Coding (DAC) based on Slepian-Wolf coding framework. The source and side information are modelled as a virtual BEC. The DAC decoder uses *maximum a posteriori* (MAP) as the criterion to recover the source. A deep residual network is used to boost the DAC decoding process. The experimental results show that our algorithm nearly achieves the same performance with LT codes under different erasure probabilities.

Keywords: Binary erasure channel; Distributed arithmetic coding; Deep residual network

1 Introduction

Binary Erasure Channel (BEC) [1] is a very simple communication model proposed by Elias [2] in 1955. In BEC, a bit is either transmitted correctly or erased with probability p_e , in which the receiver knows exactly which bits are erased. The erasure channel is of great importance in communication industry. For example, if we want to send files over internet, a file is first chopped into packets, and each packet is either received with no error or missed. We can model this situation with an erasure channel, in which a probability $1-p_e$ of input is transmitted without error and probability p_e of delivering the output 'e'. A common method to this problem is to employ an Automatic Repeat Request (ARQ) technique. When the receiver realizes the missing packets, it sends back messages, and sender retransmits the erased packet. Obviously, ARQ may lead to large overhead and network congestion.

To address these shortcomings, many attentions have been paid to erasure-correcting codes. Reed-Solomon (RS) codes [3] is first studied over erasure channel. As an important class of Maximum Distance Separable (MDS) codes, an (N, K) RS codes can recover original K source symbols if any K of N transmitted symbols are received. But RS codes are practical only for small N and K . Luby et al. [4, 5] proposed fountain codes for erasure channel and realized the fountain codes as LT codes [6, 7], which has demonstrated excellent performance over erasure channel. In addition, the encoding/decoding complexity of LT codes is much smaller than that of RS codes.

All above works are performed by channel codes. In this letter, we are going to develop a different scheme, *i.e.*, the source code model over BEC. Distributed Arithmetic Coding (DAC) [8, 9] is a variant of Arithmetic Coding (AC), which is considered the most important lossless source coding technique. DAC could implement the distributed source coding [10], *i.e.*, Slepian-Wolf coding, and outperforms

channel codes for short-to-medium length data blocks and for biased source. Chen et al. [11, 12] proved that at high rates nonlinear Slepian-Wolf codes performs better than linear counterpart in terms of rate-error tradeoff. As a non-linear Slepian-Wolf cods, DAC has the potential to beat channel codes.

To the best of our knowledge, DAC over BEC has not been investigated yet. The major contributions of this paper can be summarized:

- We construct a DAC model, in which source and side information are correlated by BEC, and use *maximum a posteriori* (MAP) criterion to recover the source.
- We proposed the decoding algorithm of DAC based on Deep Residual Network (ResNet) to deal with the large amount of side information paths in the decoding process.
- We compared three algorithms, *i.e.*, DAC, DAC-ResNet and LT codes by numerical simulation experiments.

The rest of this paper is organized as follows. The model is described in Sect. II. The standard decoding algorithm is shown in Sect. III. The decoding algorithm based on ResNet is described in Sect. IV. The experimental results are demonstrated in Sect V, and Sect VI concludes the paper.

2 Methods

Let source $X^n \triangleq (X_0, X_1, \dots, X_{n-1})$ and side information $Y^n \triangleq (Y_0, Y_1, \dots, Y_{n-1})$ be two blocks of n independent and identically-distributed (i.i.d) random variables. The *probability mass function* (pmf) of X_i is $\Pr(X_i = 1) = p$ and $\Pr(X_i = 0) = 1 - p$, where p is the bias probability. In this paper, only equally-likely source is considered, hence $p=0.5$. Under Slepian-Wolf framework, DAC encoder compresses source X^n independently with the required rate, and side information Y^n is only available losslessly at the DAC decoder. Our model is illustrated in Fig.1.

Let $\mathbb{B} \triangleq \{0, 1\}$ and $\mathbb{E} \triangleq \{0, 1, e\}$, where e is the erasure symbol, be two sets. Let $x^n \in \mathbb{B}^n$ and $y^n \in \mathbb{E}^n$, be the realization of X^n and Y^n respectively. Encoder compresses source x^n into codeword C_X in the absence of y^n . According to Slepian-Wolf theorem, if the rate $R \geq H(X|Y)$ bits per symbol (bps), x^n can be losslessly recovered. X_i and Y_i are correlated with conditional probabilities $\Pr(Y = 0|X = 0) = 1 - p_e$, $\Pr(Y = 0|X = 1) = 0$, $\Pr(Y = 1|X = 0) = 0$, $\Pr(Y = 1|X = 1) = 1 - p_e$, $\Pr(Y = e|X = 0) = p_e$, and $\Pr(Y = e|X = 1) = p_e$, where p_e is the erasure probability. The correlation between X_i and Y_i can be modelled as a virtual BEC model, with X_i being the input and Y_i being the output of the channel as shown in Fig.2.

We first briefly introduce the DAC model, and see [13] for the detailed description. The encoder of DAC compresses x^n by iteratively mapping source symbols 0 and 1 into partially-overlapped intervals $[0, (1 - p)^\gamma]$ and $[(1 - p)^\gamma, 1]$ respectively, where $\gamma \in (0, 1)$ is the overlapping factor. After all n source symbols have been compressed, we can obtain the final interval I and any number in this interval can be selected as the codeword C_X . Thanks to the overlap, the rate R is reduced to $R = -p \log_2 p^\gamma - (1 - p) \log_2 (1 - p)^\gamma = \gamma H(X)$. To achieve lossless recovery, we must set $R \geq H(X|Y)$, hence $\gamma \geq H(X|Y)/H(X)$.

3 Standard Decoding Over BEC

The standard decoding process of DAC is to recover the sequence x^n by the codeword C_X with the help of side information y^n , and the MAP is applied as the criterion:

$$\hat{x}^n = \arg \max P(X^n | C_X, Y^n), \quad (1)$$

where $\hat{x}^n$ is the recovered source. The decoding algorithm is listed in Algorithm 1. In the next 3 subsections, we will explain this algorithm.

Algorithm 1: Algorithm 1: DAC decoder over BEC

Input: C_X, y^n, γ

Output: $\hat{x}^n$

- 1 Use C_X and γ to generate code tree τ_c with n_c paths;
 - 2 Use y^n to generate side information tree τ_s with n_s paths;
 - 3 Use MAP criterion to recover the source $\hat{x}^n$
-

3.1 Generating Code Tree

The step 1 in Algorithm 1 generates a depth- n incomplete binary tree τ_c , in which each node is characterized by 4 parameters: *low*, *high*, *code*, and *u*. *low/high* is the lower/higher bound of the interval and *code* is the codeword in the buffer. Initially, for the root node, *low*=0, *high*=1, and *code*= C_X . *u* is calculated by:

$$u = \frac{\text{code} - \text{low}}{\text{high} - \text{low}}. \quad (2)$$

Obviously $0 \leq u < 1$. Based on the value of *u*, Decoder takes one of the following three actions:

1. $0 \leq u < (1 - p^\gamma)$: determine the current source symbol to 0 and generate 0-branch
2. $(1 - p^\gamma) \leq u < (1 - p)^\gamma$: cannot determine, so generate both 0-branch and 1-branch
3. $(1 - p)^\gamma \leq u < 1$: determine the current source symbol to 1 and generate 1-branch

The parameters of each branch are then updated:

1. 0-branch: $\text{high} = \text{low} + (\text{high} - \text{low})(1 - p)^\gamma$
2. 1-branch: $\text{low} = \text{low} + (\text{high} - \text{low})(1 - p^\gamma)$

The code tree is formed in a breadth-first manner. After n stages the code tree is completed with n_c paths, each path will correspond to a sequence of n binary symbols. We can find that with the increase of code length n , the number of paths grows exponentially. In order to reduce the computational complexity and not to exhaust memory, the code tree only retains the first M paths and additional branches will be pruned. M is a predefined parameter. To avoid pruning the correct path, we set M to be large enough and n to be small enough.

3.2 Generating Side Information Tree

The side information y^n will form a depth- n incomplete binary tree τ_s , in which each node is generated by one of following tree actions:

1. $y_i = 0$: determine the current side information symbol to 0 and generate 0-branch
2. $y_i = 1$: determine the current side information symbol to 1 and generate 1-branch;
3. $y_i = e$: cannot determine, so generate both 0-branch and 1-branch

For example, if $y^n = (0, 1, e, 1, 0)$ with $n=5$, the corresponding side information tree τ_s is illustrated in Fig. 3.

After n stages the side information tree is also completed with n_s paths, each path will correspond to a sequence of n binary symbols. Following the same setup in above subsection, side information tree τ_s also retains the first M paths.

3.3 Recovering the Source

Denote the j -th path of code tree τ_c as $\mathbf{c}_j = \{c_{ji}\}_{i=0}^{n-1}$, where c_{ji} is the i -th symbol of the j -th path. Denote the u -th path of side information tree τ_s as $\mathbf{s}_u = \{s_{uv}\}_{v=0}^{n-1}$, where s_{uv} is the v -th symbol of the u -th path. Given $\mathbf{s}$, the *a posteriori* probability of $\mathbf{c}_j$ is:

$$\Pr(\mathbf{c}_j|\mathbf{s}_u) = \prod_{i=0, v=0}^{n-1} \Pr(c_{ji}|s_{uv}), \quad (3)$$

where

$$\Pr(c_{ji}|s_{uv}) = \begin{cases} p_e, & c_{ji} \neq s_{uv} \\ 1 - p_e, & c_{ji} = s_{uv} \end{cases} \quad (4)$$

After all paths are calculated, the path with the MAP probability is chosen as the recovered source $\hat{x}^n$.

4 Decoding based on ResNet

Recall the standard decoding process in Sect. 3, and we find that the number of paths in the side information tree τ_s grows exponentially with the increase of the code block length. This situation leads to the decoding process very complex and poor performance. To address this problem, we employ a deep learning technique, *i.e.*, Deep Residual Network (ResNet) to find out the MAP path.

4.1 Network Architecture of ResNet

The ResNet [14] was proposed by He et al. in 2016, which is a variant of convolutional neural network with ability to extract high level features of raw data. ResNet has been widely used in many areas of computer science, such as, object identification, semantic image segmentation, nature language processing and so on. ResNet could achieve the same performance with the traditional convolutional neural network but lower computing complexity. For example, comparing 19.6 billion FLOPS operations (multiply-adds) of VGG-19 [15], ResNet-26 has only 2.8 billion FLOPS.

Our proposed ResNet-26 has 26 layers, with one 7×7 convolutional layer, one fully connected layer with softmax, and 8 residual building blocks, each of which includes 3 convolutional layers with shortcut connections. The input of ResNet-26 is a vector with length $K = 7 \times 7$. The network architecture is shown in Fig.4.

4.2 Implementation

The side information and code tree with length n are resized into $\lfloor \frac{n}{K+1} \rfloor$ segments. The length of each segment is $K+1$ and all segments are feed into ResNet-26. The weights of all residual nets are initialized from scratch. We take MAP as the loss function, which is defined as

$$\begin{aligned} L &= \frac{1}{n} \sum_{j=0, u=0}^{n-1} \Pr(c_j | s_u) \log_2 \frac{1}{\Pr(c_j | s_u)} \\ &= \frac{1}{n} \sum_{j=0}^{n-1} \prod_{i=0, v=0}^{n-1} \Pr(c_{ji} | s_{uv}) \log_2 \frac{1}{\Pr(c_{ji} | s_{uv})}. \end{aligned} \quad (5)$$

Our training platform is a NVIDIA TITAN V GPU with 5120 CUDA cores. We use Adam as the optimizer, and set the batch size at 1024. The learning rate is from $1e-3$ and maximum epoch is 100. In each epoch, we update the weights of the model, if there is a decrease in L from the previous minimum value. We stop the training process after no further improvement.

5 Experiments

In this section, we present the experimental results obtained by numerical simulation. Our source codes were implemented in C++ language and running on a PC platform. The experimental parameters were set as: $p = 0.5$, $\gamma = 0.5$, $M = 4096$. Two metrics, *i.e.*, Symbol Error Rate (SER) and Frame Error Rate (FER), were used to evaluate the performance of our model. We used an equally-likely binary code with block length $n = 128$. The erasure probability p_e varies from 0.45 to 0.2 with the step $= -0.05$. We compared three models, *i.e.*, LT codes, DAC, and DAC-ResNet. The experimental data of LT codes is obtained by a C++ implementation of [6], DAC is from the algorithm in Sect. 3, and DAC-ResNet is from Sect. 4.

We demonstrate all experimental results in Fig. 5, from which we can find that with the decrease of p_e , LT codes achieve the smallest SER and FER than others. Although DAC and DAC-ResNet perform worse than LT codes, the differences between DAC-ResNet and LT codes are very small. We can also find that DAC-ResNet outperforms DAC in both SER and FER.

To investigate the computing complexity, we show the decoding time of three different model as a function of p_e in Fig. 6. The running hardware is a PC platform with an Intel Core i7 10700F@2.9GHz CPU, a 16GB Memory, and a 2T SSD hard disk. Note that the training time is not considered into the decoding time. From Fig.6 we can find that DAC need most time to perform decoding, and DAC-ResNet and LT codes almost has the same decoding complexity which is much smaller than DAC.

6 Conclusion

We have proposed a DAC model over BEC based on Slepian-Wolf coding framework. The DAC encoder compresses source without the knowledge of side information. The decoder constructs the code tree and the side information tree, and recovers the source by applying the MAP criterion. To improve the performance and reduce

the computing time, a deep learning technique, i.e., ResNet-26 was applied to the decoder. The proposed models were evaluated by the numerical experiments, and we can find DAC-ResNet has almost the same performance with LT codes under different erasure probabilities.

In the future work, we are going to extend our model from BEC to the insertion/deletion channel and investigate the performance under different parameters.

Acknowledgements

Not applicable.

Funding

Not applicable.

Abbreviations

BEC: Binary Erasure Channel; DAC: Distributed Arithmetic Coding; MAP: maximum a posteriori; ARQ: Automatic Repeat Request; RS: Reed-Solomon; MDS: Maximum Distance Separable; pmf: probability mass function; SER: Symbol Error Rate; FER: Frame Error Rate.

Availability of data and materials

The datasets used and/or in this study are available from the corresponding author on reasonable request.

Ethics approval and consent to participate

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Consent for publication

All authors agree to submit this version and claim that no part of this manuscript has been published or submitted elsewhere.

Authors' contributions

XW constructed the model and finished the writing of this manuscript. CL, JT, RZ and ZL participated in algorithm simulation. HW proofread the manuscript and gave good suggestions on the paper. All the authors read and approved the final manuscript.

Authors' information

Not applicable.

Author details

¹The State Key Laboratory of Integrated Service Networks, Xidian University, 710071, Xi'an, China. ²Academy of Space Electronic Information Technology, 710100 Xi'an, China. ³School of Aerospace Science and Technology, Xidian University, 710071 Xi'an, China. ⁴Shaanxi Aerospace Technology Application Research Institute Co., Ltd, 710100 Xi'an, China.

References

- Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley, Hoboken, New Jersey (1991)
- Elias, P.: Coding for two noisy channels. In: Proceedings of the 3rd London Symp. Information Theory, pp. 61–76 (1955)
- Lin, S., Costello, J.D.J.: Error Control Coding: Fundamentals and Applications. PrenticeHall, Upper Saddle River, New Jersey (2004)
- Byers, J.W., Luby, M., Mitzenmacher, M., A, R.: A digital fountain approach to reliable distribution of bulk data. In: Proceedings of ACM SIGCOMM, Vancouver, BC, Canada, pp. 56–67 (1998)
- Byers, J.W., Luby, M., Mitzenmacher, M.: A digital fountain approach to asynchronous reliable multicast. IEEE J. Sel. Areas Commun. **20**, 1528–1540 (2002)
- Luby, M.: Lt codes. In: Proceedings of 43rd Annu. IEEE Symp. Foundations of Computer Science, Vancouver, BC, Canada, pp. 271–280 (2002)
- MacKay, D.J.C.: Fountain codes. IEEE Communications **152**, 1062–1068 (2005)
- Grangetto, M., Magli, E., Olmo, G.: Distributed arithmetic coding. IEEE Commun. Lett. **11**, 883–885 (2007)
- Grangetto, M., Magli, E., Olmo, G.: Distributed arithmetic coding for the slepian-wolf problem. IEEE Trans. Signal Process. **57**, 1062–1068 (2009)
- Xiong, Z., L, A.D., Cheng, S.: Distributed source coding for sensor networks. IEEE Signal Process. Mag. **21**, 80–94 (2004)
- Chen, J., He, D.K., A, J.: On the duality between slepian-wolf coding and channel coding under mismatched decoding. IEEE Trans. Inf. Theory **55**, 4006–4018 (2009)
- Chen, J., He, D.K., A, J., Lastras-Montano, L.A., H, Y.E.: On the linear codebook-level duality between slepian-wolf coding and channel coding. IEEE Trans. Inf. Theory **55**, 5575–5590 (2009)
- Fang, Y.: Dac spectrum of binary sources with equally-likely symbols. IEEE Trans. Commun. **61**, 1584–1594 (2013)

14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, pp. 770–778 (2016)
15. Simonyan, K., A, Z.: Very deep convolutional networks for large-scale image recognition (2014). arXiv preprint arXiv:1409.1556

Figure 1: DAC model.

Figure 2: The correlation between X_i and Y_i .

Figure 3: An example of the incomplete binary tree generated by side information.

Figure 4: Network architecture of ResNet. (a) a building block of Residual network (b) network of ResNet-26.

Figure 5: Performance comparison of LT codes, DAC and DAC-ResNet under different p_e (a) SER (b) FER.

Figure 6: Decoding time as the function of p_e .

Figures

Figure 1

DAC model.

Figure 2

The correlation between X_i and Y_i .

Figure 3

An example of the incomplete binary tree generated by side information.

Figure 4

Network architecture of ResNet. (a) a building block of Residual network (b) network of ResNet-26.

Figure 5

Performance comparison of LT codes, DAC and DAC-ResNet under different p_e (a) SER (b) FER.

Figure 6

Decoding time as the function of p_e .