Last Theory of Data Compression (LTDC) and Rise of Re-Compression of data with Re-compressor AI-Compiler

Nitish kumar, BCA, SV Subharthi University, nitishkumar7nk11@gmail.com

January 29, 2025

Abstract

Note: in this paper, Theoretical model of re-compression will be discussed.
Last Theory of Data Compression: Data is a Sequence of Number, find a Function for it, in case of complex sequence, find longest Pattern as sub-sequence, write Function for this.
Re-Compression Theory: According to LTDC, compressed data is also a Sequence of number, So, it will be Compressible again and again.

Main Result:
Data Re-Compression possible using LTDC.

1 Keyword

Lossless Data Compression, Data Re-Compression

2 Introduction

While there are several highly efficient compression algorithms available, each with its own strengths, the need for new concepts persists. These concepts should not only optimize data compression but also enable **Re-compression-** the ability to further compress already compressed data. Ideally, this re-compression should be lossless, preserving the integrity of original data, though achieving this without loss of quality remains a challenge in many cases.
In our daily lives, **Data** has become one of the most valuable **Virtual Gems** requiring secure storage. However, the size of data is growing exponentially, crating a need for advanced hardware solutions or **Re-compression Algorithm** that can efficiently store large volumes of data.

3 Related Work

Compression relies on two main principles: **Data Repetition and Redundancy.** Traditional algorithms and tools exploit these patterns to reduce data size, while **Deep Learning** extends this by discovering patterns in data that lack obvious repetition, enabling compression in cases where traditional methods fall short.
Re-compression of compressed data using a different algorithm, whether lossless or lossy, can yield positive results for certain types of data. However, in other cases, it may lead to an increase or no reduction in the compressed data size.
I am introducing a **Theoretical model for Re-Compression with every type of data.**

4 Problem Statement

how can we develop efficient re-compression algorithms that not only reduce the size of already compressed data but also ensure that this re-compression process either preserves the original data (lossless) or provides minimal loss in quality, all while optimizing computational resources and storage requirements?

5 Definition of some keywords

Formula-Form: group of Functions by order for a Sequence.
Ex:

$$1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 1, 4, 9, 16, 25, 36, 47$$

Formula form of above Sequence:

$$f(x, 1, 11) = x, f(x, 1, 7) = x^2$$

Unstructured Sequence: which can't be defined in Formula-Form.
Ex: 1,29, 281, 2, 7, 3, 71
Pure Sequence: the Sequence which can be defined in Formula-Form.
Alp Language: it is a programming language, its source code is **Optimized Form** of Formula Form, and its size will less than Formula Form.
Target of this language:
1. Reduce file size of Formula Form and represent it as a source code.
2. out of code will be Decompressed data of Re-compression
Note: since no programming language is developed to fulfill this requirement but any language can be used to test this work.

6 Last Theory of Data Compression (LTDC)

Data is a Sequence of Symbol, look at the ASCII and Uni-Code value of text or RGB or RGBA value of pixels of image, every thing is a Sequence of number.

Even any file with any extension can be open as binary mode, which is also a Sequence.
EX: Text-'ABC':ASCII(65, 66, 67), Image- 'Pixel' :RGB(34, 133, 211)
Find a Function-form for it, in case of complex sequence, find longest Pattern as sub-sequence, write Function-Form for this which is a very complex job.
For solve Sequence in Formula-Form **Re-compressor AI-Compiler** will be use. Also, Lexicographical order and Distance measure will be use.

7 Re-compressor AI-Compiler

It translates **High level, Unstructured and complex Sequence** into **Pure Sequence** which can be a complex translating process, it will take multiple Stage: encoding, chunk, analysis, Transformation.
Main work of ai has to recognize type of object in data before start compilation like **a tree in image**, **a letter to president**, also every phase of compiler where analysis and optimization require.
Create such model which can help to reduce compiler complexity if object type already recognize: how it works- suppose if it is a tree, then definitely it have some leaves and branches, so it overall structure will already defined in such model, after this object need to fill some color, like may be tree have all leaf green color, orange color, or red color or mixture of color. same with car, if a car is recognized then it is fix that it has some structure like: it will have a body, window, door, wheel, a driver or not, so if overall structured identified then only some data remain to fill in structure.
Same with video files, where only a single object can be identify and compressed and rest part can be according to original video or can be changed by developer.

8 Phases of AI-Compiler

Original data will be use as Input (Metadata will store), then it will Encode, after that Data will be Analysis: this phase will interact with Boundary and **Report of Unstructured Data** can be Analysis and Manipulate by user also. (If Unstructured Sequence is more than Pure Sequence then process will fail and Input data will Transform and Re-Encode; again, it will come for Analysis), then Formula Form will generate (it is Compressed Form of data), then **Alp Language's code** will generate (it will be Optimized Compressed Form of data), here data is **fully compressed.**
Here, Compressed Data will be assumed as Original Data and it can be send for **Re-Compression**.
Now, **Decompression** will start, hence, compression is very complex and Original Data is very large, so, **Decompression** in traditional method will not be effective, User should have full control over Decompression Process, that's why, i have introduced **Alp language** as a programming language, when it will execute, give Original Data as output, also user can stop process whenever need.

Below a **Textual Diagram** of process is given:
1: Original Data
2: Encode
3: Analysis (Boundary, User Analysis Unstructured Data) [Encode, Transform]
4: Formula Form
5: Optimize
6: Alp Language [Re-Compression]
7: Assembly Language
8: Machine Code
9: Original Data

This Whole Encoding Process will be under **Re-Compressor AI-Compiler.**

9 Data Encoding

in this section, All symbols that have used in data, will encoded with unique numbers.

9.1 Single Symbol Encoding:

Each symbol is uniquely assigned a number ensuring consistency in encoding and decoding. Example:
01 = A, 02 = B, ..., 26 = Z. The word HELLO is encoded as: 08 05 12 12 15.

9.2 Multi-Symbol Encoding:

A meaningful multi-symbol can be encoded with simple/small unique number. Example:
(1, 2, 15, 41 = S), (2, 154, 98, 236 = M), etc.
enabling flexible data representation.

9.3 Chunk Encoding:

If any sequence is extremely large then first chunk it then encode each chunk separately, using different number system and Grammar. Chunking is necessary when symbol related to encoded number takes more spaces than symbol space. Example:
Input: ... This is a card of Computer
coding: ... 1111111, 1111112, 1111113, 1111114, 1111115, 1111116
after chunk: ... 1, 2, 3, 4, 5, 6

10 Analysis

Before this, metadata of file with some extra information(is it a Re-Compress?) will be saved, Sequence will be analysis in small chunks, it **classify pure and unstructured sequence**, if result is not positive, Input will be re-analysis. A interface is given here for user interact with unstructured data and solve it, here, AI comes to help user.

11 Boundary

Note: it is a separate project in itself.
It will define data in a **close finite data-set**. A Boundary is a finite, closed dataset structure that **defines all possible data entries created from a fixed set of symbols**, where the number of entries is determined by the formula:

$$T = S^L$$

where T = Total-Entries, S = Symbol, L = Length
allowing for repeated symbols in each combination.
Example: Suppose, (5000 x 2500), is a standard dimension for Phone's image.
Suppose, a book has (300 words) in each page, and (1200) total page.

11.1 Boundary of Boundaries:

inside this consider multiple boundary in a single boundary, means one boundary is a combination of multiple boundary.

11.2 Arranging Mode

Lexicographical order: Each sequence will be store and analysis in this way so that developer has a better control over boundary.
Mathematical Model: it will define in a cube or sphere, in the center of model Head data will be define with position (0, 0, 0), other data will defines as distance from 'head' to itself.

11.3 Store Function and Formula Form of Sequence

Each boundary will have multiple combination/sequence, separately find Function, and store in boundary's database.

12 Data Transformation

It is introducing for **Unstructured data** which will be re-shape to get pure sequence.

12.1 Base Conversion

Change number systems (e.g., decimal to binary or hexadecimal, best is encoding very high bits).

12.2 Re-Encoding

it will help to get pure sequence, in case when analysis fails.

12.3 Binary Conversion

Such file, which can't be express in a sequence directly like text, image; easily binary form can be find then it can be used as input data (binary can be converted into other number system).

13 Binary Transformation and Processing

Transform input data into binary form, create four file:
one for Binary, second for counting, third for extra information like Merging Point and breaking point of data example:
file1: [[100100011110], [100]], file2: 100000
binary data: 10101010
selector: 1, 0 (Selector gives sub-string details)
Counting: 1, 2, 1, 3, 4, 1, 1, 5
informer (index): f1[C1:12, C2:3], f2[c1:6]
merger: it will save extra information (metadata) of merging of files.

14 Assembly Language and Machine Code

Alp language is a programming language, so, it will compile like c language and Assembly code will generated then **Machine code** will generated. Here, data is ready for execution/(Decompression).

15 Method: Re-compression

Using **LTDC**, data can be compressed then suppose, **Compressed Data as Original Data**; again, use LTDC again to compress data, Repeat this process, as you need. Here, some metadata will store, that will contain information about each loop of Re-Compression which will be helpful into **Decompression**.

16 Result

OD- Original Data, CD- Compressed Data
OD = Collection(Unstructured Sequence + Pure Sequence)
CD = Collection(Unstructured Sequence + Formula-Form)
Re-compression: loop[CD = New OD],
Special-Case:
Case I.

$$OD = PureSequence = FormulaForm = CD$$

Case II. (very rare)

$$loop(CD = PureSequence)$$

Finally, Re-Compression of any type of file is possible using **LTDC.**

17 Conclusion and Summary

A target size can be set, at which you want to compress all your files using LTDC. Multiple can merge together and this merged file can compress directly. Every file metadata will be saved in a other file. In this case, if metadata file contains a large size due to multiple file merge, this file also can be compress, using same method.
Input data can be in any file format, with any size; Output depends on user's **Re-Compression decision**.
It has two main point:
1. Re-compression creates complexity.
2. All files can be **Compress** at a certain constant size (1MB or any constant).

Re-compression is a challenging job in computer world specially when we have different type of data(files), generating re-compression algorithm for each type of file is not a better solution, Focusing on a such idea which can handle multiple type of files is a better option.
So, I suggest a idea where consider each **type of file** in a **Sequence form** and Create a **Re-Compressor AI-Compiler** which can understand pattern of data, also can create a Single Function for that sequence(file).
An AI-Compiler can Re-compress any type of data.

Other use: any message can highly **coded** for security reason.

This theoretical model has not practically tested in current time but I say, this is the FUTURE of DATA.