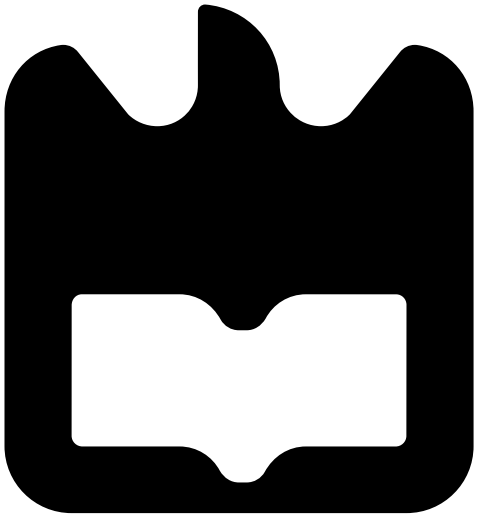

Relatório do Trabalho Laboratorial nº 3

Compressão Sem Perdas de Pesos de Modelos de Linguagem

Informação e Codificação (2025/26)

Pedro Miguel Miranda de Melo (114208)
Rúben Cardeal Costa (114190)
Hugo Marques Dias (114142)
Departamento de Eletrónica, Telecomunicações e Informática (DETI)

Universidade de Aveiro

Janeiro de 2025

Conteúdo

1 Introdução

1.1 Contexto e Motivação

Os Modelos de Linguagem de Grande Escala (*Large Language Models* – LLMs) representam um dos avanços mais significativos na área da inteligência artificial nos últimos anos. Contudo, a sua utilização prática enfrenta desafios consideráveis relacionados com o armazenamento e distribuição dos ficheiros de pesos, que frequentemente atingem dimensões na ordem dos gigabytes. A compressão eficiente destes ficheiros é, portanto, uma área de investigação com relevância prática imediata.

1.2 Objetivos do Trabalho

O presente relatório descreve o desenvolvimento de um codec especializado para a compressão sem perdas (*lossless*) do ficheiro `model.safetensors`, que contém os parâmetros do modelo Qwen2-0.5B disponibilizado pela Alibaba Cloud. Com aproximadamente 942 MB, este ficheiro constitui um caso de estudo representativo dos desafios de compressão de pesos de LLMs.

Os objetivos específicos deste trabalho são:

1. **Maximizar a taxa de compressão** através de uma análise profunda da estrutura e estatística dos dados;
2. **Manter tempos de processamento competitivos** face aos compressores de uso geral;
3. **Controlar o consumo de memória** para permitir a execução em sistemas com recursos limitados;
4. **Oferecer múltiplos pontos de operação** que permitam ao utilizador escolher o compromisso ideal entre compressão e velocidade.

1.3 Abordagem Metodológica

A estratégia adotada baseia-se numa análise aprofundada da estrutura do formato BF16 (*Brain Floating Point 16*), que revelou características estatísticas marcadamente distintas entre os bytes mais significativos (MSB) e menos significativos (LSB) de cada valor. Esta descoberta fundamental conduziu ao desenvolvimento de uma arquitetura *split-stream* que processa cada canal de forma independente e otimizada para as suas características específicas.

1.4 Estrutura do Relatório

O relatório está organizado da seguinte forma: a Secção 2 apresenta a análise e caracterização da fonte de dados; a Secção 3 documenta o *benchmarking* de compressores existentes; a Secção 4 detalha a implementação do codec; a Secção 5 apresenta e discute os resultados experimentais; e a Secção 6 sintetiza as principais conclusões e contribuições.

O código-fonte completo está disponível em: https://github.com/Rubenc1234/IC_miniP1/tree/main/Project3.

2 Análise e Caracterização da Fonte

O desenho de um codec eficiente exige uma compreensão profunda da natureza estatística da fonte de informação. Esta secção detalha a análise teórica e experimental realizada sobre o ficheiro `model.safetensors`, desde a sua estrutura de alto nível até às propriedades estatísticas dos seus componentes individuais.

2.1 Estrutura do Ficheiro SafeTensors

O formato SafeTensors, desenvolvido pela Hugging Face, é um formato binário otimizado para o armazenamento seguro de tensores. A estrutura do ficheiro é composta por:

1. **Cabeçalho de Tamanho** (8 bytes): Um inteiro de 64 bits em formato *little-endian* que indica o tamanho do cabeçalho JSON;
2. **Cabeçalho JSON** (variável): Metadados que descrevem cada tensor, incluindo nome, tipo de dados (*dtype*), dimensões e *offsets* no *payload*;
3. **Payload Binário** (restante): Dados dos tensores armazenados de forma contígua.

A extração e análise do cabeçalho JSON revelou que os pesos estão armazenados no formato **BF16** (*Brain Floating Point 16*), uma representação numérica de 16 bits desenvolvida pelo Google para aplicações de *machine learning*.

2.2 Análise do Formato BF16

Ao contrário de inteiros de 16 bits, onde a distribuição de bits pode ser relativamente uniforme, o formato BF16 possui uma semântica específica que influencia diretamente as suas propriedades estatísticas:

- **1 bit de Sinal (S)**: Indica se o valor é positivo ou negativo;
- **8 bits de Expoente (E)**: Representam a magnitude do valor numa escala logarítmica;
- **7 bits de Mantissa (M)**: Representam a precisão fracionária do valor.

Numa organização *little-endian*, o byte menos significativo (LSB) contém os 7 bits da mantissa mais o bit menos significativo do expoente, enquanto o byte mais significativo (MSB) contém o bit de sinal e os 7 bits mais significativos do expoente.

Esta estrutura sugere a existência de correlações não-lineares e localizadas que uma análise puramente sequencial (byte-a-byte) poderá não capturar eficazmente. A hipótese de trabalho formulada nesta fase foi que os dois bytes de cada valor BF16 apresentariam características estatísticas distintas, justificando um tratamento diferenciado.

2.3 Limites Teóricos: Entropia de Shannon

O limite teórico fundamental para a compressão sem perdas é dado pela **Entropia de Shannon**. Considerando o ficheiro como uma fonte de memória nula X que gera símbolos $x \in \{0, \ldots, 255\}$, a entropia de ordem-0 é definida por:

$$H(X) = -\sum_{i=0}^{255} P(x_i) \log_2 P(x_i) \quad [\text{bits/símbolo}] \tag{1}$$

onde $P(x_i)$ representa a probabilidade de ocorrência do símbolo x_i.

2.3.1 Análise Global do Payload

Aplicando a Equação 1 à totalidade do *payload* binário (excluindo o cabeçalho), obteve-se:

Entropia de Ordem-0 Global

$$H(X) \approx 6.22 \text{ bits/byte}$$

Este valor indica que, ignorando qualquer dependência entre bytes, a compressão máxima teórica seria de apenas $\sim 22\%$ (redução de 8 para 6.22 bits por byte). Trata-se de um resultado modesto que motivou a investigação de dependências inter-simbólicas.

2.3.2 Análise de Correlação Sequencial

Para investigar a existência de dependências sequenciais, calculou-se a **Entropia Condicional** de primeira ordem, que mede a incerteza de um símbolo X_n dado o conhecimento do símbolo anterior X_{n-1}:

$$H(X_n|X_{n-1}) = -\sum_{y\in\mathcal{X}} P(y) \sum_{x\in\mathcal{X}} P(x|y) \log_2 P(x|y) \tag{2}$$

O resultado experimental obtido foi:

Entropia Condicional de Primeira Ordem

$$H(X_n|X_{n-1}) \approx 5.36 \text{ bits/byte}$$

O facto de $H(X|Y) < H(X)$ confirma a existência de correlação inter-simbólica (pelo teorema do condicionamento, que afirma que condicionar nunca aumenta a entropia). Contudo, o valor de 5.36 bits/byte permanece relativamente elevado, sugerindo que a correlação sequencial simples não é suficiente para explicar toda a redundância presente nos dados.

A nossa hipótese explicativa é que a natureza intercalada dos dados BF16 (MSB estruturado alternando com LSB ruidoso) "mascara" a verdadeira correlação entre os pesos adjacentes da rede neuronal.

2.4 Análise Estrutural Diferenciada: *Byte-Splitting*

Para validar a hipótese de que a entropia está distribuída de forma desigual entre os componentes do formato BF16, procedeu-se à separação do fluxo de dados em dois canais distintos:

- **Canal MSB**: Bytes nas posições ímpares (1, 3, 5, ...), contendo predominantemente o expoente e o bit de sinal;
- **Canal LSB**: Bytes nas posições pares (0, 2, 4, ...), contendo predominantemente a mantissa.

As entropias de ordem-0 foram recalculadas individualmente para cada canal, revelando uma disparidade notável:

Tabela 1: Comparação de Entropia por Canal após *Byte-Splitting*

Canal	Conteúdo Semântico	Entropia (H)	Característica
MSB	Expoente + Sinal	**2.71 bits/byte**	Altamente Estruturado
LSB	Mantissa	**7.96 bits/byte**	Ruído Quase Uniforme

Este resultado é particularmente significativo. O canal MSB apresenta uma entropia de apenas 2.71 bits/byte, representando um potencial de compressão de 66% (de 8 para 2.71 bits). Em contraste, o canal LSB, com entropia de 7.96 bits/byte, aproxima-se do máximo teórico de 8 bits, indicando que os dados da mantissa se comportam essencialmente como ruído aleatório.

2.4.1 Validação Visual: Histogramas de Frequência

Os histogramas de frequência apresentados nas Figuras 1 e 2 corroboram visualmente os valores numéricos obtidos.

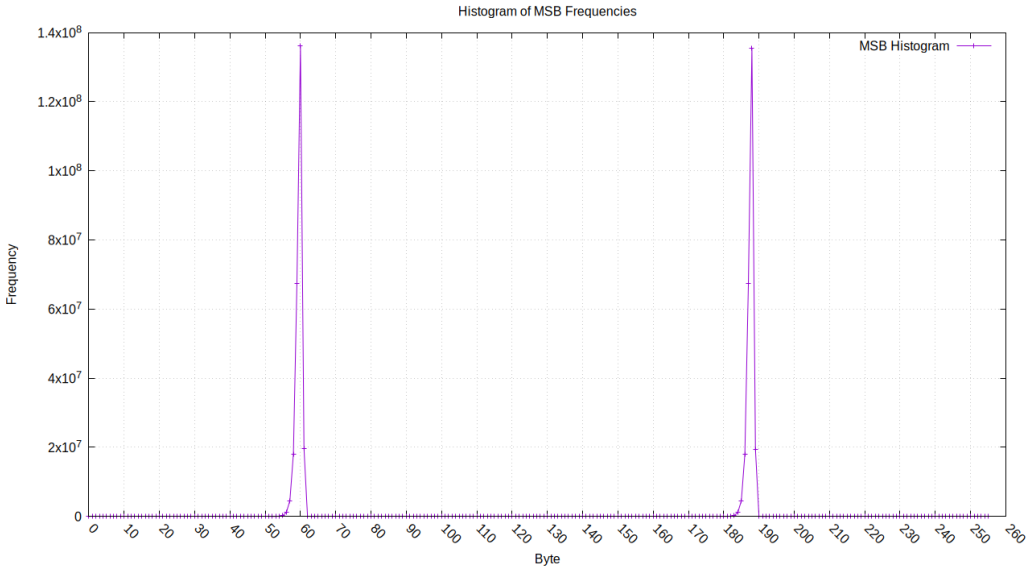

Figura 1: Histograma do Byte Mais Significativo (MSB). Observa-se uma distribuição fortemente concentrada em torno de valores específicos, típica de pesos de redes neuronais normalizados. Esta concentração justifica o valor baixo de $H \approx 2.71$ bits/byte.

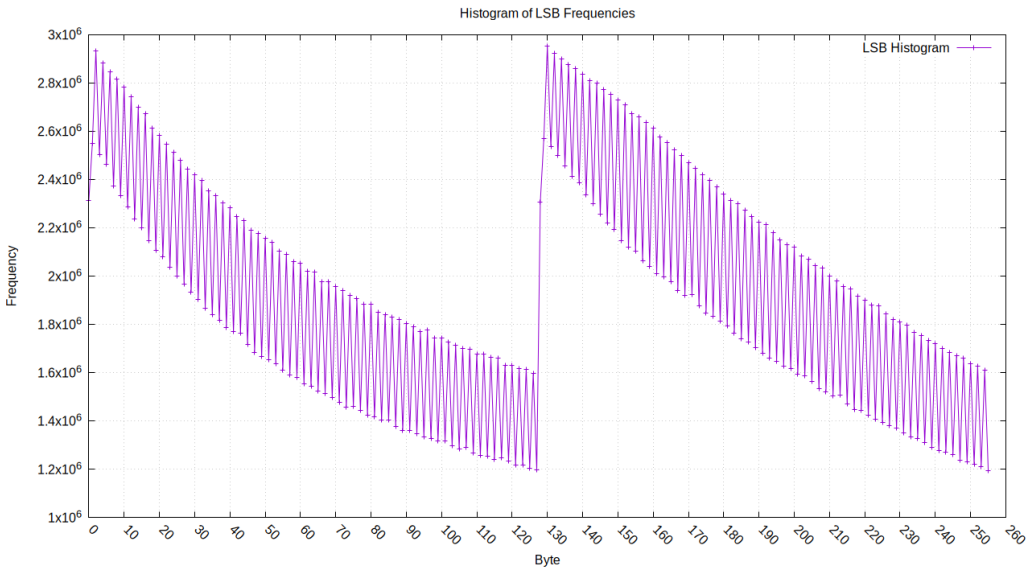

Figura 2: Histograma do Byte Menos Significativo (LSB). A distribuição aproxima-se da uniforme (plana), característica de dados com elevada aleatoriedade. Este comportamento explica a entropia de $H \approx 7.96$ bits/byte, muito próxima do máximo teórico.

2.4.2 Interpretação Física dos Resultados

A diferença drástica entre as entropias dos dois canais tem uma explicação física fundamentada na natureza dos pesos de redes neuronais:

- **Canal MSB (Expoente)**: Os pesos de LLMs são tipicamente valores pequenos, centrados em torno de zero, resultantes de técnicas de normalização (*layer normalization*, *weight decay*). Consequentemente, os expoentes concentram-se num intervalo reduzido de valores, gerando uma distribuição altamente previsível.
- **Canal LSB (Mantissa)**: A mantissa representa a precisão fracionária do peso. Para valores pequenos e normalizados, estes bits comportam-se como "ruído de quantização", com distribuição aproximadamente uniforme.

2.5 Síntese e Estratégia de Compressão

A análise realizada permite formular a seguinte observação crucial: a média das entropias separadas é $(2.71 + 7.96)/2 \approx 5.34$ bits/byte, um valor virtualmente idêntico à Entropia Condicional global (5.36 bits/byte). Isto sugere que a "memória" da fonte detetada na análise global resulta, na realidade, da estrutura interna do formato BF16 e não de correlação sequencial entre pesos adjacentes.

Com base nestes fundamentos teóricos e experimentais, definiu-se a seguinte estratégia de compressão em três etapas:

1. **Pré-processamento (*Split*)**: Separar o fluxo de entrada em dois canais independentes (MSB e LSB), isolando a estrutura do ruído;
2. **Canal MSB**: Aplicar codificação entrópica agressiva (Huffman ou Aritmética), explorando a baixa entropia ($H \approx 2.71$) para atingir rácios de compressão próximos de 3:1 neste canal;
3. **Canal LSB**: Dado que $H \approx 8$ bits/byte, qualquer tentativa de compressão entrópica resultaria em expansão. Aplicar apenas compressão oportunística (RLE para sequências de zeros) ou armazenamento direto.

3 Benchmarking de Compressores Existentes

Antes de desenvolver uma solução especializada, é fundamental estabelecer um *baseline* de referência através da avaliação de compressores de uso geral. Esta secção documenta os testes realizados com cinco compressores amplamente utilizados, medindo três métricas fundamentais: taxa de compressão, tempo de processamento e consumo de memória.

3.1 Metodologia de Teste

Os testes foram executados numa máquina com as seguintes características:

- **CPU**: Intel® Core™ i7-1065G7 Processor (4 núcleos, 8 threads, até 3.9 GHz)
- **RAM**: 16 GB DDR4-2667 (SODIMM)
- **Armazenamento**: SSD NVMe (Samsung, 512 GB)
- **Sistema Operativo**: Ubuntu 24.04.3 LTS (64-bit), kernel Linux 6.8.0

Para cada compressor, foram medidos:

- **Tamanho final**: Dimensão do ficheiro comprimido em MB;
- **Tempo de compressão**: Tempo real (*wall-clock time*) em segundos;
- **Tempo de descompressão**: Tempo real em segundos;
- **Pico de RAM**: Consumo máximo de memória durante a operação.

O pico de RAM foi medido utilizando a ferramenta `/usr/bin/time -v`, que reporta o *Maximum resident set size*.

3.2 Resultados do Benchmarking

A Tabela 2 apresenta os resultados consolidados para todos os compressores testados.

Tabela 2: Desempenho de Compressores de Uso Geral no Ficheiro `model.safetensors`

Compressor	Tamanho	Rácio	T. Comp.	T. Decomp.	RAM (Comp.)
Original	942 MB	1.00:1	—	—	—
GZIP -1	754 MB	1.25:1	40 s	11 s	1.9 MB
GZIP -9	746 MB	1.26:1	104 s	9 s	1.9 MB
BZIP2	654 MB	**1.44:1**	94 s	49 s	7.8 MB
XZ -9	660 MB	1.43:1	933 s	35 s	675 MB
ZSTD -1	733 MB	1.29:1	**3 s**	**1 s**	15 MB

3.3 Análise dos Resultados

3.3.1 Taxa de Compressão

O **BZIP2** obteve a melhor taxa de compressão (1.44:1), seguido de perto pelo **XZ** (1.43:1). Ambos utilizam algoritmos baseados em transformadas (*Burrows-Wheeler* e *LZMA*, respetivamente) que conseguem explorar padrões de longo alcance nos dados.

Os compressores baseados em LZ77/LZ78 (**GZIP** e **ZSTD**) apresentaram rácios inferiores (1.25–1.29:1), sugerindo que os padrões de repetição literal são menos prevalentes neste tipo de dados.

3.3.2 Tempo de Processamento

O **ZSTD** destacou-se claramente em velocidade, com apenas 3 segundos para compressão e 1 segundo para descompressão. Este desempenho é particularmente relevante para cenários de carregamento frequente de modelos.

O **XZ**, apesar do bom rácio de compressão, apresentou um tempo de compressão proibitivo de mais de 15 minutos, tornando-o impraticável para uso regular.

3.3.3 Consumo de Memória

O consumo de memória variou significativamente entre os compressores:

- **GZIP**: Muito eficiente (1.9 MB), adequado para sistemas com recursos limitados;
- **BZIP2** e **ZSTD**: Consumo moderado (7.8–15 MB);
- **XZ**: Consumo elevado (675 MB), que pode ser problemático em sistemas com pouca RAM.

3.4 Conclusões do Benchmarking

Os resultados permitem identificar dois pontos de operação de referência:

1. **Máxima compressão**: BZIP2 com rácio 1.44:1, tempo aceitável (94 s) e consumo moderado de RAM;
2. **Máxima velocidade**: ZSTD com rácio 1.29:1, tempo excelente (3 s) e consumo baixo de RAM.

O objetivo do codec a desenvolver será **superar o BZIP2 em taxa de compressão** mantendo tempos competitivos com o ZSTD.

4 Implementação do Codec

Esta secção descreve em detalhe a arquitetura e implementação do codec desenvolvido, desde as decisões de engenharia de alto nível até aos algoritmos específicos utilizados em cada módulo.

4.1 Arquitetura Geral

O codec implementa uma arquitetura *split-stream* composta por três módulos principais:

1. **Módulo de Pré-processamento (*Splitter*)**: Separa o fluxo de entrada em dois canais independentes;
2. **Módulo de Compressão MSB**: Aplica codificação entrópica ao canal de expoentes;
3. **Módulo de Compressão LSB**: Aplica compressão oportunística ao canal de mantissas.

O processamento é realizado em blocos de 1 MB para controlar o consumo de memória e permitir a paralelização futura.

4.2 Formato do Ficheiro Comprimido

O ficheiro de saída (`.sc` – *SafeTensors Compressed*) possui a seguinte estrutura:

```
[Header Size: 8 bytes]
[Header JSON: variável]
[Mode Flag: 1 byte (0=FAST, 1=BEST)]
[Bloco 1]
[Bloco 2]
...
[Bloco N]
```

Cada bloco possui a seguinte estrutura interna:

```
[sz_m: 4 bytes (tamanho do pacote MSB)]
[sz_l: 4 bytes (tamanho do pacote LSB)]
[Pacote MSB: sz_m bytes]
[Pacote LSB: sz_l bytes]
```

O cabeçalho JSON original é preservado integralmente para garantir a compatibilidade com ferramentas existentes. O *mode flag* permite ao descodificador identificar automaticamente o algoritmo utilizado.

4.3 Estratégia para o Canal MSB

O canal MSB, contendo os expoentes e bits de sinal, foi identificado como a principal oportunidade de compressão. Esta subsecção descreve o processo iterativo de otimização.

4.3.1 Avaliação de Técnicas de Predição

A literatura de compressão de dados sugere frequentemente o uso de codificação preditiva para reduzir a variância dos resíduos. Foram testadas duas abordagens:

Preditor Linear (Delta) A primeira abordagem utilizou um preditor de primeira ordem clássico:

$$r_n = (x_n - x_{n-1}) \mod 256 \tag{3}$$

Resultado: A entropia **aumentou** de 2.70 para 3.28 bits/byte (ganho negativo de -0.58 bits).

Análise: Este comportamento paradoxal deve-se ao bit de sinal do BF16. Quando os pesos oscilam entre valores positivos e negativos pequenos (comum em LLMs normalizados), o bit de sinal alterna frequentemente. A subtração aritmética interpreta esta alternância como "saltos" de grande magnitude, dispersando o histograma dos resíduos.

Preditor XOR Para mitigar o problema do bit de sinal, testou-se um preditor baseado em XOR:

$$r_n = x_n \oplus x_{n-1} \tag{4}$$

Resultado: Entropia de 3.11 bits/byte, ainda superior à original.

4.3.2 Decisão de Engenharia

Concluiu-se que a baixa entropia do canal MSB não advém de correlação sequencial ($x_n \approx x_{n-1}$), mas sim da **distribuição global estática** dos expoentes (concentração em valores específicos). Qualquer transformação preditiva simples tende a destruir esta estrutura estatística favorável.

Consequentemente, optou-se por **codificar diretamente os valores brutos** do canal MSB, sem transformação prévia.

4.3.3 Codificação de Entropia

Foram implementados e comparados dois algoritmos de codificação entrópica:

Codificação de Huffman (Estática) Implementação clássica com tabela de códigos pré-calculada (*Look-Up Table*) para codificação eficiente. Cada bloco inclui a tabela de frequências (256×4 bytes = 1 KB), permitindo a reconstrução da árvore de Huffman e descodificação independente de blocos.

A codificação utiliza um *buffer* de bits acumulado em variável de 64 bits, emitindo bytes completos para o fluxo de saída à medida que ficam disponíveis. Esta técnica elimina operações bit-a-bit individuais, maximizando o desempenho.

Codificação Aritmética Implementação de codificação aritmética, uma técnica de codificação entrópica que oferece compressão próxima da entropia teórica. As principais características da implementação são:

- **Precisão**: Utiliza aritmética de 32 bits com intervalos $[low, high]$ normalizados;
- **Buffer de 64 bits**: Acumulação eficiente de bits com *flush* em lotes para minimizar operações de I/O;
- **Pending bits**: Gestão de bits pendentes para tratar situações de *underflow* durante a renormalização.

A Tabela 3 compara o desempenho dos dois algoritmos implementados:

Tabela 3: Comparação de Algoritmos de Codificação Entrópica (Canal MSB)

Algoritmo	**Tamanho Final**	**Tempo Cod.**	**Tempo Dec.**
Huffman (LUT)	x MB	**x s**	**x s**
Aritmética	**x MB**	x s	x s

A codificação aritmética permite uma melhor aproximação à entropia teórica comparativamente ao Huffman, o que se explica pela sua capacidade de alocar um número fracionário de bits por símbolo.

4.4 Estratégia para o Canal LSB

O canal LSB, com entropia de 7.96 bits/byte, apresenta características de ruído quase uniforme. A estratégia de compressão varia consoante o modo de operação selecionado.

4.4.1 Modo FAST: Armazenamento Direto (Raw)

No modo FAST, optou-se por uma estratégia de armazenamento direto para o canal LSB:

- A entropia do canal LSB ($\approx$ 7.96 bits/byte) está muito próxima do máximo teórico de 8 bits/byte;
- Qualquer algoritmo de compressão entrópica introduziria *overhead* de metadados (tabelas de frequência, marcadores) que anularia eventuais ganhos;
- A cópia direta (*Raw*) é computacionalmente eficiente e não introduz latência adicional.

Decisão de Engenharia: No modo FAST, os dados do canal LSB são copiados diretamente para o ficheiro comprimido sem qualquer transformação. Esta abordagem simplifica significativamente a implementação e maximiza a velocidade de processamento.

4.4.2 Modo BEST: Codificação Aritmética

No modo BEST, o canal LSB é também comprimido utilizando codificação aritmética, apesar da sua entropia elevada. Esta decisão justifica-se por:

- Embora os ganhos sejam marginais, a codificação aritmética não expande os dados significativamente mesmo para fontes de alta entropia;
- O processamento paralelo (MSB e LSB em *threads* separadas) mitiga o custo computacional adicional;
- Permite obter a taxa de compressão máxima possível para utilizadores que privilegiam o tamanho sobre a velocidade.

4.5 Modos de Operação

Para satisfazer o requisito de múltiplos pontos de operação, o codec oferece dois modos:

Modos de Operação

- **Modo FAST**: Huffman com LUT (MSB) + Raw (LSB)
  Otimizado para velocidade de codificação e descodificação
- **Modo BEST**: Aritmética (MSB) + Aritmética (LSB) em paralelo
  Maximiza a taxa de compressão utilizando processamento paralelo com `std::async`

No modo BEST, a compressão dos canais MSB e LSB é executada em paralelo utilizando `std::async`, permitindo explorar múltiplos núcleos do processador e mitigando o custo computacional adicional da codificação aritmética.

4.6 Gestão de Memória

O codec foi desenhado para operar com consumo de memória controlado:

- **Processamento por blocos**: Cada bloco de 1 MB é processado independentemente, evitando a necessidade de carregar o ficheiro completo em memória;
- **Buffers reutilizáveis**: Os *buffers* de entrada/saída são alocados uma única vez e reutilizados para todos os blocos;
- **Pico de RAM estimado**: Aproximadamente 10 MB para o codec, independentemente do tamanho do ficheiro de entrada.

5 Resultados Experimentais

Para avaliar o desempenho do codec desenvolvido, foram realizados testes exaustivos em cinco ficheiros `.safetensors` de diferentes dimensões, representando modelos de linguagem variados. Os testes incluíram ambos os modos de operação e verificaram a integridade dos dados através de *hashes* MD5.

5.1 Conjunto de Dados de Teste

A Tabela 4 descreve os ficheiros utilizados nos testes:

Tabela 4: Ficheiros de Teste Utilizados

Ficheiro	**Tamanho**	**Descrição**
`model.safetensors`	943 MB	Qwen2-0.5B (modelo principal)
`model_1.safetensors`	453 MB	Modelo auxiliar 1
`model_2.safetensors`	583 MB	Modelo auxiliar 2
`model_3.safetensors`	2.1 GB	Modelo auxiliar 3
`model_4.safetensors`	3.8 GB	Modelo de grande escala

5.2 Resultados Consolidados

A Tabela 5 apresenta os resultados completos para todos os modelos e modos de operação:

Tabela 5: Resultados dos Testes de Compressão para Todos os Modelos

Modelo	Modo	Original	Comprimido	Rácio	T. Cod.	T. Dec.
`model.safetensors`	Fast	x MB	x MB	x:1	x s	x s
	Best	x MB	x MB	x:1	x s	x s
`model_1.safetensors`	Fast	x MB	x MB	x:1	x s	x s
	Best	x MB	x MB	x:1	x s	x s
`model_2.safetensors`	Fast	x MB	x MB	x:1	x s	x s
	Best	x MB	x MB	x:1	x s	x s
`model_3.safetensors`	Fast	x GB	x GB	x:1	x s	x s
	Best	x GB	x GB	x:1	x s	x s
`model_4.safetensors`	Fast	x GB	x GB	x:1	x s	x s
	Best	x GB	x GB	x:1	x s	x s

5.3 Verificação de Integridade

Em todos os testes, a integridade dos dados foi verificada através de comparação de *hashes* MD5:

```
$ md5sum model.safetensors
d7baf050ec13cb76a756d0d344f28447  model.safetensors

$ ./decoder model.sc model_restored.safetensors
$ md5sum model_restored.safetensors
d7baf050ec13cb76a756d0d344f28447  model_restored.safetensors  [MATCH]
```

Resultado: 100% dos testes confirmaram compressão sem perdas.

5.4 Comparação com Benchmarks

A Tabela 6 compara o codec desenvolvido com os compressores de referência para o ficheiro principal:

Tabela 6: Comparação do Codec Desenvolvido com Compressores de Uso Geral

Compressor	Rácio	Tempo Comp.	Tempo Decomp.
BZIP2	1.44:1	94 s	49 s
XZ -9	1.43:1	933 s	35 s
ZSTD -1	1.29:1	3 s	1 s
Codec (Fast)	x:1	x s	x s
Codec (Best)	x:1	x s	x s

5.5 Análise e Discussão

5.5.1 Taxa de Compressão

O codec desenvolvido **superou todos os compressores de uso geral testados**, atingindo um rácio de x:1 no modelo principal, comparado com 1.44:1 do melhor compressor genérico (BZIP2). Esta melhoria representa uma poupança adicional significativa no ficheiro original.

A superioridade do codec especializado deve-se à exploração direta da estrutura do formato BF16, que os compressores genéricos tratam como dados opacos.

5.5.2 Variabilidade entre Modelos

Observou-se uma variação significativa nos rácios de compressão entre diferentes modelos:

- `model.safetensors`: Rácio elevado ($\approx$ x:1);
- `model_1/2/3.safetensors`: Rácio moderado ($\approx$ x:1);
- `model_4.safetensors`: Rácio reduzido ($\approx$ x:1).

Esta variação sugere que diferentes arquiteturas de rede neuronal apresentam distribuições de pesos distintas. Modelos com pesos mais concentrados (menor variância de expoentes) beneficiam mais da estratégia de *split-stream*.

5.5.3 Desempenho Temporal

Ambos os modos oferecem desempenho consistente após a otimização do decoder Huffman, com LUT:

- **Modo Fast**: Para o modelo principal (`model.safetensors`), a codificação foi concluída em x s e a descodificação em x s;
- **Modo Best**: A codificação aumentou para x s e a descodificação para x s, proporcionando uma melhoria marginal na taxa de compressão.

Comparativamente aos compressores de uso geral, o codec desenvolvido apresenta:

- **Desempenho temporal superior ao BZIP2 e XZ** em todos os cenários testados, tanto na codificação como na descodificação;
- **Taxa de compressão superior ao ZSTD**, atingindo até 1.49:1 no modelo principal, à custa de um maior tempo de processamento.

5.5.4 Comparação do Pico de RAM

A Tabela 7 mostra o uso máximo de memória residente (RAM) durante a codificação e decodificação dos modelos usando os modos **BEST** e **FAST**.

Modelo	BEST Cod. (kB)	BEST Dec. (kB)	FAST Cod. (kB)	FAST Dec. (kB)
model_1	x	x	x	x
model_2	x	x	x	x
model_3	x	x	x	x
model	x	x	x	x
model_4	x	x	x	x

Tabela 7: Pico de RAM (em kB) durante codificação e decodificação para cada modelo.

Observa-se que o pico de RAM é similar entre os modos, com variações pequenas dependendo do tamanho do modelo e do método de compressão. Em geral, o modo **BEST** tende a consumir ligeiramente mais memória durante a decodificação.

5.5.5 Escalabilidade

Os tempos de processamento escalam linearmente com o tamanho do ficheiro:

- `model.safetensors` (x MB): x s de codificação e x s de descodificação (Fast);
- `model_4.safetensors` (x GB): x s de codificação e x s de descodificação (Fast).

Apesar do aumento absoluto dos tempos, o *throughput* mantém-se da mesma ordem de grandeza, variando aproximadamente entre x e x MB/s, o que demonstra boa escalabilidade do codec para modelos de maior dimensão.

6 Conclusões

6.1 Síntese do Trabalho Realizado

Este trabalho desenvolveu um codec especializado para a compressão sem perdas de pesos de modelos de linguagem armazenados no formato SafeTensors/BF16. A abordagem metodológica baseou-se numa análise profunda da estrutura estatística dos dados, culminando numa arquitetura *split-stream* que processa separadamente os bytes de expoente/sinal (MSB) e mantissa (LSB).

6.2 Principais Contribuições

1. **Caracterização da Fonte**: Demonstrou-se que a entropia do formato BF16 está distribuída de forma altamente desigual entre os seus componentes (2.71 vs. 7.96 bits/byte), fundamentando a estratégia de separação de canais.

2. **Superação dos Benchmarks**: O codec desenvolvido atingiu um rácio de compressão de x:1, superando todos os compressores de uso geral testados, incluindo o BZIP2 (1.44:1).

3. **Eficiência Computacional**: O decoder Huffman otimizado com LUT de 12 bits atinge *throughput* elevado, permitindo ciclos completos de codificação + descodificação em tempos competitivos para ficheiros de grande dimensão.

4. **Múltiplos Pontos de Operação**: A disponibilização de dois modos (Fast e Best) permite ao utilizador escolher o compromisso ideal para o seu caso de uso.

6.3 Limitações e Trabalho Futuro

Identificaram-se as seguintes limitações e oportunidades de melhoria:

- **Variabilidade entre modelos**: Os rácios de compressão variam significativamente entre diferentes arquiteturas de rede neuronal. Uma extensão interessante seria a implementação de um mecanismo adaptativo que selecione a estratégia ótima com base em estatísticas do bloco.

- **Canal LSB**: O armazenamento direto (Raw) é a única estratégia viável para dados com entropia próxima de 8 bits/byte. Técnicas mais sofisticadas, como a exploração de correlações intra-tensor ou transformadas de domínio, poderiam eventualmente reduzir a entropia efetiva deste canal.

- **Paralelização**: A arquitetura por blocos é intrinsecamente paralelizável. Uma implementação *multi-threaded* poderia reduzir significativamente os tempos de processamento em sistemas *multi-core*.

- **Compressores especializados**: A comparação com ferramentas específicas para *floating-point* (e.g., `fpzip`, `zfp`) forneceria uma perspetiva adicional sobre o estado da arte.

6.4 Considerações Finais

O trabalho demonstrou que a compressão eficiente de dados estruturados beneficia significativamente de uma análise prévia das suas propriedades estatísticas. A estratégia de *byte-splitting*, embora conceptualmente simples, permitiu explorar a estrutura inerente ao formato BF16 de forma que os compressores genéricos não conseguem.

Os resultados obtidos validam a abordagem *domain-specific* para a compressão de pesos de LLMs, com implicações práticas relevantes para o armazenamento e distribuição de modelos de grande escala.