

TELECOM
Paris



December, 2025

Algorithmic Information & Artificial Intelligence

Micro-studies

teaching.dessalles.fr/FCI

This document contains students' contributions written during the course “Algorithmic Information & Artificial Intelligence” (CSC_5AI25_TP) taught in September-November 2025.

I am very thankful to students for having been active participants, and also for the great quality of their work.

Jean-Louis Dessalles

Contents

			page
Mohamed	Ahmed Bouha	How much can we compress a chess position?	61
Pablo	Bertaud--Velten	Algorithmic complexity and perceived musical interest: A compression-based approach	5
Christelle	Clervilsson	Alzheimer's disease	13
Mostafa	Djanbaz	French author classifier	49
Tristan	Donzé	Algorithmic complexity and perceived musical interest: A compression-based approach	5
Zeinab	Ghamlouch	Predicting the 'Interestingness' of News Headlines Using Simplicity Theory	93
Astrid	Giuliani	Mesurer l'esthétique perçue d'une musique au travers de sa compressibilité.	85
Théodore	Halley	Time-series correlation	117
Mikhail	Leontev	Complexity analysis of Marion Bros levels	23
Ziyi	Liu	Detecting singer-songwriters without knowing styles	41
Alessa	Mayer	Predicting the 'Interestingness' of News Headlines Using Simplicity Theory	93
Vijay Venkatesh	Murugan	Implicit Neural Representations as Minimum Description Length Models for 2D Images	137
Ivana	Nassar	General Model Selection Principle	51
Marcin	Porwisz	MDL as an Optimization Criterion in Metaheuristics	69
Stepan	Svirin	Compression-Based Metrics for Code Quality	31
Thanh Nam	Tran	Measuring Pun Unexpectedness with Decoder Models and Phonetic/Semantic Compression	127
yina	xia	MDL Finds Customer Archetypes: Prototype Clustering & Outliers on UCI Wholesale Customers	77
Yixing	Yang	Detecting singer-songwriters without knowing styles	41
Ada	Yetis	Separation Between Categories: How much information do distinguishing features carry?	107
Ines	Zait	Alzheimer's disease	13
Yufei	Zhou	Compression-Based Metrics for Code Quality	31



November, 2025

CSC-5AI25-TP

Algorithmic Information & Artificial Intelligence

Micro-study

teaching.dessalles.fr/FCI

Name: Pablo Bertaud--Velten & Tristan Donzé

Algorithmic complexity and perceived musical interest: A compression-based approach

Abstract

This project investigates the relationship between the structural complexity of music and its popularity. By using LZMA compression on MIDI files as a proxy for Kolmogorov complexity, we analyzed thousands of tracks across various genres. We observe a weak negative correlation between complexity and popularity, suggesting that mass audiences tend to favor higher predictability and repetition in musical structures.

Problem

Our research stems from a fundamental question: do listeners prefer the comfort of familiar patterns, or do they seek the novelty of the unexpected?

Specifically, we address whether high algorithmic complexity, characterized by high information content and low compressibility, inherently limits a song's commercial appeal. We aim to determine if popular music is successful precisely because it is algorithmically simple, or if there is a "sweet spot" of complexity that maximizes listener engagement.

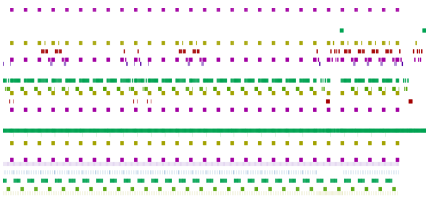


Figure 1: Example of the MIDI visual representation of a highly compressible song (compression ratio of 0.02). Note the many repetitions and similarities across the range of the song.

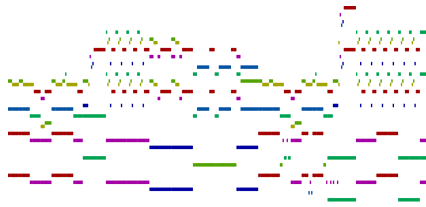


Figure 2: Example of the MIDI visual representation of a song with average compression ratio (0.22).

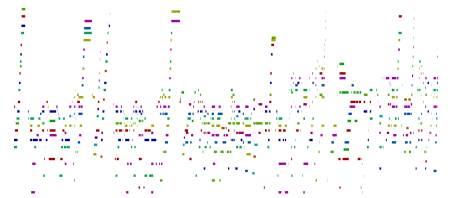


Figure 3: Example of the MIDI visual representation of a song that is not very compressible (0.746). Note the lack of clear structure and great variability, even in the parts that look somewhat similar.

Method

We utilized the Lakh MIDI dataset, comprising 10,278 songs from the years 1950 to the 2000s encoded as MIDI files, to capture pure musical structure (melody, harmony, and rhythm) without the confounding variables of timbre or production quality. To enrich this structural data, we queried the Spotify API to retrieve popularity scores (normalized to a logarithmic range of $[0, 100]$) and genre tags for each artist and song. To facilitate a more robust statistical analysis of genre-specific patterns, we also reduced the dimensionality of the genre classification. We achieved this by mapping the 381 distinct sub-genres found in the dataset to 20 broader global categories.

To quantify the structural complexity of these musical sequences, we employed the LZMA compression algorithm as an approximation of their Kolmogorov complexity $K(x)$. We defined the **compression ratio** of a song as the ratio between its size after compression and its original size, expressed as:

$$\text{Compression Ratio} = \frac{\text{Compressed size}}{\text{Original size}} \in [0, 1] \quad (1)$$

In our experiments, this metric served as a proxy for structural density. A high compression ratio (specifically, a ratio exceeding 0.4) indicates a song that is not easily compressible, exhibiting sufficient pattern variability to prevent significant size reduction. Conversely, a song with a low compression ratio (typically below 0.1) implies high compressibility, characterizing music with repetitive structures and high redundancy.

We further analyzed the nature of information contained in musical sequences by contrasting 0-order **Shannon Entropy** with K-complexity (approximated by LZMA). For this purpose, we parsed MIDI files into chronological sequences of tokens representing pitch and duration. We computed these metrics on both the original sequences and randomly shuffled versions of the same data. The objective was to demonstrate a fundamental distinction: while 0-order entropy measures only the statistical frequency of symbols, and remains invariant to order, compression-based complexity captures the temporal structure and inherent patterns of music.

In addition to individual song analysis, we explored artist classification based on the diversity of their repertoire using the **Normalized Compression Distance (NCD)**. For any two musical

sequences x and y , where $C(x)$ represents the compressed size of x and $C(x \parallel y)$ as well as $C(y \parallel x)$ represents the compressed size of their concatenations, the pairwise distance is calculated as follows:

$$\text{NCD}(x, y) = \frac{\max(C(x \parallel y), C(y \parallel x)) - \min(C(x), C(y))}{\max(C(x), C(y))} \quad (2)$$

We attempted to classify artists according to the average NCD between every pair of their songs. However, the results were not discriminative enough for conclusive reporting, with all artists exhibiting an average pairwise NCD in the narrow range of $[0.88, 0.99]$. While comparing an artist's catalog against foreign music might yield better distinctiveness, such an approach is computationally prohibitive, as the number of required pairs evolves quadratically with the total number of songs.

Finally, we introduced a metric of *Unexpectedness* to measure the information cost of a musical segment given its preceding context. This metric operates on a sliding window of size w . We define the Context C as the concatenation of tokens $[i : i + w]$ and the Target T as the concatenation of tokens $[i + w : i + 2w]$. Letting $C(\cdot)$ denote the LZMA compressed size and $|T|$ the length of the target string in characters, the unexpectedness score is defined as:

$$\text{score}(C, T) = \frac{C(C \parallel T) - C(C)}{|T|} \quad (3)$$

where $(C \parallel T)$ denotes the concatenation of the context and the target. This formula acts as a proxy for **Conditional Kolmogorov Complexity**, quantifying the "surprise" of the target window relative to the context. A score near 0 indicates that the target is structurally identical to the context (pure repetition), whereas structured music typically falls within the $[0.05, 0.15]$ range. Conversely, values exceeding 0.30 imply a high density of new information, characteristic of chaotic or uncorrelated musical textures.

Results

To ensure that our compression-based approach truly measures structural complexity rather than simple statistical frequency, we first validated our metric by comparing original tracks against randomly shuffled versions of the same data.

As illustrated in Figure 4, Shannon entropy remains invariant under shuffling because the frequency of notes does not change. However, the compression ratio increases significantly for the shuffled versions. This confirms that LZMA successfully captures the temporal structure, repetition, and order inherent in music. The original songs are far more compressible than their randomized counterparts, validating compression ratio as a proxy for musical structure.

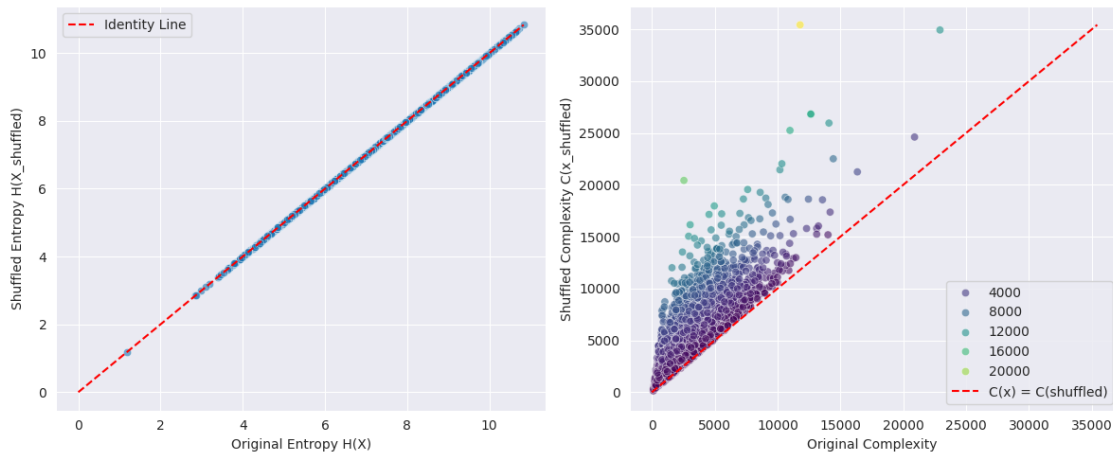


Figure 4: Validation of the metric. **Left:** Entropy is shuffle-invariant (points lie on the identity line). **Right:** Algorithmic complexity increases significantly after shuffling. Points close to the diagonal represent songs that were already highly complex or unstructured.

We then proceeded to analyze the distribution of popularity and complexity across genres. Figure 5 reveals a striking inverse relationship.

The popularity ranking (Left) shows that genres like Hip-hop, Metal, and Pop dominate the dataset in terms of commercial success. Conversely, the compression analysis (Right) shows that these same genres are the most compressible (lowest ratios). They are characterized by high repetitiveness and structural predictability. On the other end of the spectrum, Classical and Jazz music appear as the least popular but the most algorithmically complex, indicating a lower amount of stable, repetitive structure.

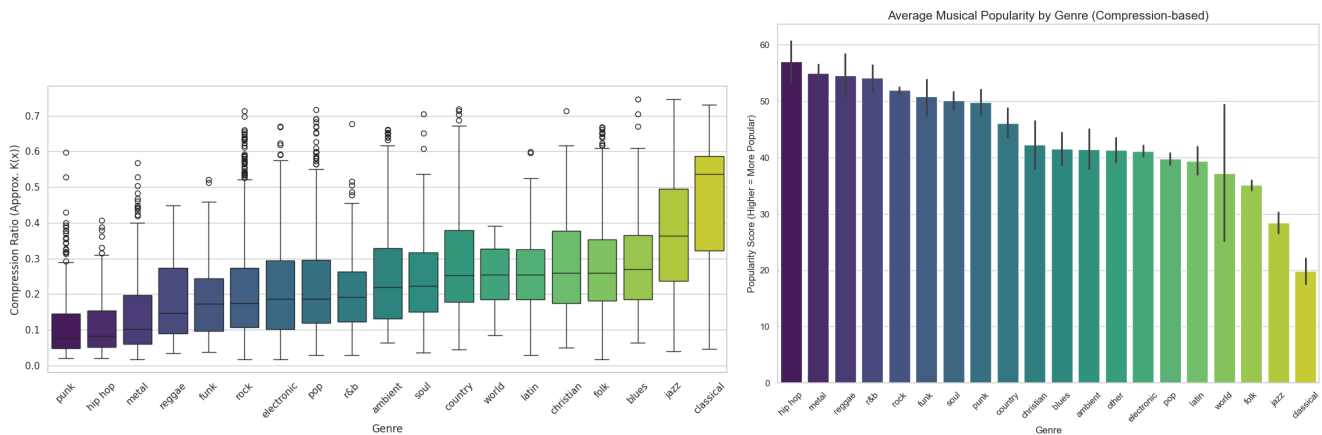


Figure 5: **Left:** Variability of compression ratios across genres (Lower is simpler). **Right:** Average popularity across genres. We observe an almost exact inversion: the most complex genres (Jazz, Classical) are the least popular.

We expanded this analysis to the entire dataset to observe global trends. As shown in Figure 6, we observe a weak but statistically significant negative correlation between compressibility and popularity (Pearson: -0.197).

The linear model indicates that as complexity increases (moving right on the x-axis), popularity generally decreases. The quadratic model refines this observation, suggesting a plateau of popularity for low-to-medium complexity songs, followed by a sharp drop-off as complexity becomes

too high. This suggests some type of "sweet spot" where listeners prefer a balance of structure but reject high levels of disorder (high incompressibility).

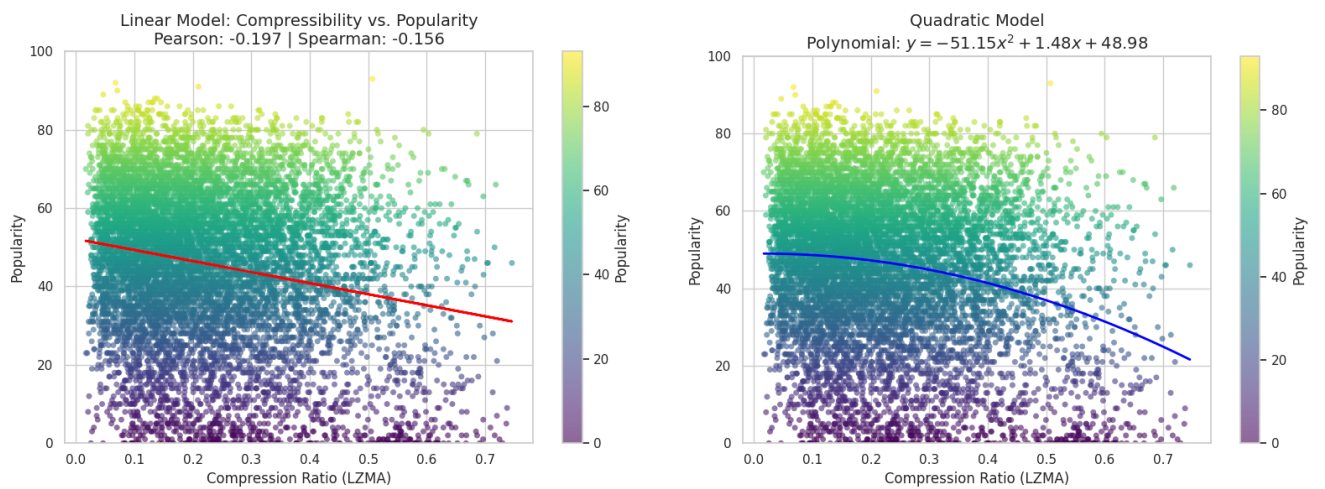


Figure 6: **Left:** Linear regression showing a negative correlation between complexity and popularity. **Right:** Quadratic fit suggesting popularity drops sharply after a certain complexity threshold.

To refine our understanding of "surprise" in music, we calculated the *Unexpectedness* score. This metric mirrors the results found with raw compression. Figure 7 confirms that listeners generally disfavor highly surprising content, with a trend very similar to the compression ratio plots (5).

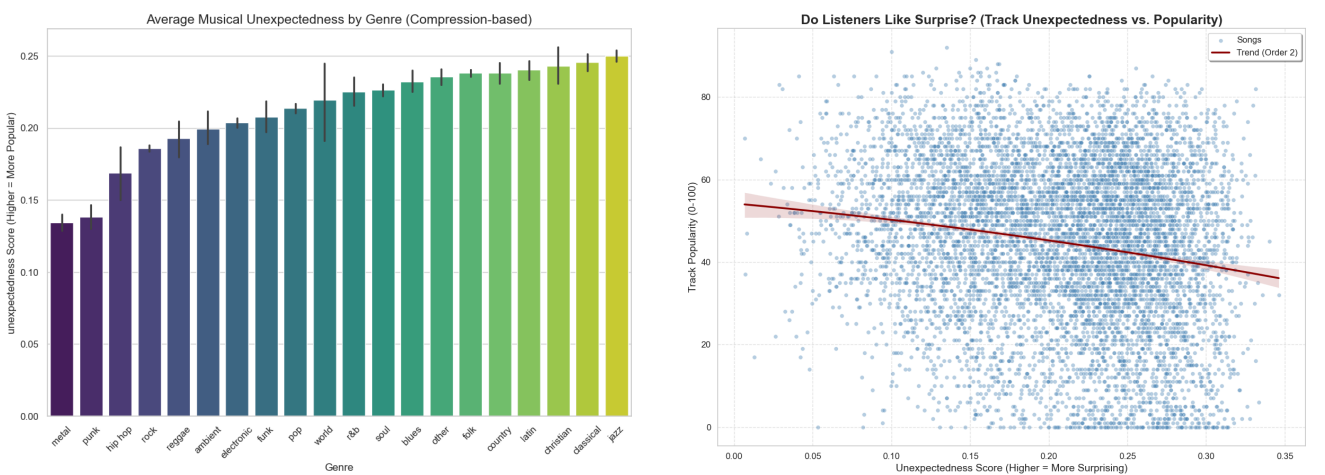


Figure 7: **Left:** Average Unexpectedness by genre. **Right:** Scatter plot of Unexpectedness vs. Popularity. The trends align with previous compression findings: "surprising" music is generally less popular.

Synthesizing these dimensions, Figure 8 maps the landscape of musical genres. Rather than forming isolated clusters, the genres are distributed along a continuous diagonal spectrum. Mass-appeal genres (Hip-hop, Metal, Punk) occupy the region of high popularity and low unexpectedness (top-left), while niche genres (Jazz, Classical) sit at the opposite end of the spectrum (bottom-right). Bridging these extremes, genres like Rock, Pop, and Soul occupy a central transitional zone. Notably, the "Unpopular and Expected" quadrant (bottom-left) remains empty, suggesting that if a song is structurally simple and predictable, it is statistically likely to achieve some degree of commercial success.

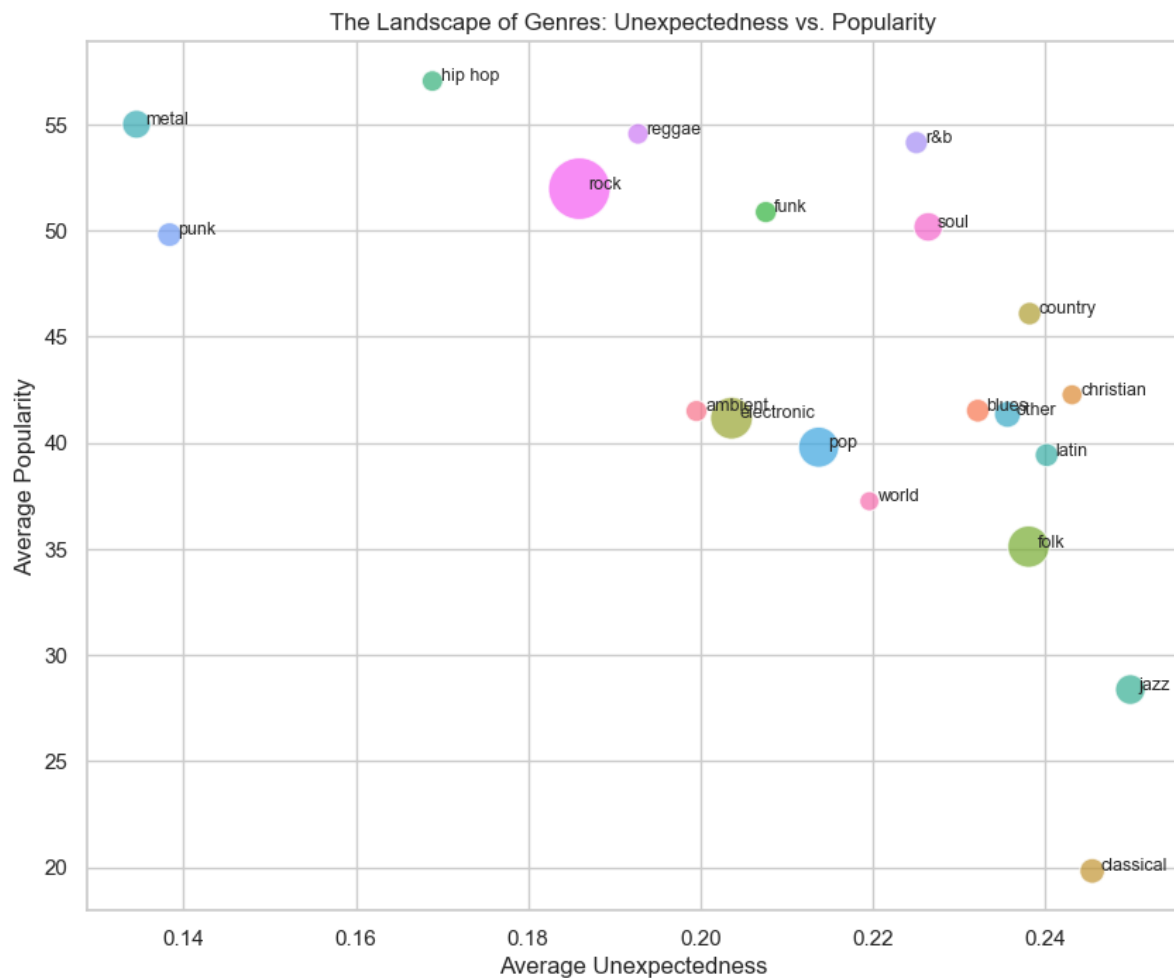


Figure 8: The Landscape of Genres: Popularity vs. Unexpectedness. Bubble size represents the number of songs in the dataset. A clear diagonal trend separates mainstream, predictable music from niche, complex compositions.

Finally, we attempted to cluster songs based purely on algorithmic similarity using the Normalized Compression Distance (NCD). However, as shown in Figure 9, this approach yielded limited results. The distribution of distances is heavily skewed toward 1.0 (maximum distance), meaning the algorithm found almost all songs to be maximally different from one another. Consequently, the Multi-Dimensional Scaling (MDS) projection resulted in a uniform sphere with no distinct genre clusters. This suggests that while compression is a good scalar metric for complexity, it is not sufficiently nuanced to capture stylistic similarity between tracks in this context.

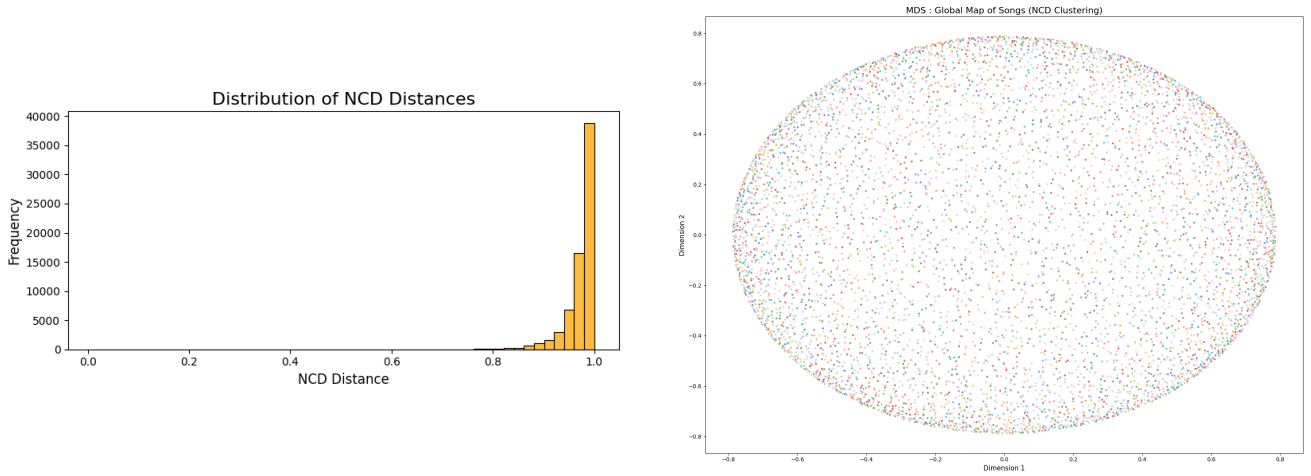


Figure 9: **Left:** Distribution of NCD distances, showing saturation near 1.0. **Right:** MDS projection showing a lack of distinct clusters, indicating that NCD failed to capture genre-specific similarities.

Discussion

The results support the hypothesis that popular music relies on repetition and low algorithmic complexity. Notably, the low unexpectedness scores in genres like Hip-hop align with their loop-based structures. However, the exclusive reliance on MIDI files introduces a significant limitation as they lack vocals. Hip-hop appears algorithmically simple because LZMA compresses the repetitive backing track, but this ignores the lyrical complexity which is the genre’s primary information source. Similarly, Electronic and Pop music rely on intricate sound design and timbre, features entirely absent in MIDI data.

Consequently, compressibility provides limited insight into the music’s full structure and serves better as an additional feature rather than a standalone metric. Future work should apply these compression-based metrics to audio spectrograms to capture the full information content, including textural and timbral nuances.

Bibliography

- Raffel, C. (2016). *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching*. PhD Thesis (Lakh MIDI Dataset).
- Cilibrasi, R., & Vitányi, P. M. (2005). Clustering by compression. *IEEE Transactions on Information Theory*.
- Spotify for Developers. (2025). Web API Reference. Retrieved from <https://developer.spotify.com/documentation/web-api>



November, 2025

CSC 5AI25 TP

Algorithmic Information & Artificial Intelligence

Micro-study

teaching.dessalles.fr/FCI

Name: Zait Nawel, Clervilsson Christelle

Alzheimer's disease

Abstract

Alzheimer's disease accounts for 60–70% of dementia cases worldwide. While AI models can detect Alzheimer's from MRI scans, they lack interpretability. We apply Algorithmic Information Theory (AIT) to test a novel hypothesis: that Alzheimer's-affected brains exhibit lower complexity than healthy ones. Using multi-stage MRI data, we provide an interpretable framework for disease detection based on information-theoretic measures.

Problem

Alzheimer's disease (AD) is the leading cause of dementia worldwide, affecting millions of people. Early detection is one of the most critical challenges in managing the disease. Structural brain changes, such as hippocampal atrophy and ventricular enlargement, begin years or even decades before clinical symptoms become severe. MRI is an essential diagnostic tool because it captures these morphological changes. However, interpreting subtle structural differences remains extremely difficult, especially in the early stages, which is precisely when intervention is most effective.

Deep Learning has become the dominant computational approach for diagnosing Alzheimer's disease from MRI data. Multiple models have achieved impressive accuracy, sometimes exceeding 90%. Despite this, these models suffer from important limitations. First, deep learning systems operate largely as black boxes, which is problematic in clinical settings where decisions must be transparent and justifiable. Second, these models are highly dependent on large, well-annotated training datasets, which introduces issues of bias,

reproducibility, and generalization. These factors make deep learning models less suitable for direct medical deployment despite their high performance.

Algorithmic Information Theory (AIT) provides a fundamentally different perspective. AIT defines the Kolmogorov complexity of an object as the length of the shortest program that can generate it, offering a universal and objective measure of information content. We introduce the hypothesis that MRI images of patients with Alzheimer's disease are algorithmically simpler than those of healthy individuals. Neurodegeneration causes neuronal loss, tissue homogenization, and structural simplification, changes that could make diseased images more compressible. This hypothesis is biologically grounded, as Alzheimer's progression reduces the amount of structural information in the brain. While Kolmogorov complexity is theoretically uncomputable, practical approximations can be obtained using lossless compression algorithms such as gzip, bz2, and lzma.

This project aims to address the following key questions:

- Can AIT-based complexity measures reliably distinguish between AD patients and healthy controls?
- Which brain regions show the strongest complexity changes during disease progression?
- How do complexity-based models compare to traditional machine learning approaches?

Method

Dataset

We chose a dataset from Kaggle, which has previously enabled Deep Learning models to reach up to 99% accuracy. The dataset consists of 11,519 MRI scans distributed across four classes: No Impairment (3,200 images, 27.8%), Very Mild Impairment (3,008 images, 26.1%), Mild Impairment (2,739 images, 23.8%), and Moderate Impairment (2,572 images, 22.3%). Each class contains both real and synthetically augmented scans. While the distribution is relatively balanced overall, the test set exhibits some class imbalance. This reflects real-world clinical distributions where advanced AD cases are less frequently scanned, but may introduce bias in evaluation metrics, particularly affecting sensitivity for moderate impairment detection.

Complexity measure

We used multiple methods to quantify the complexity of MRI images:

Kolmogorov Complexity (KC)

Formally, KC is the length of the shortest program capable of reproducing an object. Because it is uncomputable, we approximate it using lossless compression algorithms such as gzip, bz2, and lzma. The compressed size provides a practical proxy for the information content of an image. Lower compressed size indicates lower algorithmic complexity.

Conditional Complexity (Normalized Compression Distance, NCD)

NCD measures the informational similarity between two objects. To compute it, images are converted into bytes and compressed using gzip. The NCD formula combines the compressed lengths of the individual images and their concatenation to produce a normalized distance in the range $[0, 1]$. Low NCD between two images suggests they share similar algorithmic structure, making them informationally similar regardless of pixel-level differences.

Four Class Classification

Pure AIT Classification

We first implemented two classification methods based exclusively on algorithmic information theory, without traditional machine learning optimization:

1.1 - Minimum Description Length (MDL) Classifier

The MDL principle selects the model that minimizes the combined cost of encoding both the model itself and the data given that model. Our implementation proceeds in three steps: first, we compute a prototype for each class by averaging the training images. Then, for each test image, we calculate its NCD to each class prototype. Finally, we assign the test image to the class with the minimum NCD. This approach provides a fully interpretable classifier rooted in algorithmic information theory, as classification decisions are based purely on informational similarity without learned parameters.

1.2 - Binary Neural Network-inspired classifier (BNN)

This classifier is similar to MDL but does not compute class prototypes. Instead, each test image is compared to all training images using NCD, and the label of the closest image (minimum NCD) is assigned. While conceptually straightforward, the computational cost is extremely high ($O(n^2)$), making it impractical for large datasets. In practice, the BNN was not fully executed due to excessive runtime.

AIT-based Features

To evaluate how Algorithmic Information Theory (AIT) could improve classical machine learning models, we first needed to extract meaningful features from MRI images. The goal was to convert each MRI into a structured feature vector that encodes algorithmic complexity, local variability, and anatomical information.

Feature Extraction Pipeline

We first developed a rich feature extraction function composed of several elements:

Multi-scale Compression: Images were resized to $0.25\times$, $0.5\times$, and $1\times$ of their original size. For each scale, we calculated the compression-based complexity (KC) using lossless compressors. This captures how image complexity changes at different spatial resolutions, as coarser scales may reveal global patterns while finer scales capture local details.

Approximation of Anatomical Regions: We focused on biologically relevant regions known to be affected by Alzheimer's, such as the hippocampus and cortex. These regions were approximated by cropping fixed subregions of the MRI. Compression ratios were calculated

for each region, capturing region-specific complexity loss that may correlate with known patterns of neurodegeneration.

Local Variability: The image was divided into small patches (32×32 pixels). For each patch, compression was computed. We then calculated the mean and standard deviation of patch complexities, reflecting heterogeneity across the brain. High variability suggests diverse tissue types, while low variability may indicate homogenization due to atrophy.

Gradient Complexity: We applied a Sobel filter to capture edges and structural variations. The compression ratio of the edge map quantifies structural irregularity, as sharp boundaries between tissue types produce complex edge patterns.

Shannon Entropy: Calculated on the full image to capture the overall randomness of pixel intensity distribution. While Shannon entropy measures statistical randomness, it complements Kolmogorov complexity by providing a probability-theoretic perspective.

Multi-compressor Ensemble: We used gzip, bz2, and lzma to calculate compression ratios of the full image. Different compressors capture different statistical regularities, enhancing robustness to algorithm-specific biases.

We then used ML classifiers like SVMs, Random Forests, Gradient Boosting, and Logistic Regression to test the results obtained with these features. Initial results were not satisfactory, prompting further refinement.

Improved Feature Extraction

To improve classification performance, we developed a richer feature extraction pipeline that generates approximately 285 features per image. Compared to the previous version, which only considered global compression, a few anatomical regions, and basic local variability, the new extraction captures multi-scale, multi-region, and multi-metric information. Specifically, for each image scale ($0.25\times$, $0.5\times$, $1\times$), we compute compression, Shannon entropy, and edge complexity. We extract more precise anatomical regions (hippocampus, frontal cortex, temporal cortex, parietal cortex, and ventricles) rather than just hippocampus and cortex. Instead of simple local patch statistics, we divide the image into multiple grid sizes (4×4 and 8×8) and compute compression, entropy, and edge complexity for each patch. Overall, this richer representation captures both global and localized structural variations, providing more discriminative features for machine learning models and increasing the potential to detect subtle changes associated with Alzheimer's disease.

Features Classification

Since the results with the pure AIT approach were not fully satisfactory, we implemented a feature-based classification pipeline. The underlying idea remains the same as pure AIT: we aim to compare new images to reference patterns. However, instead of computing conditional complexity directly, we first extract rich, informative features from each image and then perform classification using standard distance measures in feature space.

Binary Classification

We reformulated the original four-class problem into a binary classification task (Healthy vs. Impaired) for several converging reasons. Preliminary analysis revealed that AIT-based complexity measures produced clear separation between healthy and impaired brains, but consecutive impairment stages (Very Mild, Mild, Moderate) showed substantially smaller complexity differences with significant distributional overlap. This reflects the gradual nature of neurodegeneration, where structural simplification occurs most dramatically in the transition from healthy to diseased states, while subsequent progression involves more subtle quantitative changes. Additionally, the test set's class imbalance particularly affected severe stages, introducing evaluation bias. By aggregating all impaired classes (Mild + Moderate) into a single category against healthy controls (No Impairment + Very Mild), we maximize signal-to-noise ratio, improve statistical power, and address the most clinically critical question: detecting the presence of Alzheimer's disease, which is the actionable threshold for early intervention, regardless of precise staging. The final binary split resulted in 6,208 healthy images (53.9%) and 5,311 impaired images (46.1%), providing a reasonably balanced dataset.

For the binary classification experiments, we focused primarily on the MDL classifier as it yielded the best results among pure AIT methods, while also benchmarking against classical machine learning approaches to establish performance baselines.

Results

Pure AIT Classification (Four Classes)

The pure AIT methods performed poorly on the four-class problem. The MDL classifier achieved only 34.2% accuracy, with precision of 36.5%, recall of 34.2%, and F1-score of 33.8%. This poor performance demonstrates that raw complexity measures, while theoretically grounded, lack the discriminative power needed for fine-grained classification when disease stages show overlapping complexity distributions.

Feature Classification with Initial Feature Extraction (Four Classes)

Classifier	Accuracy	Precision	Recall	F1-score
Logistic Regression	0.65	0.64	0.66	0.65
Random Forest	0.74	0.74	0.74	0.74
Gradient Boosting	0.68	0.68	0.68	0.68
SVM	0.65	0.65	0.65	0.65

The initial feature extraction pipeline showed substantial improvement over pure AIT methods, with Random Forest achieving the best performance at 74% accuracy. However, these results remained below clinical requirements.

MDL Classifier with Updated Features (Four Classes)

After implementing the enriched 285-feature extraction pipeline, the MDL classifier improved significantly to 55.9% accuracy (precision: 57.6%, recall: 55.9%, F1-score: 55.5%). While still modest, this represented a 21.7 percentage point improvement over the basic MDL approach, demonstrating that richer feature representations enhance AIT-based methods.

Binary Classification Results

Model	Features	Accuracy	Precision	Recall	F1-Score
MDL	All	57.60%	59.31%	57.60%	57.16%
MDL	Top 100	65.54%	67.02%	65.54%	65.39%
Logistic Regression	All	90.58%	90.94%	90.58%	90.52%
Logistic Regression	Top 100	89.15%	89.41%	89.15%	89.09%
Random Forest	All	94.05%	94.63%	94.05%	94.01%
Random Forest	Top 100	93.88%	94.43%	93.88%	93.83%
SVM	All	89.89%	90.73%	89.89%	89.77%
SVM	Top 100	96.18%	96.23%	96.18%	96.17%

Cross-Validation Results (5-Fold):

- Random Forest: 92.77% \pm 0.76%
- Logistic Regression: 89.91% \pm 0.47%
- SVM: 88.74% \pm 0.76%

The binary classification task yielded dramatically improved results. SVM with feature selection achieved the highest test accuracy at 96.18%, while Random Forest demonstrated the most stable cross-validation performance. Feature selection using SelectKBest (top 100 features) had mixed effects: MDL benefited substantially (+7.94%), and SVM improved significantly (+6.29%), while Random Forest and Logistic Regression showed slight decreases, suggesting they were already utilizing the full feature space effectively.

Complexity Analysis

Compression ratio analysis across disease stages revealed unexpected patterns:

Four-Class Analysis:

- No Impairment: 0.5746 ± 0.0208
- Very Mild Impairment: 0.5823 ± 0.0251
- Mild Impairment: 0.5947 ± 0.0210
- Moderate Impairment: 0.5879 ± 0.0182

Binary Analysis:

- Healthy (No + Very Mild): 0.5783 ± 0.0233
- Impaired (Mild + Moderate): 0.5914 ± 0.0200

Contrary to our initial hypothesis, impaired brains showed higher compression ratios (lower complexity) than healthy brains, though the difference was modest (0.0131). Importantly, the progression across four stages was non-monotonic, with Moderate Impairment showing lower compression than Mild Impairment. This suggests that the relationship between neurodegeneration and algorithmic complexity is more nuanced than initially anticipated.

Discussion

At the beginning of the project, our objective was to evaluate whether Algorithmic Information Theory could extract meaningful structural differences between MRI images of healthy and impaired individuals. Our working hypothesis was that Alzheimer-affected brains should appear algorithmically simpler than healthy brains, leading to measurable reductions in compressibility-based complexity. The biological rationale was straightforward: neurodegeneration causes tissue loss, ventricular enlargement, and structural homogenization, all of which should reduce the information content encoded in brain images.

Our results provided partial validation of this hypothesis. The binary complexity analysis showed that impaired brains indeed exhibited higher compression ratios (0.5914) compared to healthy brains (0.5783), indicating lower algorithmic complexity. However, this difference was relatively small (0.0131), and the four-class analysis revealed non-monotonic progression, with complexity not decreasing uniformly across disease stages. This suggests that the relationship between neurological deterioration and algorithmic simplicity is more complex than a simple linear decline. Possible explanations include compensatory mechanisms in early stages, heterogeneity in disease progression patterns, or artifacts introduced by synthetic data augmentation in the dataset.

When used as the basis for classification, pure AIT models performed far below expectations. The MDL classifier achieved only 34.2% accuracy in four-class classification and 57.6% in binary classification, demonstrating that raw complexity measures lack sufficient discriminative power for clinical deployment. This poor performance can be attributed to the

fact that complexity differences between classes, while statistically significant, are too subtle to serve as reliable decision boundaries when used in isolation.

In stark contrast, after designing a rich 285-feature AIT-inspired representation, classical machine learning approaches performed dramatically better. Random Forest achieved 94.05% accuracy in binary classification, while SVM with feature selection reached 96.18%. Cross-validation results confirmed the robustness of these models, with Random Forest showing remarkable stability at $92.77\% \pm 0.76\%$. This demonstrates that while raw complexity metrics are insufficient as standalone classifiers, they become highly effective when transformed into engineered features capturing multi-scale, multi-region, and multi-metric information. The synergy between AIT-derived features and traditional machine learning methods proved far more powerful than either approach alone.

Despite these encouraging results, the approach presents several important limitations that must be acknowledged. First, compression algorithms only approximate Kolmogorov complexity and remain bound by algorithm-dependent biases. For instance, gzip favors repeated byte patterns and may not capture structural regularities that other compressors would detect. This means our complexity measures are implementation-dependent rather than universal. Second, compression metrics proved insufficiently discriminative across disease stages. The small differences in compression ratios between classes, combined with substantial variance within each class, limited the separability achievable through methods like NCD and MDL. Third, applying compression-based distances to high-dimensional flattened MRI images led to significant computational challenges, including heavy memory usage, long computation times, and numerical instability. The BNN-like classifier became completely infeasible for our dataset size. Fourth, dataset imbalance in certain test subsets introduced evaluation bias, potentially inflating accuracy for overrepresented classes while underestimating performance on rare categories. Finally, and perhaps most fundamentally, compression focuses on pixel-level redundancy and byte-pattern regularities, whereas medically meaningful biomarkers such as hippocampal atrophy and cortical thinning are spatially localized anatomical changes. These may not correlate directly with overall image compressibility, as critical pathological changes could be masked by the dominance of non-affected regions in global compression measures.

Perspectives and Future Directions

Several promising research directions emerge naturally from this work. First, testing the approach on longitudinal MRI data would enable evaluation of algorithmic complexity as a biomarker of progressive cognitive decline. Tracking complexity changes within individual patients over time might reveal patterns invisible in cross-sectional comparisons and could provide early warning signals for accelerated deterioration. Second, integrating clinical metadata such as age, APOE genotype, and MMSE cognitive scores could enable the construction of truly multimodal predictive models that combine structural complexity with established risk factors and functional assessments.

From a methodological perspective, improving the MDL prototype construction represents an accessible enhancement. Rather than using simple arithmetic means of training images, employing medoid-based prototypes or multiple centroid representations through clustering could better capture intra-class variability and improve classification robustness. More ambitiously, applying AIT principles to learned representations rather than raw pixels may unlock new potential. Computing NCD or compression ratios on feature embeddings extracted by pretrained MRI models such as ResNet or DenseNet could reveal whether

algorithmic complexity aligns better with high-level semantic features than with low-level pixel patterns. This would bridge the gap between end-to-end deep learning and interpretable AIT-based analysis.

Developing hybrid AIT-ML architectures represents another fertile direction. Rather than treating AIT features as static inputs to classical models, we could design systems where complexity growth, multiscale compression dynamics, entropy evolution, and patch variability serve as complementary signals in ensemble frameworks. Such systems might leverage the interpretability of AIT while exploiting the pattern recognition strengths of modern machine learning. Finally, shifting from whole-image analysis to region-based complexity measurement could dramatically improve biological relevance. By computing complexity measures specifically within anatomically defined regions of interest, we could better capture the spatially heterogeneous nature of neurodegeneration and potentially identify novel imaging biomarkers tied directly to the known neuropathology of Alzheimer's disease.

Bibliography

Dataset: Best Alzheimer's MRI Dataset (Kaggle) -

<https://www.kaggle.com/datasets/lukechugh/best-alzheimer-mri-dataset-99-accuracy>

Github Repository:

<https://github.com/Nawel08/Complexity-Loss-as-a-Biomarker-for-Alzheimer-s-Disease>



September, 2025

CSC 5AI25 TP

Algorithmic Information & Artificial Intelligence

Micro-study

teaching.dessalles.fr/FCI

Name: Leontev Mikhail

Complexity analysis of Mario Bros levels

Abstract

The project aims to develop a quantitative method for evaluating the descriptive complexity of game levels, drawing on concepts from Algorithmic Information Theory and computational complexity. To achieve this, I employed three primary methods of evaluation: compression analysis to estimate Kolmogorov complexity, path analysis to examine navigational challenge, and fourgram structural analysis to capture local motif diversity. This multi-faceted approach grounds level analysis in established theoretical frameworks while addressing practical aspects of game design and evaluation.

Problem

“Mario Bros.” (*Nintendo, 1985*) is a platformer video game. In this game, players control the character (Mario) to navigate through various obstacles and ultimately reach the end of the level. The Mario games franchise is the best-selling video game franchise of all time, according to Wikipedia (*Wikipedia, 2020*). Due to its simplicity, the game has generated significant interest, as it is used to experiment with various Procedural Content Generation (PCG) techniques and test game mechanics.

The problem faced by game developers is finding an optimal way to evaluate the difficulty and complexity of video game levels so that they are both appealing and surprising to players. Currently, there are a few key methodologies, including user testing, bots, and the use of static functions and algorithms for evaluation (*Schaa and Barriga, 2024*).

To address the level-quality evaluation problem, I apply computational complexity techniques—grounded in Algorithmic Information Theory (AIT)—to Mario Bros. levels to produce objective, reproducible metrics that complement playtesting. In particular, I employ the size of zlib-compressed (*Gailly and Adler , 2024*) data as a practical instantiation of descriptive Kolmogorov complexity, leveraging the AIT concept that a string's compressibility reflects its algorithmic randomness and structural regularity. I further connect AIT principles to gameplay analysis by comparing A* path lengths to recorded human play traces, thereby relating navigational complexity to the inherent unpredictability of optimal versus observed paths. The extraction and analysis of 2×2 tile motifs (fourgrams) operationalises the idea of local information content and diversity, as explored in AIT, where greater motif variety indicates increased structural complexity. Combined, these measures create an AIT-informed framework that not only highlights repetitive or overly predictable regions and identifies bottlenecks and rare structural motifs, but also provides theoretically motivated, quantitative signals to guide PCG parameter tuning, support level selection for user studies, and enable further validation against subjective difficulty rating

Method

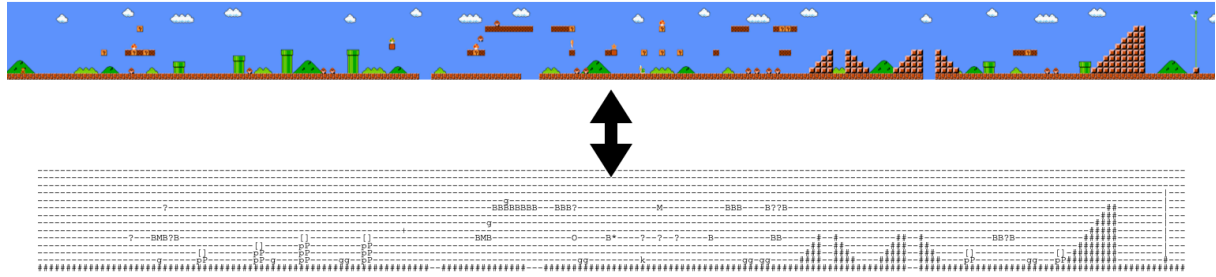


Figure 1: mario levels textual representation

Dataset

To evaluate Mario levels, I have been using the Video Game Level Corpus (VGLC) dataset (*Summerville et al., 2016*). The dataset itself represents the level as a two-dimensional array of structures, where each element corresponds to a specific level element, as shown in Figure 1. In total dataset contains 13 parsed levels respectively: 'mario-1-1', 'mario-2-1', 'mario-3-1', 'mario-3-2', 'mario-4-1', 'mario-5-1', 'mario-5-2', 'mario-6-1', 'mario-6-2', 'mario-7-1', 'mario-8-1', 'mario-8-2', 'mario-8-3' in a chronological order as how expected player to encounter this levels in the game.

Compression approach

First, I tried to apply compression and evaluate the compression ratio using the Python *zlib* library (*Gailly and Adler , 2024*). Each level was compressed in turn and then compared with the original version.

$$r = \frac{N_{compressed}}{N_{original}} \quad (1)$$

Path analysis approach

I analysed level difficulty by calculating the number of steps needed, using A* algorithm (Madlberger, Popov and Jaqubi, 2023) on the parsed grid, and extracting human traces from player path files, originating from the same VGLC dataset. I derived metrics like A* path length, human path length (mean/median), differences and ratios. Start/goal are chosen following the canonical rule (first passable tile in the left column, reachable goal near the right). Results are normalised by level size for fair comparison, and path overlays are saved for inspection. I computed correlations and filtered outliers when relating path metrics to compression and fourgram measures.

Fourgrams and bigrams

I proposed using bigrams and fourgrams for the analysis of levels' structure. Bigrams are adjacent tile pairs (a simple linear (n)-gram of length two) that capture local sequential relationships along rows or columns; fourgrams are the two-by-two tile blocks that capture the smallest planar spatial motifs.

Counting unique bigrams/fourgrams and their frequencies measures local structural variety and repetitiveness: many repeated n-grams indicate high local redundancy (more compressible), while many unique n-grams indicate richer, less predictable local structure (less compressible). The idea is to use these counts (and normalised measures such as n-grams per tile) as interpretable proxies for level complexity and to relate them to compression metrics and path difficulty—e.g., levels with more rare fourgrams may force more novel navigation decisions.

Results

Compression results

Compression of the level files yields a mean compression ratio of 0.0828 (levels retain $\approx 8.3\%$ of their original bytes), corresponding to roughly a $12\times$ reduction in size ($\approx 91.7\%$ reduction). There is effectively no linear relationship between level order and compressibility (Pearson $r = -0.0037$, $N = 13$), so later levels are not consistently more or less compressible than earlier ones. Some levels are much more compressible than others: Mario-3-2 (0.0605) and Mario-4-1 (0.0618) show a strong reduction, while Mario-6-2 (0.1085) and Mario-7-1 (0.0997) are relatively less compressible (See Figure 2). These results indicate measurable differences in structural repetitiveness across levels, but no support for a monotonic increase of proxy complexity with level progression.

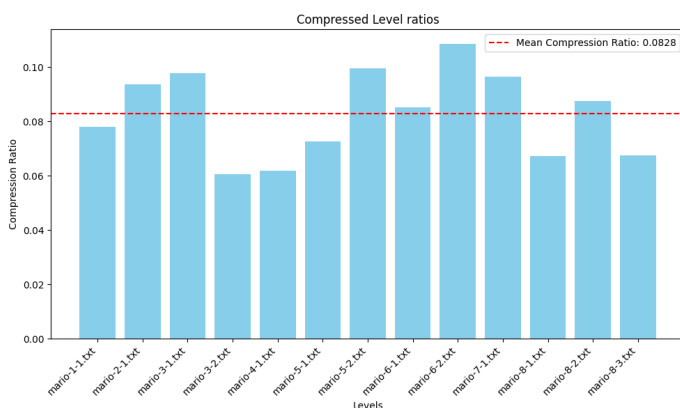


Figure 2: Compression ratio per level

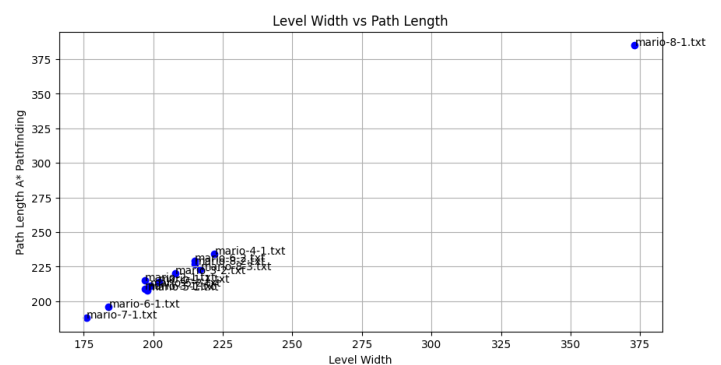


Figure 3: Relationships between path length and level width.

Path analysis analysis results

Comparing A* path lengths to recorded human traces reveals consistent differences in navigation behaviour. Across the 13 levels in the dataset, A* typically finds paths that are equal to or shorter than those taken by humans, reflecting optimal shortest-path behaviour rather than exploration and gameplay actions visible in human traces. Results showed that the mean value of normalised paths (length divided by level width) was approximately 1.058 for the A* algorithm and about 1.099 for player-created paths. This suggests that the usual number of steps needed to complete a level is very close to its width. Therefore, there are nearly linear relationships between path length and level width, as demonstrated in Figure 3. Another finding is a weak positive correlation of approximately 0.3057 between the level order and path length, indicating that, generally, as players progress through the game, more steps are required to complete it.

Bigram and Fourgram results

The approach of using bigrams and fourgrams as a proxy to measure the complexity of the level showed some potential and correlated with the compressibility ratio. The results demonstrated that fourgrams could depict greater structural diversity in Mario game levels, with 249 unique fourgrams extracted across all 13 levels, compared to only 90 bigrams.

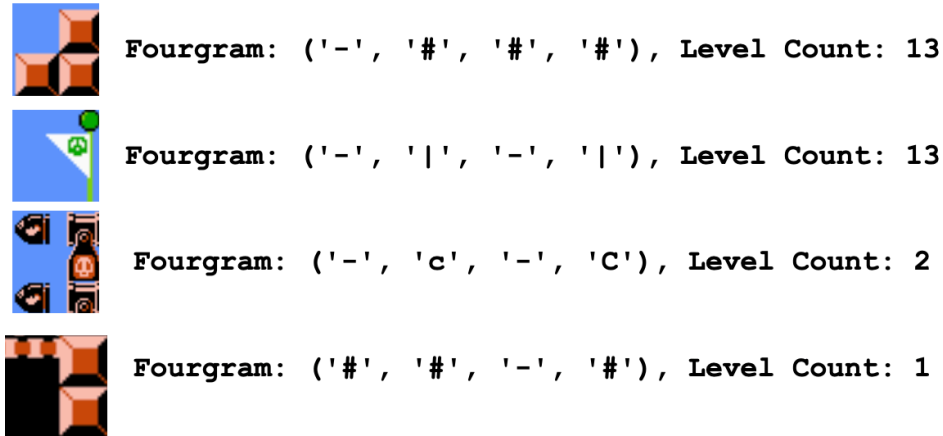


Figure 4: Fourgrams examples

It was noticed that some fourgrams are much more common than others; thus, the empty space fourgram (“-”, “-”, “-”, “-”) can be encountered 29171 times across all levels, whereas others appear only a couple of times. At the same time, some structures were level specific; thus, for example, stair fourgram (“-”, “#”, “#”, “#”) can be found in all the levels, along a flagpole, hence it can be considered not surprising structure, in contrast to two guns on top (“-”, “c”, “-”, “C”) of each other, which exists only in two levels or bridge made of solid unbreable blocks (“#”, “#”, “-”, “#”), which can only be found once in one particular levels, what makes it unique and surprising structure (See Figure 4).

As the measure of level complexity, I propose using a unique fourgram pet tile as follows:

$$U_p = \frac{N_{unique}}{H \times W} \quad (2)$$

This metric can be used as a proxy for complexity, where N_{unique} the number of unique fourgrams in a level is normalised by the total number of objects in a level. Larger values of U_p indicate that the fourgram appears in a wider variety of local structural contexts and therefore contributes to a greater perceived level of structural complexity. Hence, this metric can be used alongside other approaches to evaluate the structural complexity of game levels, presented in Table 1.

Another metrical value that can be considered is the count of rare fourgrams, which are unique to this level and not present in others. Consequently, it can be inferred that if a specific level exhibits a high frequency of rare fourgrams, then this level is likely to be more complex and distinct from the others.

Level	U_p	N_{unique}	Rare fourgram
mario-5-2.txt	0.044867	125	25
mario-3-1.txt	0.039521	109	10
mario-7-1.txt	0.037744	93	5
mario-2-1.txt	0.036258	100	6
mario-6-2.txt	0.032558	98	18
mario-8-2.txt	0.031229	94	8
mario-6-1.txt	0.028339	73	5
mario-1-1.txt	0.027935	79	7
mario-5-1.txt	0.025613	71	2
mario-8-3.txt	0.023700	72	0
mario-3-2.txt	0.023008	67	0
mario-4-1.txt	0.021557	67	2
mario-8-1.txt	0.015511	81	1

Table 1: Fourgram statistical analysis

Discussion

Evaluation result

Overall, the suggested metrics for proxy evaluation of Kolmogorov complexity showed some potential. Usage of compressability and fourgram models allowed for the evaluation of structural diversity of the levels. In addition, fourgrams can be used to evaluate unexpectedness and to analyse the probability of encountering which structures next. Also, a strong correlation between unique fourgrams per tile and compressibility was found, with a value of ≈ 0.8540 , indicating a strong positive correlation. Hence, it is fair to assume that the more complex level is in terms of structural diversity, the less compressible it is.

Another important point is that neither method showed a strong correlation between the order of levels and their complexity; there is no guarantee that the next level will be more complex than the previous one. Specifically, the most complex levels are encountered at the mid-game, specifically levels Mario-5-2 and Mario-6-2 had the highest number of rare fourgrams, and Mario-5-2 also had the highest score of unique per tile fourgrams. At the same time, these two levels were the least compressible ones, hence it is fair to assume that these two levels can be considered the most complex ones.

Limitations

The project has several notable limitations across all three methods.

First, the relatively small dataset of only 13 levels, as the methods were tested on a single dataset and just one level of the game. For greater consistency, the proposed methods could also be applied to other game datasets and titles.

Second, there are limitations regarding the compressibility approach, since only one compressor was used. It would be highly valuable to evaluate and compare results from different compressors to obtain more robust final outcomes.

Third, the restriction concerns the path-evaluation approach, which simplifies game mechanics and consequently neglects enemy movement, bonuses, and other vital elements. In some cases, more steps might be needed to complete a level than initially estimated. Unlike the A* pathfinder or player path analysis, paths generated by a reinforcement learning agent could be more accurate, similar to the approach adopted by the PyTorch team (*Pytorch Team, 2024*).

Finally, four-grams with limited receptive fields—such as 2×2 four-grams—only capture very local patterns and overlook mesoscale or global structures like staircases or long corridors. To mitigate this, analyse larger blocks such as 3×3 or 4×4 , or adopt a hierarchical approach motifs. Additionally, we need to explore different methods for evaluating four-grams, since the way tiles are labelled or clustered significantly affects the counts of unique items. To mitigate this, we should test alternative tile groupings and report the sensitivity of the results.

Overall conclusion

The proposed AIT-inspired toolkit (zlib compressibility, A* versus player path analysis, and 2×2 fourgram motifs) offers complementary, testable proxies for descriptive complexity in Mario levels. Empirically, unique fourgrams per tile correlate strongly with compressibility (≈ 0.8540), and rare or unique fourgram counts help identify genuinely novel structures (notably mario-5-2 and mario-6-2). This supports the idea that greater local structural diversity is associated with lower compressibility. Neither level order nor simple A*/player path reliably represent increasing levels of complexity. Key limitations — such as small sample size, codec dependence, simplified path dynamics, and fourgram locality — temper strong conclusions; extending the corpus, comparing compression algorithms, employing physics-aware agents, and analysing larger or hierarchical motifs are necessary next steps. In summary, fourgrams are a useful, empirically supported proxy for structural complexity but require broader validation before being adopted as a definitive measure.

Bibliography

1. Gailly, J. and Adler, M. (2024). zlib 1.3.1. [online] www.zlib.net. Available at: <https://www.zlib.net/>.
2. Madlberger, T.A., Popov, F. and Jaqubi, M. (2023). A* Algorithms. [online] Available at: https://ecer.pria.at/archive/ecer-2023/papers/A__Algorithms.pdf [Accessed 30 Nov. 2025].
3. Nintendo (1985). Super Mario Bros. [online] The official home for Mario - History. Available at: <https://mario.nintendo.com/history/>.
4. Pytorch Team (2024). Train a Mario-playing RL Agent — PyTorch Tutorials 2.7.0+cu126 documentation. [online] Pytorch.org. Available at: https://docs.pytorch.org/tutorials/intermediate/mario_rl_tutorial.html.
5. Schaa, H. and Barriga, N.A. (2024). Evaluating the Expressive Range of Super Mario Bros Level Generators. Algorithms, [online] 17(7), pp.307–307. doi:<https://doi.org/10.3390/a17070307>.
6. Summerville, A.J., Snodgrass, S., Mateas, M. and Villar, S.O. (2016). The VGLC: The Video Game Level Corpus. arXiv (Cornell University), (7). doi:<https://doi.org/10.48550/arxiv.1606.07487>.
7. Wikipedia. (2020). List of best-selling video game franchises. [online] Available at: https://en.wikipedia.org/wiki/List_of_best-selling_video_game_franchises.



November, 2025

CSC_5AI25_TP

Algorithmic Information & Artificial Intelligence

Micro-study

teaching.dessalles.fr/FCI

Name: Yufei Zhou, Stepan Svirin

Can NCD Help Prevent Bug Propagation?

Abstract

Bug propagation is often caused by copy paste reuse: the same buggy logic can reappear elsewhere after minor edits, making it hard to locate with exact matching. In this study we evaluate a lightweight clone search method based on Normalized Compression Distance (NCD), where similarity is estimated by how well two code snippets compress together. Using benchmarks built from real Python repositories with semantics preserving mutations, we show that NCD based search can reliably recover the injected clones.

Problem

Software bugs often propagate through code duplication. A developer fixes a bug in one location, but the same logic has already been copied elsewhere and slightly adapted. These copies are rarely exact: identifiers are renamed, constants are tweaked, and small refactorings change the surface form while preserving the underlying pattern. As a result, simple text matching is brittle, while heavier program analysis may be too expensive to run repeatedly during development, making it easy to miss other bug instances and allowing the defect to persist.

The problem we address is: *given a known buggy snippet, how can we quickly locate other fragments in the same repository that implement the same pattern, despite superficial edits?* Following the idea of Normalized Compression Distance (NCD)[1] search, we treat code as text and measure similarity via compression: if two snippets share structure, compressing their concatenation adds little extra information. Our goal is to test whether this lightweight

approach can reliably retrieve mutated clones as potential bug-propagation sites, and to study the impact of key design choices (e.g. compressor choice and optional identifier skeletonisation).

Method

Dataset Generation

To evaluate clone retrieval with clear ground truth, we generate synthetic but realistic benchmarks directly from real open source Python repositories (e.g. `flask`[2] and `requests`[6]). The core requirement is that each query has at least one known matching location inside a target repository, while the corresponding clone is modified by semantics preserving edits that mimic copy and adapt behavior. We considered two dataset variants: a simpler benchmark where injected clones are stored in a dedicated location, and a harder combined benchmark with partial queries, richer mutations, clones scattered across many files, and additional unlabeled decoys.

Function Extraction

We start by extracting real functions using the Python AST. We traverse all `.py` files in the chosen repository, parse them with `ast.parse`, and collect `FunctionDef` nodes. To avoid trivial queries, we keep only non-trivial functions above a minimum length. We then sample a set of functions at random to construct the query set.

In the combined benchmark, queries are *partial body snippets* rather than whole functions. For each sampled function we cut a contiguous slice of its body into a snippet of a few lines. These fragments do not need to be executable as standalone code, but they better reflect bug hunting practice, where the starting point is often a small region around the bug.

Code Mutation

For each selected function, we generate a mutated clone intended to preserve behavior while changing surface form. Mutations are applied at the AST level to avoid producing syntactically invalid code. We always rename the function name to a fresh identifier to avoid collisions, and we rename parameters and local variables using a mapping derived from argument lists and assignment targets.

We use a family of semantics preserving edits:

- **Identifier renaming:** rename locals and parameters while preserving control flow and operations.
- **No-op and constant rewriting:** insert dead blocks guarded by constant conditions (e.g. `if False`) and rewrite integer literals into equivalent expressions.
- **Control-flow rewrites:** apply safe transformations such as branch inversion with swapped blocks, splitting conjunctions into nested conditionals, and wrapping statements under `if True`.
- **Additional noise (combined benchmark):** conservative statement reordering for independent constant assignments and dead logging guarded by `if False`.

Figure 1 illustrates the three core mutation families on a toy example.

Original

```
def foo(a, b):  
    if a > 1:  
        return b + 2  
    else:  
        return 0
```

RenameVars

```
def mutated_func_1(var_0, var_1):  
    if var_0 > 1:  
        return var_1 + 2  
    else:  
        return 0
```

RenameVars + Noop + Consts

```
def mutated_func_2(var_0, var_1):  
    if False:  
        pass  
    if var_0 > 1:  
        return var_1 + (1 + 1)  
    else:  
        return 0
```

RenameVars + ControlFlow

```
def mutated_func_3(var_0, var_1):  
    if True:  
        if not (var_0 > 1):  
            return 0  
    else:  
        return var_1 + 2
```

Figure 1: Illustration of mutation strategies: an original function (left) and three semantics preserving mutations (right).

Combined Dataset

The combined benchmark is designed to approximate the bug hunting scenario more closely. First, the queries are partial body snippets, which increases ambiguity because many unrelated functions can share short structural patterns. Second, mutated clones are scattered across many Python files rather than being placed in a single dedicated file, which forces the retrieval method to search the whole codebase.

To make ranking non-trivial, we also inject additional hard negative decoys. These decoys are mutated functions produced by the same transformation pipeline, but they are intentionally left unlabeled, so they do not correspond to any query answer. This creates realistic confounders: the retrieval method must distinguish the true injected clone from other near-duplicate fragments that look similar under the same mutation style.

For each query we store ground truth as a relative file path and a line interval describing where the clone was injected. This allows evaluation with ranking metrics (HIT@K and MRR) while keeping the benchmark generation fully automatic and reproducible.

Our Approach

Figure 2 provides an overview of the pipeline of our approach, which we describe in this section. For each query snippet in our dataset, we apply a skeletonisation step that masks most identifiers while preserving control-flow and structural keywords, and then encode the resulting text as bytes.

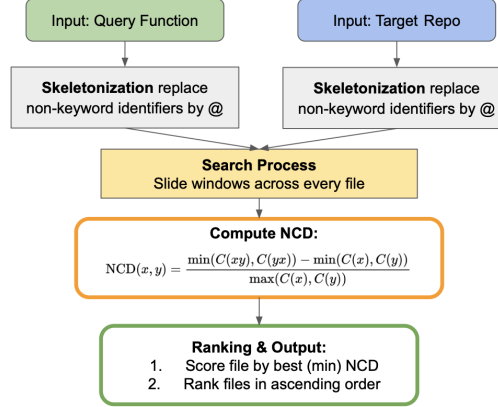


Figure 2: Overview of our compression-based clone detection pipeline. Query snippets are (optionally) skeletonised, compared via NCD to sliding windows over the target repository, and the files are ranked by their best (lowest) NCD score.

We then search through all Python files of the repository using a sliding-window approach: for each file, we consider code windows and compute a Normalized Compression Distance (NCD) between the query and every window. Window size determines how much context we compare the query against and thus strongly affects both accuracy and cost. Very small windows capture only short phrases and create many wrong matches, while very large windows dilute the true clone with unrelated surrounding code and can hide good matches. Possible strategies include a single fixed window size, token/AST-based windows, or multi-scale windows. We are using an adaptive multi-window scheme: for each query we estimate its length in lines and scan each file with several window sizes that are simple multiples of this length (e.g. 0.8, 0.9, 1.0, 1.1, 1.2). For every file we keep the minimum NCD over all windows, which makes the method tolerant to small insertions/deletions around the clone while avoiding the mismatch of a rigid global window size.

After estimating the best NCD observed, we rank all candidate files globally in ascending order of this score, yielding a top- k list of most similar locations per query.

Skeletonization

One of the design choices in our pipeline was to introduce a skeletonisation step before computing similarities. The main idea is to map code snippets to a more abstract representation that preserves their control-flow and syntactic structure, while reducing most of the surface-level naming information. We tokenize each snippet and replace non-keyword identifiers (variable names, function names, attribute names) by a generic placeholder symbol (e.g. @), while keeping Python keywords, operators and punctuation unchanged. The main motivation for introducing skeletonisation was to make our similarity measures less sensitive to edits and vocabulary differences. In realistic settings, copy-pasted buggy code is often modified by renaming variables, changing parameter names or adapting local identifiers to project conventions, while the underlying control-flow pattern remain intact. If we compare raw token

<pre> def sum_prices(prices): total = 0 for price in prices: total += price return total </pre>	<pre> def @(@): @ = 0 for @ in @: @ += @ return @ </pre>
---	--

Figure 3: Example of identifier skeletonisation.

sequences directly, these renames inflate the apparent distance between two semantically similar snippets. By collapsing identifiers to a placeholder, we hoped to (i) reduce the impact of arbitrary renaming, (ii) mitigate the explosion of distinct tokens across large codebases, and (iii) emphasise structural similarity (loops, conditionals, branching patterns) over local naming choices.

For NCD, the intuition was that two skeletonised snippets implementing the same pattern should compress much better together than two structurally unrelated snippets, even if all original identifiers have been rewritten. For token-based baselines such as Jaccard, skeletons were expected to increase overlap when clones use different names but the same control-flow template. This is why skeletonisation was initially integrated as a pre-processing stage in our clone-search pipeline.

Normalized Compression Distance

To measure the similarity between a query snippet and a candidate code window, we use the *Normalized Compression Distance* (NCD). Let $Z(x)$ denote the compressed length (in bytes) of a string x under a fixed compressor. The intuition is that if two strings x and y share a lot of structure, then the compressed size of their concatenation should be close to the compressed size of each one individually, because the compressor can reuse patterns learned from x when encoding y (and conversely).

We use a symmetric variant of NCD:

$$\text{NCD}(x, y) = \frac{\min(Z(xy), Z(yx)) - \min(Z(x), Z(y))}{\max(Z(x), Z(y))}.$$

Classical NCD is usually written with $Z(xy)$ only, but for practical compressors the concatenation is order-dependent: $Z(xy)$ and $Z(yx)$ can differ because of the internal state and sliding window of the compression algorithm. In our setting there is no natural notion of "first" and "second" snippet, so we take $\min(Z(xy), Z(yx))$ to enforce symmetry and reduce artefacts caused purely by the compressor's directionality. The numerator then measures how much additional compressed length is required to encode x and y together, beyond the cheaper of the two individually, while the denominator normalises by the size of the larger snippet to keep the distance in a roughly comparable range across different lengths. We experimented with three standard compressors for computing $Z(\cdot)$: LZMA [4], BZ2 [3], and zlib [5].

Lower NCD values correspond to higher similarity. In our retrieval pipeline, we use this score to rank candidate windows for each query snippet.

Evaluation and Baselines

We evaluate our method on the clone benchmark described in "Data Generation" Section using standard ranking metrics. For each query snippet we obtain a ranked list of candidate files (one score per file, given by the best window inside that file). We then compute HIT@K (fraction of queries for which the ground-truth file appears in the top K positions, for $K \in \{1, 3, 10\}$) and the Mean Reciprocal Rank (MRR), defined as the average of $1/\text{rank}$ over all queries where the correct file is retrieved. We also record runtime to compare the computational cost of different variants.

To put NCD in context we compare it against a simple baseline based on *Jaccard similarity*. For each code snippet we build a set of normalized tokens: Python keywords are kept, while all other identifiers are mapped to `@`, numbers to `NUM` and string literals to `STR`. Given two snippets with token sets A and B , the Jaccard similarity is

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|},$$

i.e. the proportion of shared tokens among all distinct tokens. Higher values indicate that the snippets use a similar multiset of (normalized) tokens. In the baseline, we score each function in the repository by its Jaccard similarity to the query token set, take the maximum score per file, and rank files in descending order. This gives a lightweight, bag-of-tokens clone detector against which we can compare the more expressive, structure-aware NCD approach.

Results

In this section we present the experimental results as well as the comparison of the approaches.

Comparison of Compressors

To understand how the choice of compressor affects NCD-based retrieval, we repeated the same experiment with three standard compressors for computing $Z(\cdot)$: LZMA, BZ2 and `zlib`. The rest of the pipeline was kept identical (same dataset, window search, symmetric NCD, and ranking). For each compressor we report HIT@1/3/10 and Mean Reciprocal Rank (MRR), runtime is summarised qualitatively as slowest/fastest/medium since we observed stable relative ordering.

Compressor	HIT@1	HIT@3	HIT@10	MRR	Time
LZMA	42 (84.0%)	46 (92.0%)	48 (96.0%)	0.876	slowest
BZ2	37 (74.0%)	40 (80.0%)	46 (92.0%)	0.787	fastest
Zlib	42 (84.0%)	47 (94.0%)	47 (94.0%)	0.880	medium

Table 1: Comparison of compressors on the custom clone-detection benchmark.

Table 1 shows that *LZMA* and *zlib* achieve very similar quality, both reach 42 successful queries at top-1 and HIT@10 of 48/47 with MRR of 0.876 and 0.880 respectively. BZ2 stays behind in accuracy (HIT@1 = 37, MRR = 0.787), although it is the fastest option.

Influence of Skeletonisation

To explore the effect of identifier skeletonisation on our approach, we compare three configurations: (i) raw code without any skeletonisation, (ii) skeletonisation with a small keyword set, and (iii) skeletonisation with an extended keyword set including all Python keywords

and a few common library tokens. In both skeletonised variants, non-keyword identifiers are mapped to `@` before computing NCD, while the search procedure and compressor (`zlib`) remain unchanged.

Config	#Keywords	HIT@1	HIT@3	HIT@10	MRR
No Skeleton	0	1.00	1.00	1.00	1.000
Skeleton Base	24	0.86	0.92	0.96	0.901
Skeleton Extended	52	0.90	0.96	0.98	0.932

Table 2: Comparison of configurations by hit rates and number of keywords.

Table 2 suggests that on this synthetic benchmark, NCD on raw code achieves perfect scores ($\text{HIT@1/3/10} = 1.00$, $\text{MRR} = 1.000$). Both skeletonised variants perform slightly worse: the base keyword set yields $\text{HIT@1} = 0.86$ and $\text{MRR} = 0.901$, while the extended set recovers part of the loss ($\text{HIT@1} = 0.90$, $\text{MRR} = 0.932$) but still does not reach the raw configuration.

Comparison of Clone-Search Approaches

We finally compare our NCD-based search against a simpler token-based Jaccard baseline, both with and without skeletonisation. Table 3 shows the results on the synthetic benchmark, plain NCD on raw code achieves perfect retrieval ($\text{HIT@1/3/10} = 1.00$, $\text{MRR} = 1.00$) with a runtime of about 215 seconds for a total of 100 queries. Adding skeletonisation slightly worsens quality ($\text{HIT@1} = 0.94$, $\text{HIT@10} = 0.98$, $\text{MRR} = 0.96$) and is actually slower (≈ 254 seconds). The Jaccard baseline, computed on normalised token sets, reaches only $\text{HIT@1} = 0.66$ and $\text{MRR} = 0.76$, with runtime comparable to NCD with skeletons.

Method	Skeletonization	HIT@1	HIT@3	HIT@10	MRR	Time (s)
NCD	No	1.00	1.00	1.00	1.00	214.8
NCD	Yes	0.94	0.96	0.98	0.96	254.1
Jaccard	Yes	0.66	0.84	1.00	0.76	252.8

Table 3: Comparison of NCD vs a Jaccard baseline on the clone benchmark.

Results presented in the Table 4 explore the even more realistic combined benchmark (partial snippets, scattered clones and hard negatives), the gap widens. NCD on raw code still performs strongly ($\text{HIT@1} = 0.88$, $\text{HIT@3} = 0.94$, $\text{HIT@10} = 0.96$, $\text{MRR} = 0.91$) at around 119 seconds, while NCD with skeletonisation drops to $\text{HIT@1} = 0.72$ and $\text{MRR} = 0.78$. The token Jaccard baseline collapses in this setting ($\text{HIT@1} = 0.06$, $\text{HIT@10} = 0.38$, $\text{MRR} = 0.17$) and is also slower (≈ 263 seconds).

Method	Skeletonization	HIT@1	HIT@3	HIT@10	MRR	Time (s)
NCD	No	0.88	0.94	0.96	0.91	119.3
NCD	Yes	0.72	0.80	0.88	0.78	118.4
Jaccard	Yes	0.06	0.16	0.38	0.17	263.2

Table 4: Comparison of NCD vs a Jaccard baseline on the clone benchmark.

Discussion

What is the most suitable compressor?

The compressor comparison suggests that, for our clone-search task, the exact choice of compressor matters less than the overall NCD formulation, but there is a clear trade-off between accuracy and runtime. LZMA achieves very similar retrieval quality to `zlib`, but its slower speed makes exhaustive sliding-window search impractical: in our setup, running LZMA for just 50 queries took more than one hour, whereas the same experiment with `zlib` finished in a few minutes. BZ2 is attractive from a computational perspective (fastest runs), but the noticeable drop in HIT@1 and MRR indicates that its compressed lengths $Z(\cdot)$ are a less informative proxy for code similarity in this setting.

`zlib`/DEFLATE therefore strikes the most balanced compromise. It matches or slightly improves on LZMA in terms of MRR, achieves essentially the same HIT@K scores on our benchmark, and is dramatically faster and lighter-weight. Since our pipeline may need to scan large repositories and recompute NCD scores many times, this balance between retrieval quality and computational cost is crucial. For this reason we adopt `zlib` as the default compressor for all subsequent experiments.

Why Skeletonisation Didn't Help

Although identifier skeletonisation was designed to make our method more robust to renaming and superficial edits, it does not improve results on our synthetic clone benchmark and in fact slightly degrades both accuracy and runtime. Qualitatively, skeletonisation introduces extra preprocessing work (tokenisation and rewriting) while leaving the overall search unchanged: we still slide the same windows over the same files and compress strings of almost the same length. As a result, we pay an additional CPU cost on every window without reducing the number of NCD evaluations or the amount of data passed to the compressor, which explains the longer execution time.

From an accuracy perspective, the benchmarked clones still share many informative tokens that our compressor-based NCD can already use, even in the presence of variable renaming. By collapsing most identifiers to `@`, skeletonisation removes part of this signal and makes unrelated functions with similar control-flow look more alike, which increases confusion and slightly lowers HIT@K and MRR.

There are nonetheless scenarios where skeletonisation could be useful. In large cross-project search or heavily modified code, where identifier names carry little stable information, a skeleton-based view can emphasise recurring control structures and shrink the token vocabulary. Combined with extra tricks (e.g. deduplicating identical skeletons or using them as a pre-filter before NCD), this might also reduce runtime by narrowing the candidate set. In our setting, however, benchmarks still contain informative surface tokens, so plain NCD on raw code remains both more accurate and more efficient.

Comparison of Approaches

The method comparison confirms that compression-based NCD on raw code is the most effective clone-search strategy in our setup. It dominates both skeletonised NCD and the Jaccard baseline on all metrics, and remains competitive in runtime. Skeletonisation does not act as a useful abstraction layer here: it removes informative surface tokens without shortening the inputs or reducing the number of NCD evaluations, which explains why scores deteriorate and runtime slightly increases.

The token-based Jaccard baseline is simple but ignores ordering and structure, so with partial snippets, scattered clones and hard negatives it rarely ranks the true clone near the top. In contrast, raw-code NCD remains robust in these realistic conditions. In our experiments, it clearly outperforms both skeletonised NCD and Jaccard, so we adopt raw NCD as the primary method.

Conclusions

Our experiments show that NCD-based search is a viable way to detect bug propagation. It reliably retrieves heavily mutated clones across both realistic benchmarks. Among the compressors we tested, `zlib` offers the best trade-off between retrieval quality and runtime, making it a natural default for practical use. Finally, in our setting, identifier skeletonisation does more harm than good: it largely ruins useful signal without reducing computation, and consistently degrades retrieval performance compared to using raw code with NCD.

Bibliography

References

- [1] C.H. Bennett, P. Gacs, Ming Li, P.M.B. Vitanyi, and W.H. Zurek. Information distance. *IEEE Transactions on Information Theory*, 44(4):1407–1423, 1998.
- [2] Pallets Team and contributors. Flask. GitHub repository, 2025. Accessed: 2025-11-28.
- [3] Python Software Foundation. bz2 — support for bzip2 compression. <https://docs.python.org/3/library/bz2.html>.
- [4] Python Software Foundation. lzma — compression using the lzma algorithm. <https://docs.python.org/3/library/lzma.html>.
- [5] Python Software Foundation. zlib — compression compatible with gzip. <https://docs.python.org/3/library/zlib.html>.
- [6] Python Software Foundation and contributors. Requests: Http for humans. GitHub repository, 2025. Accessed: 2025-11-28.

Algorithmic Information & Artificial Intelligence

Micro-study

teaching.dessalles.fr/FCI

Name: Yixing YANG Ziyi LIU

Lyric-Aware Compression for Improving NCD-Based Clustering

Abstract

Song lyrics differ from ordinary prose because they mix linguistic structure with rhythmic form, repetition, and cross-language variability, which makes raw compression-based similarity unreliable. Building on the idea that compression approximates Kolmogorov complexity and has been effective in unsupervised similarity tasks, we design lyric-aware compressors that reduce language entropy or expose musical structure before computing NCD. Our experiments show that these adaptations substantially improve clustering by language, with moderate gains for genre and artist.

The Complexity of Lyric Text

Song lyrics present a form of natural language that is fundamentally different from ordinary prose. While they are written in human languages and therefore follow linguistic regularities, lyrics simultaneously behave as highly stylised artefacts shaped by musical form, rhythmic constraints, and intentional repetition. These characteristics pose unique challenges when applying compression-based similarity methods to lyric corpora.

Our project originates from the observation in **Lab 2** [4] that compression can approximate Kolmogorov complexity [2, 7], enabling unsupervised detection of linguistic similarity and historical relatedness. The same principle has been used in authorship attribution, conceptual similarity estimation [6, 3, 8], and biological sequence phylogeny [5].

Motivated by these findings, we ask:

Does the compressibility of lyrics encode information about their artist, genre, or language, and can we leverage this structure for clustering?

However, unlike typical written documents, lyric text exhibits several layers of complexity that systematically interfere with compression-based similarity. These challenges arise from properties intrinsic to lyrics rather than from algorithmic choices, and they motivate the development of lyric-aware modelling. We highlight three major sources of difficulty:

1. **Multilingual and Cross-Script Variation.** Modern lyric datasets span many languages and writing systems. Differences in morphology, token distribution, and script (e.g., Latin versus CJK) dramatically alter a text's statistical profile. Since compressors operate on byte-level or token-level redundancy,

cross-language variability often dominates more subtle musical or stylistic signals. As a result, compressed sizes may primarily reflect linguistic entropy rather than attributes such as artist or genre.

2. **Structural Repetition and Musical Form.** Lyrics are intentionally repetitive: choruses recur, hooks mirror one another, and verses often follow parallel syntactic and rhythmic templates. Although compressors can detect repeated substrings, they lack awareness of musical structure. Minor formatting inconsistencies with different bracketing, spacing, or annotation conventions, can obscure repetitions that are musically identical. This makes raw compressibility a noisy proxy for the underlying structural patterns shared across songs.
3. **The Limits of Raw NCD for Lyric Similarity.** Normalised Compression Distance assumes that an off-the-shelf compressor can approximate Kolmogorov complexity for the given domain. For lyrics, this assumption breaks down. Language-related entropy differences overwhelm structural signals; musical features such as rhyme and rhythm are invisible to generic compressors; and annotation noise or dataset heterogeneity introduces non-semantic variation. Consequently, NCD computed directly on unprocessed lyrics tends to cluster by language or formatting rather than by artist or genre.

These issues show that lyric text possesses its own form of complexity at the intersection of linguistic diversity, musical structure, and annotation variability, which can be addressed before compression-based similarity can reveal meaningful relationships in songs.

Method: Lyrics-Aware Compressions

We adopt a compression-based similarity pipeline that is adapted specifically for lyric text. The goal is not to change the definition of Normalised Compression Distance (NCD), but to design preprocessing and compressor functions that make the complexity signal more representative of lyrical structure rather than of cross-language entropy or annotation noise.

Dataset. A multilingual lyric dataset [1] containing song texts paired with including *Track_id*, *Song*, *Artist*, *Genre*, *Song year*, and *Language*. This variety provides a realistic test: lyrics differ not only in linguistic form but also in genre conventions and artist-specific stylistic patterns.

Preprocessing. Lyrics cannot be tokenised or normalised in the same way as standard NLP text: lower-casing, punctuation removal, or word-stemming would destroy key signals such as rhyme, line boundaries, chorus markers, or repetition cues. We therefore apply only minimal cleaning: we harmonise structural tags (Intro/Verse/Chorus), remove HTML noise, unify repetition annotations, and drop metadata headers, while strictly preserving line order, casing, and lexical surface form. This produces a cleaned field `Lyrics_clean` that removes noise without disturbing musical structure.

Compressors. All compressors follow a unified API $C(l) \rightarrow \text{bytes}$. We benchmark three families of general-purpose compressors: **i) Baseline family:** `zlib`, `gzip`, dictionary-based LZ77 + Huffman coding; stable classical baselines. **ii) High-ratio compressors:** `lzma`, `brotli`, larger windows and stronger probabilistic modelling; slower but higher compression. **iii) Fast LZ variants:** `lz4`, `snappy`, speed-oriented LZ77 implementations with reduced compression ratio. These provide a spectrum from fast-low-ratio to slow-high-ratio compressors, allowing us to examine how NCD behaves under different modelling strengths.

To adapt compression to lyric-specific structure, we further introduce two custom compressors, *Linguarix* and *Echorix*. Both keep the NCD formula unchanged and only modify $C(l)$.

Linguarix. To reduce cross-language entropy while preserving lexical information, we design *Linguarix*, a linguistic normalisation compressor that transforms each lyric l into a canonical lexical form before applying standard baseline compression.

Given the input byte sequence l , we decode it as UTF-8 and convert alphabetic content to lowercase. Let Σ_{LatinCJK} denote the union of Latin and CJK Unicode blocks. *Linguarix* filters the text by retaining

only characters in $\Sigma_{\text{LatinUCJK}}$, digits, and whitespace, removing all other punctuation via

$$\tilde{l} = \text{Filter}(l) = [c \in l \mid c \in (\Sigma_{\text{LatinUCJK}} \cup \text{digits} \cup \text{space})].$$

The text is then tokenised by whitespace and transformed into a sorted Bag-of-words representation:

$$T(l) = \text{sort}(\text{split}(\tilde{l})).$$

This produces a canonical lexical signature

$$S(l) = t_1 \parallel t_2 \parallel \cdots \parallel t_{|T|},$$

where \parallel denotes concatenation with a separating space.

Finally, baseline compression is applied:

$$C_{\text{Ling}}(l) = \text{zlib}(S(l)).$$

but significantly reduces artefacts from script differences, orthographic conventions, and heterogeneous punctuation across languages. Because the representation collapses all syntactic structure while preserving vocabulary distribution, the compressor emphasises lexical similarity while suppressing noise induced by multilingual orthography.

Echorix. To expose musical structure—particularly rhythm, rhyme, and section-level repetition—to the compressor, we introduce *Echorix*. Unlike standard compressors that operate on raw byte sequences, Echorix constructs a feature-enriched textual representation in which each line is mapped to a fixed-format signature capturing salient lyric properties.

Given a lyric l , Echorix splits it into non-empty lines $L = \{\ell_1, \ell_2, \dots, \ell_m\}$ and computes for each line ℓ a 4-tuple:

$$\Phi(\ell) = \left(\underbrace{|\ell|_{\text{tok}}}_{\text{rhythmic length}}, \underbrace{\text{suffix}_3(\ell)}_{\text{rhyme signature}}, \underbrace{\text{tag}(\ell)}_{\text{section marker}}, \underbrace{\ell}_{\text{raw content}} \right).$$

i) rhythmic length $|\ell|_{\text{tok}}$ is the token count of the line. **ii) rhyme signature** $\text{suffix}_3(\ell)$ extracts the last 2–3 characters of the final word, a coarse proxy for end rhyme. **iii) section marker** $\text{tag}(\ell)$ encodes whether the line begins with a structural label such as [Verse], [Chorus], or [Bridge]; otherwise it is empty. **iv) raw line** ℓ is retained verbatim, allowing the compressor to detect exact-line repetition.

Each line is converted to a single signature string:

$$E(\ell) = \text{“ length | rhyme | tag | } \ell \text{”}.$$

and zlib is finally applied:

$$C_{\text{Echo}}(l) = \text{zlib}[E(\ell_1) \parallel E(\ell_2) \parallel \cdots \parallel E(\ell_m)].$$

captures repetition, chorus structure, and rhyme similarity more effectively than raw compression. Because line structure and rhythmic patterns become explicit in $E(l)$, pairs of songs with similar verse–chorus organisation or stylistic rhyme patterns become more compressible together. This allows compression-based distance to reflect musical-level similarity instead of purely linguistic entropy.

Normalised Compression Distance (NCD). For every compressor C , we define pairwise distances between lyrics x and y using Normalised Compression Distance (NCD) [2]:

$$\text{NCD}(x, y) = \frac{C(xy) - \min(C(x), C(y))}{\max(C(x), C(y))},$$

where xy is a fixed-order concatenation. For a corpus of n songs this yields a symmetric $n \times n$ matrix D with entries $D_{ij} = \text{NCD}(x_i, x_j)$ and $D_{ii} = 0$. Each compressor (baselines, Linguarix, Echorix) therefore induces its own geometry over songs via D .

Clustering in NCD space. We apply clustering algorithms that operate directly on D , without any feature-engineered embeddings. (i) *k-medoids (PAM)* minimises the sum of within-cluster NCD distances and selects actual songs as medoids, avoiding Euclidean assumptions. (ii) *Agglomerative clustering* uses D as a dissimilarity matrix with average linkage, producing a hierarchy from which we cut a flat partition with k clusters. (iii) *Spectral clustering* converts D into an affinity matrix, computes a low-dimensional eigenspace of the graph Laplacian, and runs k -means in that spectral space. All three methods are run separately on the distance matrices from each compressor and for a small range of k .

Evaluation. We treat *Artist*, *Genre*, and *Language* as pseudo-ground-truth labels and compare them with cluster assignments using standard external metrics: adjusted Rand index (ARI), normalised mutual information (NMI), homogeneity, completeness, V-measure, Fowlkes–Mallows, and cluster purity. To account for label permutation we also report Hungarian-matched accuracy. As an intrinsic measure of cluster quality, we compute the silhouette score directly from D .

Results and Conclusions

Comp.*	ARI†			NMI†		
	KM‡	AGG‡	SP‡	KM	AGG	SP
snappy	0.086	0.049	0.001	0.206	0.204	0.036
lzma	0.422	-0.000	0.050	0.512	0.050	0.214
gzip	0.484	0.001	0.050	0.595	0.090	0.212
brotli	0.356	0.049	0.048	0.506	0.223	0.229
lz4	0.616	0.013	0.002	0.678	0.160	0.050
zlib	0.534	0.001	0.050	0.633	0.085	0.220
Linguarix	0.639	0.124	0.051	0.716	0.387	0.223
Echorix	0.447	0.001	0.051	0.560	0.095	0.221

* Comp. = compressor name.

† ARI = Adjusted Rand Index; NMI = Normalised Mutual Information.

‡ KM = k-medoids; AGG = agglomerative; SP = spectral.

Table 1: Language clustering: ARI and NMI scores across compressors and clustering methods.

We evaluate all compressor–cluster combinations on three metadata-derived clustering tasks: grouping by *Language*, *Genre*, and *Artist*. For each compressor we build an NCD matrix over the selected subset of songs and run k -medoids (KM), agglomerative clustering (AGG), and spectral clustering (SP). Tables 1, 2, 3, report Adjusted Rand Index (ARI) and Normalised Mutual Information (NMI) against the corresponding metadata labels.

Overall trends. Across all settings, k -medoids consistently dominates AGG and SP on both ARI and NMI, suggesting that directly optimising within-cluster NCD to true medoids is better matched to the non-Euclidean geometry of compression distances. We therefore focus on KM when comparing compressors, and treat AGG/SP mainly as robustness checks.

Language clustering. Language is the easiest signal for compression-based similarity, and Linguarix substantially strengthens it (Table 1). With KM, the best baseline (gzip) reaches ARI = 0.484 and NMI = 0.595, while Linguarix improves to ARI = 0.639 and NMI = 0.716 (about +0.16 and +0.12 absolute). Echorix, which is less language-normalising, remains competitive with ARI = 0.447 and NMI = 0.560. Figure 1 compares MDS projections of the NCD space across compressors. Compared to generic baselines (zlib, brotli), Linguarix yields much tighter, well-separated language clusters, while Echorix gives an intermediate geometry with clearer but still partially overlapping language regions.

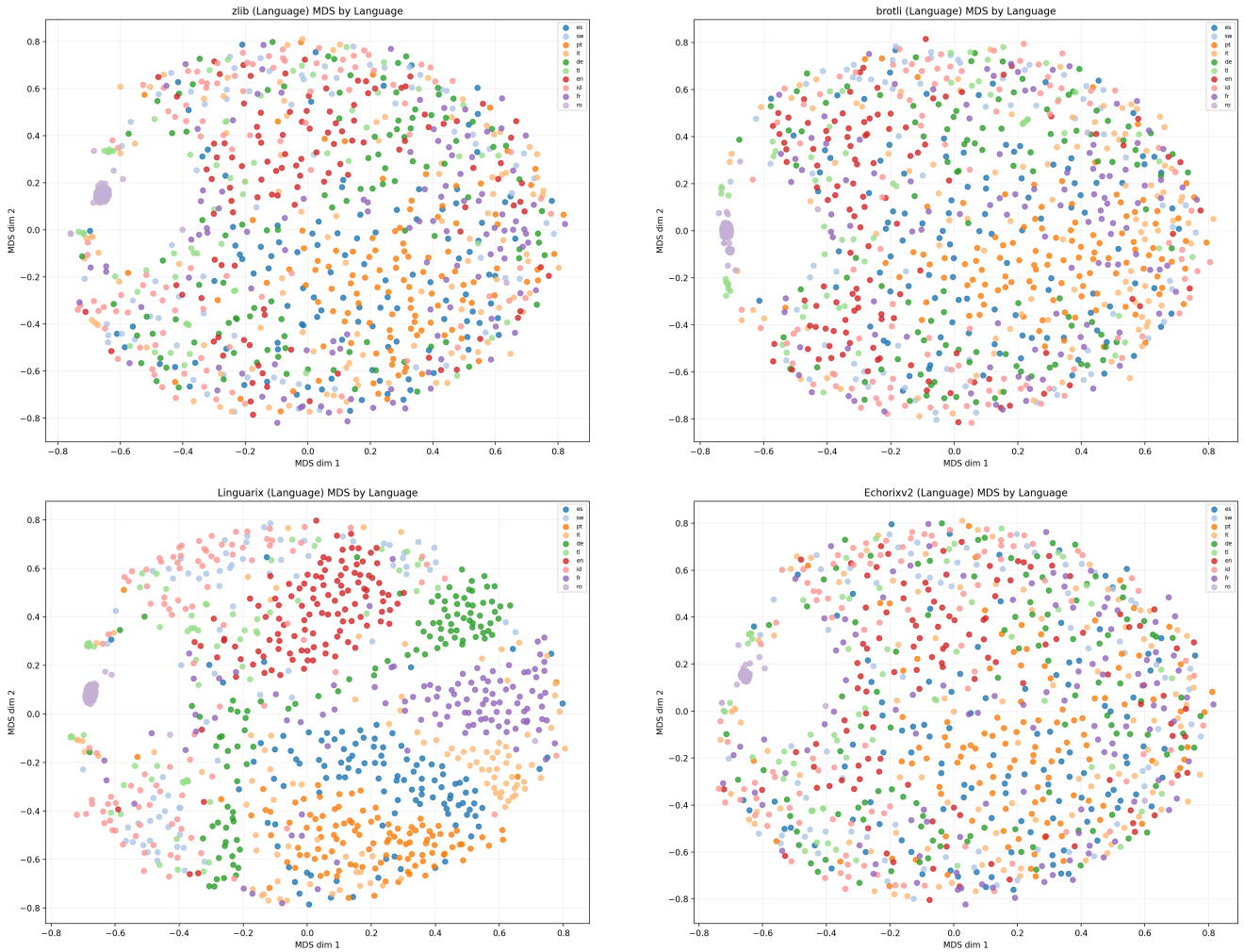


Figure 1: 2D MDS projections of NCD matrices on the *Language* subset for four compressors. Baseline compressors (zlib, brotli) already show some language structure but clusters are fuzzy and heavily intertwined. Linguarix produces compact, well-separated language islands, while Echorix lies in between: it sharpens some language clusters but preserves more cross-language overlap, consistent with its focus on structural rather than purely lexical normalisation.

Genre clustering. Genre is more weakly reflected in raw text, and all methods obtain modest absolute scores (Table 2). Still, the lyric-aware compressors provide consistent gains over baselines. For ARI, Linguarix+KM performs best with 0.062, improving on the best classical compressor (lzma, 0.047). For NMI, Echorix+KM reaches 0.130, slightly outperforming gzip+KM (0.128) and lzma+KM (0.120). Visual inspection of MDS plots (not shown) indicates that genres do not form perfectly separable clusters, but both Linguarix and Echorix produce denser local groupings and clearer boundaries between structurally distinctive genres, compared with the fuzzier structure under plain gzip/lzma NCD.

Artist clustering. Clustering by artist is the hardest setting: many artists have only a handful of songs and style differences can be subtle. As a result, all methods achieve relatively low ARI/NMI (Table 3). The best ARI is obtained by the gzip baseline with KM (0.072), indicating that simple byte-level redundancy already captures some artist-specific signature. Echorix+KM slightly sacrifices ARI (0.054 vs. 0.072) but achieves the highest NMI (0.144 vs. 0.126), suggesting that it forms smaller, purer micro-clusters that better aligns with artist identity even if the global partitioning is more fragmented. Linguarix remains close to the baselines but does not dominate in this regime, which is consistent with its design focus on language- and vocabulary-level normalisation rather than fine-grained stylistic cues.

Comp.*	ARI†			NMI†		
	KM‡	AGG‡	SP‡	KM	AGG	SP
snappy	0.014	0.002	0.056	0.081	0.038	0.102
lzma	0.047	0.007	0.040	0.120	0.084	0.076
gzip	0.038	0.022	0.047	0.100	0.128	0.085
brotli	0.048	0.028	0.006	0.104	0.144	0.048
lz4	0.038	0.018	0.064	0.115	0.116	0.104
zlib	0.038	0.000	0.055	0.099	0.020	0.090
Linguarix	0.062	0.002	0.050	0.112	0.043	0.128
Echorix	0.055	0.001	0.037	0.130	0.049	0.075

* Comp. = compressor name.

† ARI = Adjusted Rand Index; NMI = Normalised Mutual Information.

‡ KM = k-medoids; AGG = agglomerative; SP = spectral.

Table 2: Genre clustering: ARI and NMI scores across compressors and clustering methods.

Comp.*	ARI†			NMI†		
	KM‡	AGG‡	SP‡	KM	AGG	SP
snappy	0.000	0.001	0.000	0.034	0.047	0.036
lzma	0.047	0.009	0.000	0.115	0.086	0.036
gzip	0.072	0.000	0.000	0.126	0.051	0.034
brotli	0.048	0.000	0.000	0.115	0.046	0.034
lz4	0.046	0.015	0.001	0.112	0.104	0.048
zlib	0.077	0.001	0.000	0.137	0.055	0.042
Linguarix	0.043	0.001	-0.000	0.113	0.041	0.038
Echorix	0.054	0.001	0.000	0.144	0.050	0.048

* Comp. = compressor name.

† ARI = Adjusted Rand Index; NMI = Normalised Mutual Information.

‡ KM = k-medoids; AGG = agglomerative; SP = spectral.

Table 3: Artist clustering: ARI and NMI scores across compressors and clustering methods.

Summary. Taken together, the experiments show that operating directly in NCD space is competitive with, and often strengthened by, lyric-aware compression. Linguarix significantly improves recovery of language structure and gives moderate gains for genre, while Echorix contributes complementary improvements for genre and artist clustering by exposing rhythmic and sectional repetition to the compressor.

Bibliography

- [1] Matei Bejan. *Multilingual Lyrics for Genre Classification*. <https://www.kaggle.com/datasets/mateibejan/multilingual-lyrics-for-genre-classification>. Kaggle dataset, accessed 2025-11-25.
- [2] Rudi Cilibrasi and Paul MB Vitányi. “Clustering by compression”. In: *IEEE Transactions on Information theory* 51.4 (2005), pp. 1523–1545.
- [3] Rudi L Cilibrasi and Paul MB Vitányi. “The google similarity distance”. In: *IEEE Transactions on knowledge and data engineering* 19.3 (2007), pp. 370–383.
- [4] Jean-Louis Dessalles. *Lab 2: Compression and Kolmogorov Complexity*. <https://aicourse.r2.enst.fr/FCI/Chapitre2.html>. Accessed: 2025-11-25.

- [5] Paolo Ferragina et al. “Compression-based classification of biological sequences and structures via the universal similarity metric: experimental assessment”. In: *BMC bioinformatics* 8.1 (2007), p. 252.
- [6] Ming Li et al. “The similarity metric”. In: *IEEE transactions on Information Theory* 50.12 (2004), pp. 3250–3264.
- [7] Paul MB Vit et al. “Compression-based similarity”. In: *2011 First International Conference on Data Compression, Communications and Processing*. IEEE. 2011, pp. 111–118.
- [8] Paul MB Vitányi et al. “Normalized information distance”. In: *Information theory and statistical learning*. Springer, 2009, pp. 45–82.

Appendix

Echorix.py

```

1 def Echorix(data: bytes) -> bytes:
2     text = data.decode("utf-8", errors="ignore")
3     # Split lines
4     lines = [ln.strip() for ln in text.splitlines() if ln.strip()]
5     processed_lines = []
6     for ln in lines:
7         # Rhythm: line length
8         length = len(ln.split())
9         # Rhyme: last 2-3 letters
10        tokens = ln.lower().split()
11        rhyme = ""
12        if tokens:
13            last = tokens[-1]
14            rhyme = last[-3:] # crude rhyme signature
15        # Capture uppercase section tags (Verse/Chorus/Bridge)
16        if ln.lower().startswith("[verse") or ln.lower().startswith("[chorus") or ln.lower().
17           ↳ startswith("[bridge"):
18            tag = ln.lower()
19        else:
20            tag = ""
21        # Repetition signature: full line index
22        processed_lines.append(f"{length}|{rhyme}|{tag}|{ln}")
23    final_text = "\n".join(processed_lines)
24    return zlib.compress(final_text.encode("utf-8"))

```

Linguarix.py

```

1 def Linguarix(data: bytes) -> bytes:
2     # decode bytes -> text and normalise
3     text = data.decode("utf-8", errors="ignore").lower()
4     # keep letters (Latin + CJK), digits, whitespace; drop most punctuation
5     text = re.sub(r"[^0-9A-Za-z\u00C0-\u024F\u4E00-\u9FFF\s]", "_", text)
6     tokens = text.split()
7     tokens.sort()
8     processed = "_".join(tokens)
9     return zlib.compress(processed.encode("utf-8"))

```




November, 2025

CSC 5AI25 TP

Algorithmic Information & Artificial Intelligence

Micro-study

teaching.dessalles.fr/FCI

Name: DJANBAZ Mostafa

French author classifier

Abstract

This study addresses the problem of authorship attribution for French literary texts using algorithmic information theory. A classifier is implemented based on Normalized Compression Distance (NCD) as a practical approximation of Kolmogorov complexity, combined with logistic regression. The approach achieves superior performance compared to a baseline nearest-neighbor method, demonstrating that compression-based features can effectively capture author-specific stylistic patterns.

Problem

Authorship attribution is the task of identifying the author of a given text based on stylistic characteristics. Traditional approaches rely on hand-crafted linguistic features (word frequencies, syntactic patterns, etc.), which require domain expertise and may not generalize well across languages or genres.

From an information-theoretic perspective, we can use Kolmogorov complexity to distinguish between authors. Since Kolmogorov complexity is uncomputable, we approximate it using real-world compression algorithms (gzip, bz2, lzma) through the Normalized Compression Distance (NCD), which measures similarity between texts based on their joint compressibility.

Method

After collecting and cleaning French literary texts from Project Gutenberg, I created snippets (5-10 sentences) as test samples: 100 per author and excerpts (10-30 sentences) as

reference corpus: 50 per author. The data split is: 80% training, 20% testing (stratified by author).

The NCD was implemented using compression methods (Gzip, bz2 and lzma available).

For each snippet, NCD against all 50 excerpts from each author was computed, then we have: minimum, mean, median, standard deviation of NCD values. This yielded a feature vector of dimension: 60 dimensions.

The baseline classifier assigns each test snippet to the author with minimum mean NCD (gzip only).

The Logistic Regression trains a multinomial classifier on the full feature matrix.

For faster experimentation and tests, a cache was used to avoid recomputing (compression sizes, features...).

Results

Baseline Performance:

Accuracy: 50%

Simple, interpretable, but limited

Some authors were highly misclassified.




Logistic Regression Performance

Accuracy: 94% (huge improvement)

Discussion

Using multiple compression algorithms (gzip, bz2, lzma) captures complementary patterns

Statistical aggregations (min, mean, median, std) encode author-specific similarity distributions. The 60-dimensional feature vector seems to capture well the authors' style.

  	November, 2025
	CSC-5AI25-TP
	<h1>Algorithmic Information & Artificial Intelligence</h1>
	<p>Micro-study</p> <p>teaching.dessalles.fr/FCI</p>

Name: Ivana Nassar

Model Selection with Minimum Description Length (MDL) and Kolmogorov Complexity

Table of Contents

1.	Abstract.....	2
2.	Method	2
2.1	Overview of the Experiments Conducted.....	2
2.2	MDL and Kolmogorov Complexity.....	2
2.3	Why We Use the BIC-like MDL Score	3
3.	Results	3
3.1	Polynomial Regression Experiments.....	3
3.2	CNN MDL Experiment on Fashion-MNIST.....	8
4.	Discussion	8
4.1	Conclusion	9
4.2	Limitations and Future Work.....	9
5.	Bibliography.....	9

1. Abstract

This report presents a full study of the Minimum Description Length (MDL) principle as an operational approximation of Kolmogorov Complexity (K). The goal is to use MDL for model selection across polynomial regression models and convolutional neural networks, and to demonstrate how MDL embodies the theoretical ideas of Algorithmic Information Theory.

2. Method

2.1 Overview of the Experiments Conducted

To evaluate MDL in different settings, we performed a series of structured experiments:

Experiment 1 — Linear Data:

A true linear function $y = 2x + 1 + \text{noise}$ was generated with fixed noise $\sigma = 1.0$ and sample size $n = 50$. We fitted polynomial models of degrees 1, 2, and 3 to observe which model MDL selects.

Experiment 2 — Quadratic Data:

A true quadratic function $y = x^2 - x + 0.5 + \text{noise}$ (same noise and sample size) was used, again fitting degrees 1, 2, and 3. We expected MDL to prefer degree 2.

Extended Analysis:

We then revisited Experiment 1 by:

- Varying noise levels ($0.1 \rightarrow 5.0$) to test robustness.
- Increasing sample sizes ($10 \rightarrow 200$) to see how MDL behaves with more information.

Residual Kolmogorov Complexity:

For the linear experiment, we computed the compressibility of residuals for each polynomial model using zlib compression. According to AIT, the correct model should leave only incompressible noise.

CNN Experiment:

Finally, we extended MDL to deep learning by training three CNN architectures of different sizes on Fashion-MNIST. This allowed us to test whether MDL still favors the simplest adequate architecture.

2.2 MDL and Kolmogorov Complexity

Kolmogorov Complexity defines the shortest program that can generate an object. MDL provides a computable surrogate:

$$\text{MDL}(M) = K(\text{model}) + K(\text{data} \mid \text{model})$$

In practice, for regression with Gaussian noise, this yields a BIC-like score:

$$\text{MDL} = n * \log(\text{SSE}/n) + k * \log(n)$$

Where n is the number of samples, SSE the residual error, and k the number of parameters.

2.3 Why We Use the BIC-like MDL Score

The BIC-like form arises from coding theory. Encoding the residuals under Gaussian noise yields the $n \log(\text{SSE}/n)$ term. Encoding the model parameters requires specifying k real numbers to a precision determined by n data points, producing the $k \log(n)$ complexity penalty. This ensures models with excessive parameters are penalized according to the resolution needed to describe them.

3. Results

3.1 Polynomial Regression Experiments

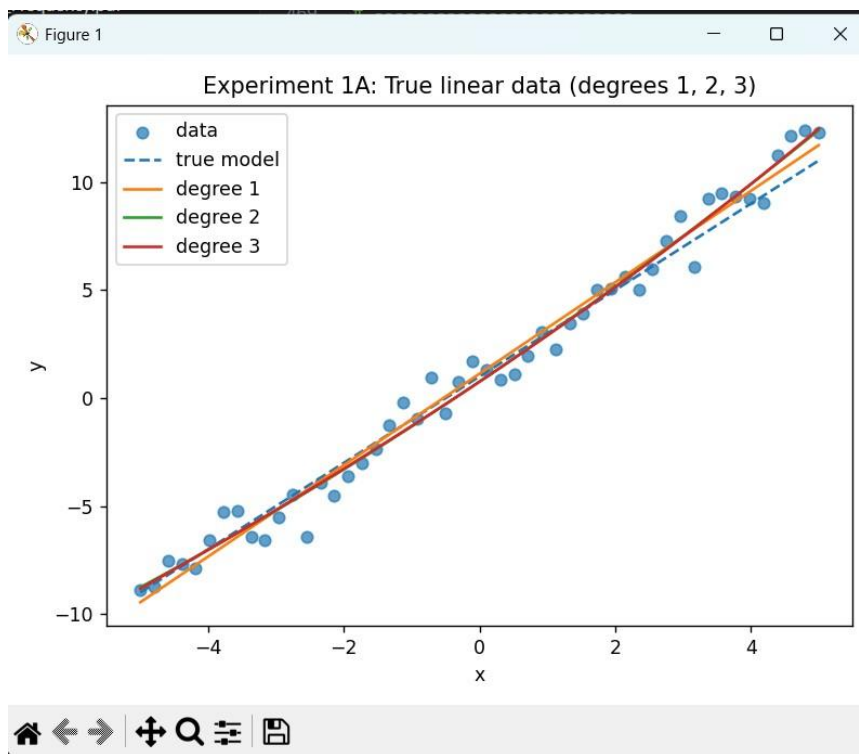


Figure 1: True linear data with polynomial fits of degree 1, 2, and 3.

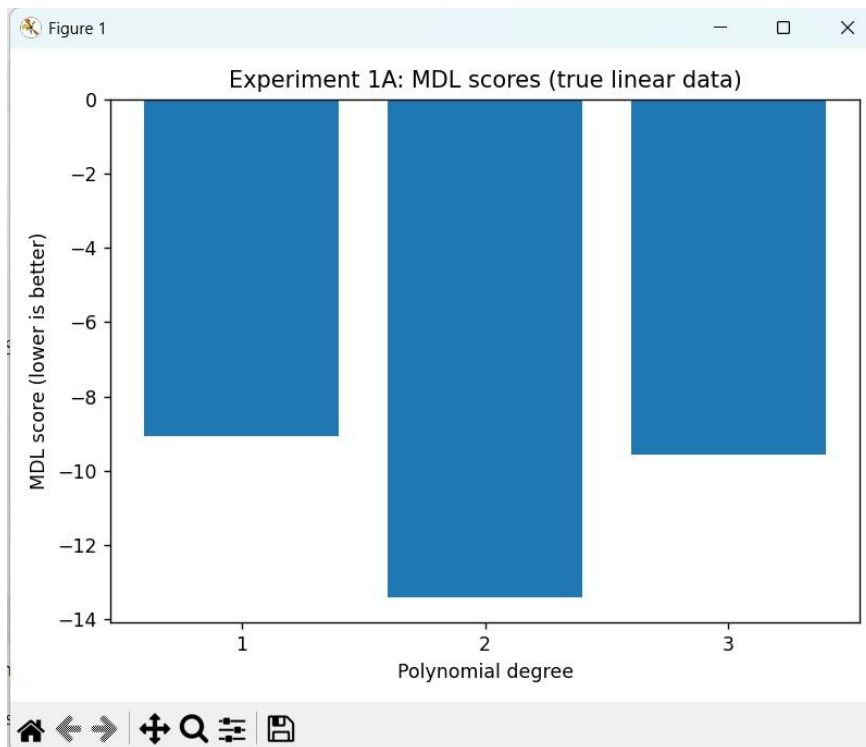


Figure 2: MDL scores for degrees 1, 2, 3 on true linear data.

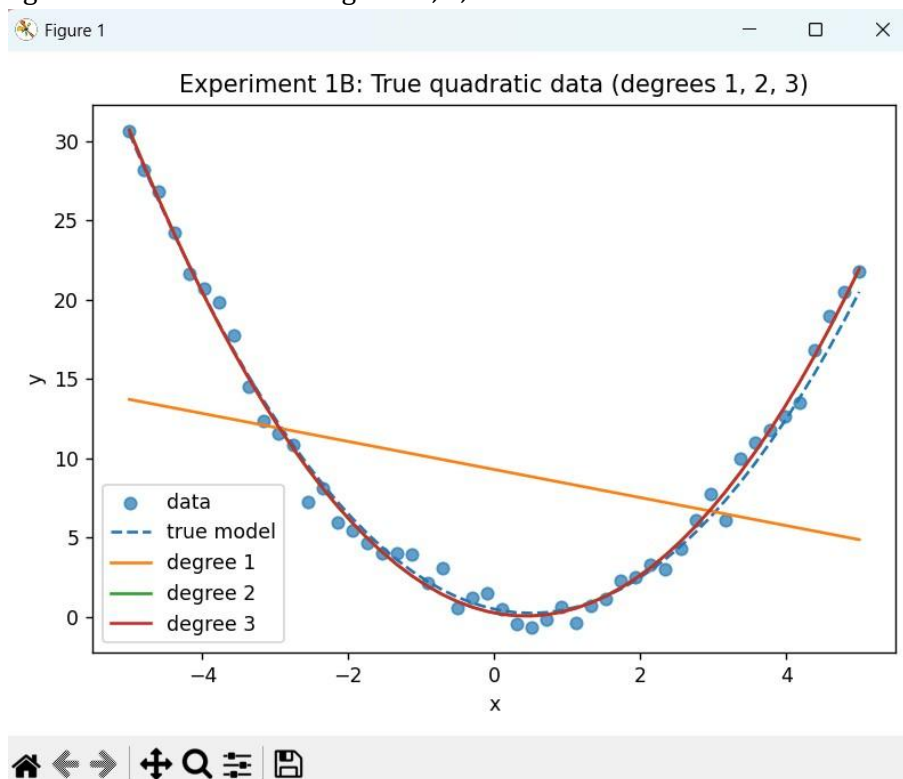


Figure 3: True quadratic data with degrees 1, 2, and 3 fits.

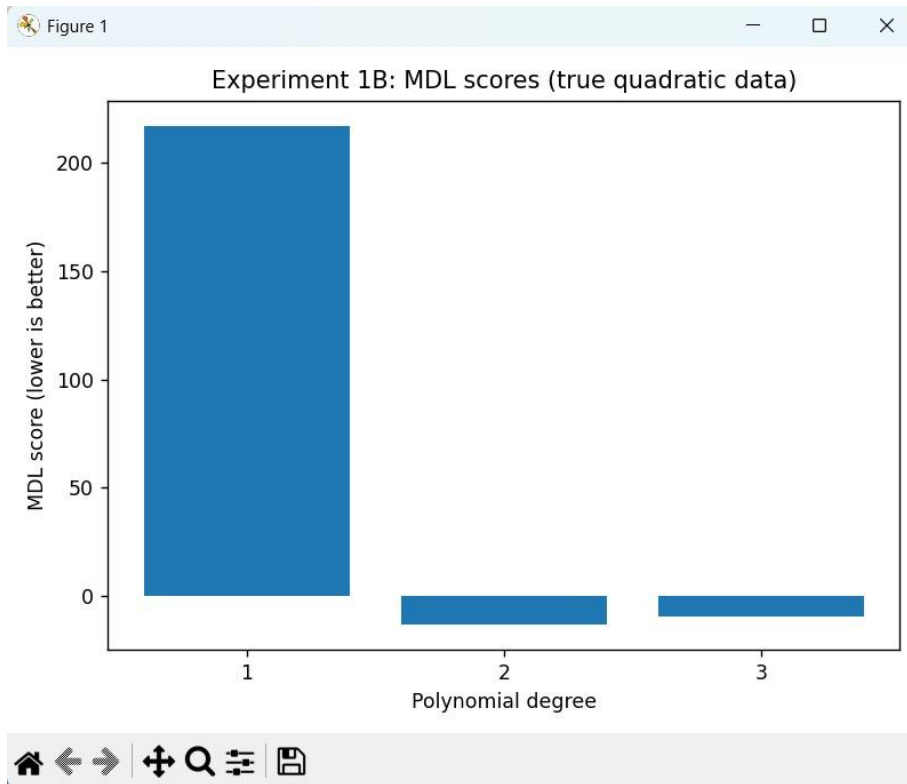


Figure 4: MDL scores for quadratic experiment, selecting degree 2.

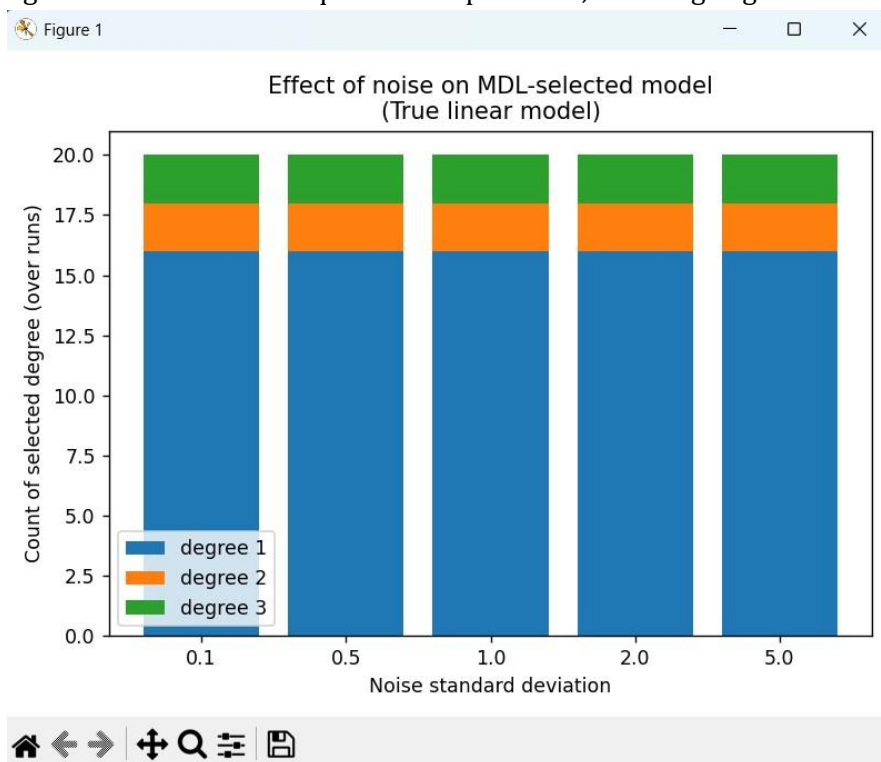


Figure 5: Noise effect on MDL-selected model.

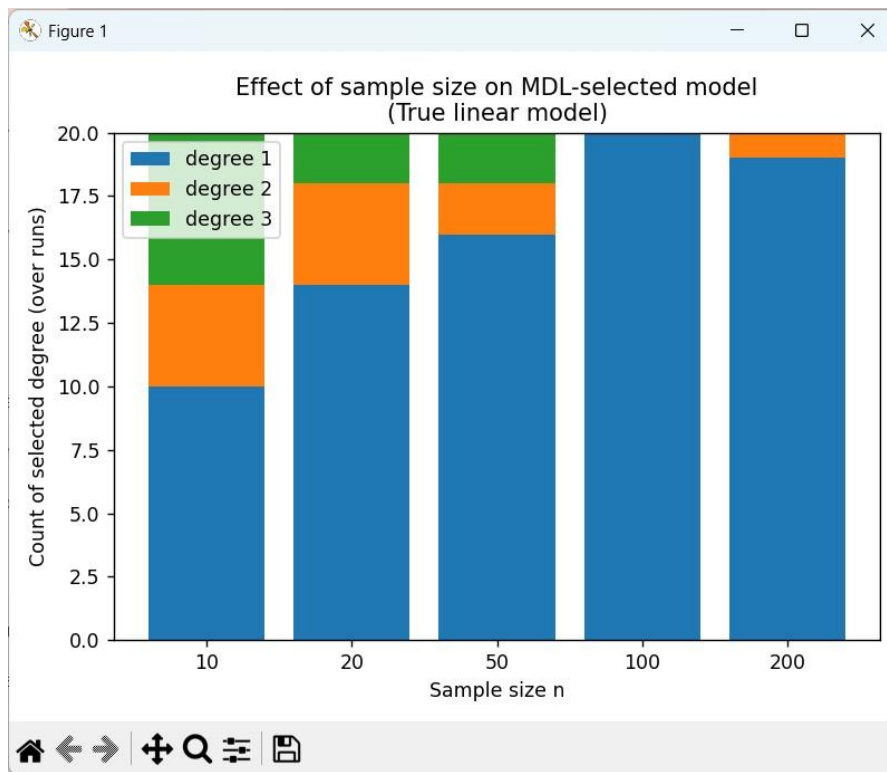


Figure 6: Sample size effect on MDL model selection.

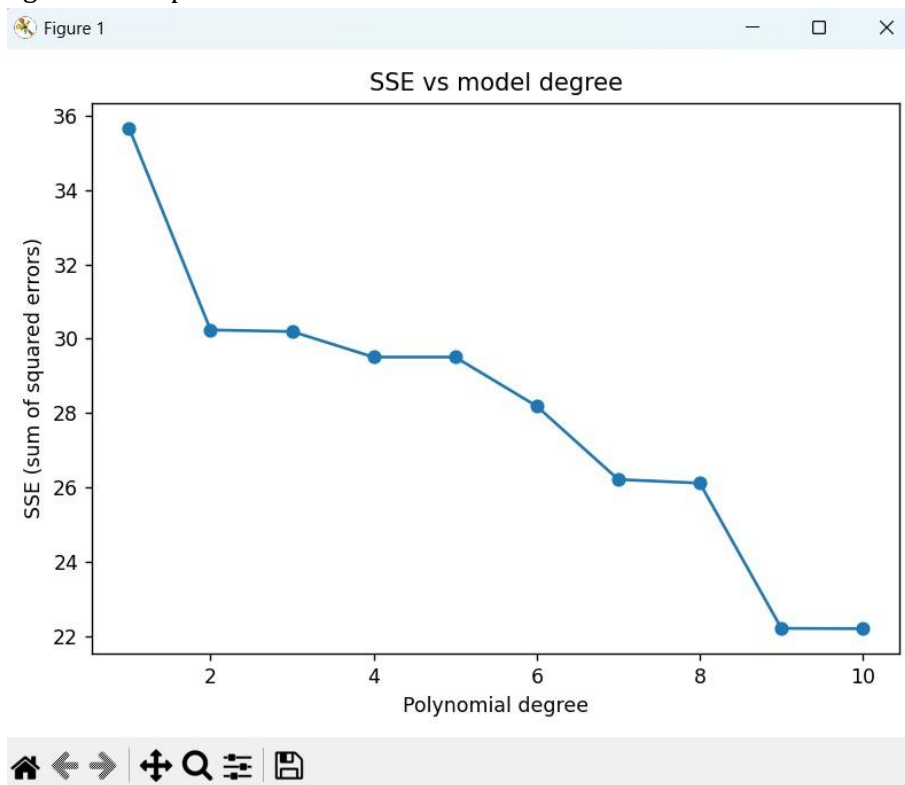


Figure 7: SSE vs model degree showing monotonic decrease.

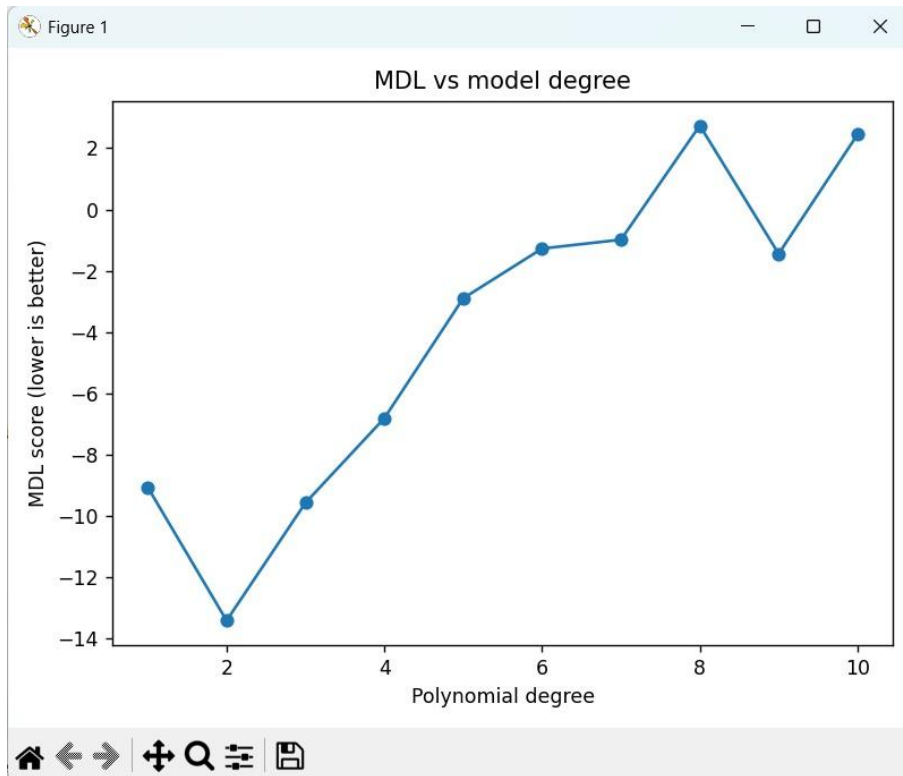


Figure 8: MDL vs degree showing optimal penalty at small degree.

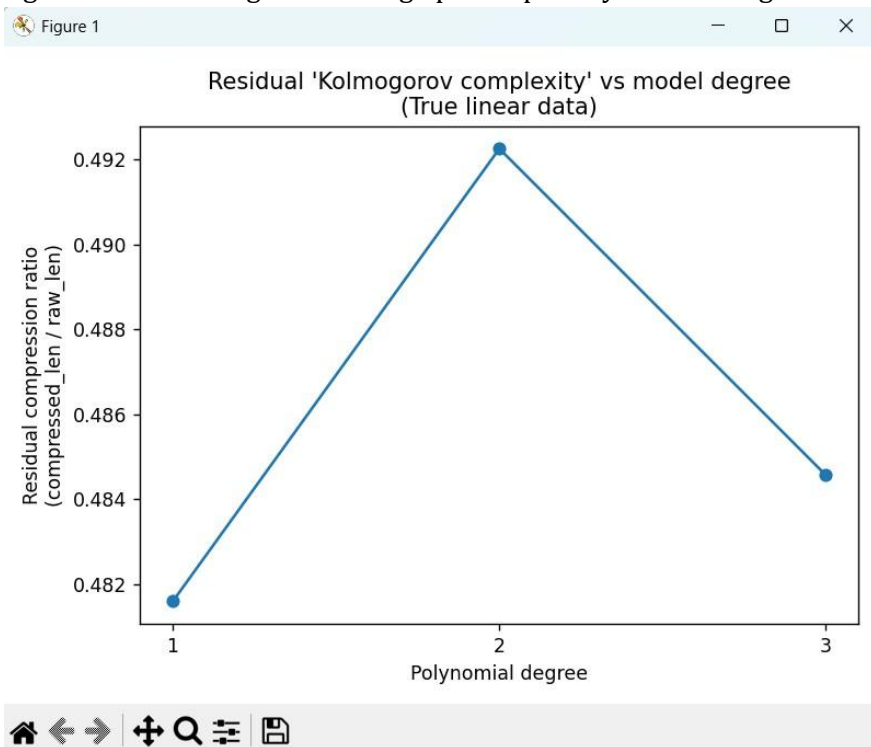


Figure 9: Residual Kolmogorov complexity estimated via compression.

The correct model (degree 2) leaves only pure noise in the residuals.

Noise is algorithmically random and therefore maximally incompressible. As a result, the compression ratio of residuals is highest for the degree-2 model, which is fully consistent with the Kolmogorov complexity view and supports the MDL decision.

3.2 CNN MDL Experiment on Fashion-MNIST

We extend MDL scoring to convolutional neural networks trained on Fashion-MNIST. Three architectures of increasing complexity were considered. The small CNN achieves the best MDL score despite the large CNN having the highest accuracy, illustrating MDL's ability to avoid overfitting by penalizing unnecessary parameters.

```
=== Training small model ===
Small model:
  Validation CE (loss) = 0.3296
  Validation accuracy  = 0.8870
  #parameters (k)     = 27210
  MDL score           = 253908.99

=== Training medium model ===
Medium model:
  Validation CE (loss) = 0.3169
  Validation accuracy  = 0.8806
  #parameters (k)     = 121930
  MDL score           = 1126185.89

=== Training large model ===
Large model:
  Validation CE (loss) = 0.2364
  Validation accuracy  = 0.9092
  #parameters (k)     = 197482
  MDL score           = 1821240.64

=== MDL comparison (Fashion-MNIST) ===
small -> MDL = 253908.99, CE = 0.3296, k = 27210
medium -> MDL = 1126185.89, CE = 0.3169, k = 121930
large -> MDL = 1821240.64, CE = 0.2364, k = 197482
Best model according to MDL: small (MDL = 253908.99)
```

Figure 10: CNN MDL comparison on Fashion-MNIST.

4. Discussion

Across all experiments, MDL consistently selected the simplest model capable of capturing the true structure of the data. In the linear dataset, MDL favored degree-1 and occasionally degree-2 models when noise fluctuations slightly improved the residual fit. In the quadratic dataset, MDL reliably selected degree-2, strongly penalizing the underfitting degree-1 model and the unnecessarily flexible degree-3 model.

Noise and sample-size experiments confirmed this behavior: higher noise levels made MDL more conservative and biased toward simpler models, while increasing sample size strengthened MDL's preference for the true model by increasing the precision penalty k

$\log(n)$. Residual-complexity analysis showed that the correct model leaves residuals that are the most incompressible, aligning with Kolmogorov's notion that true noise should be algorithmically random.

Finally, in the CNN experiments, although the largest network achieved the best accuracy, MDL selected the smallest CNN because its improvement in cross-entropy was not sufficient to compensate for the dramatic increase in parameter complexity. This demonstrates that MDL remains effective even for deep neural architectures and aligns with the theoretical expectation: models should only grow in complexity when doing so significantly reduces the description length of the data.

4.1 Conclusion

This project demonstrates that MDL provides a principled, practical, and theoretically grounded method for selecting models of appropriate complexity. Across linear, quadratic, and CNN experiments, MDL consistently favored the simplest model that captured the essential structure of the data. By linking residual randomness to algorithmic complexity, we strengthened the connection between MDL and Kolmogorov's theory. These results confirm MDL's relevance for both classical statistical models and modern deep learning architectures.

4.2 Limitations and Future Work

This project has several limitations. First, the synthetic experiments rely on simple polynomial models, which may not capture real-world irregularities. Second, the compression-based residual complexity depends on quantization choices and specific compressors such as zlib, which do not precisely approximate Kolmogorov Complexity. Finally, the CNN MDL evaluation uses only three architectures; broader neural model families could strengthen the conclusions.

Future work may include:

- Integrating more expressive models such as transformers and variational autoencoders.
- Using normalized compression distance (NCD) for deeper AIT insights. - Extending MDL scoring to multi-modal or sequential data.

5. Bibliography

- [1] CSC_5A125_TP: [Algorithmic Information and A.I.](#)
- [2] Grünwald, P. (2007). The Minimum Description Length Principle. MIT Press. [3] Li, M., & Vitányi, P. (1997). An Introduction to Kolmogorov Complexity and Its Applications.
- [4] Rissanen, J. (1978). Modeling by shortest data description.
- [5] Hutter, M. (2005). Universal Artificial Intelligence. Springer.

Python Code Reference :

https://drive.google.com/file/d/1LlzodFLd9bCB8rIgXyulMvkYsAsXGm3b/view?usp=drive_link



November, 2025

CSC-5AI25-TP

Algorithmic Information & Artificial Intelligence

Micro-study

teaching.dessalles.fr/FCI

Name: AHMED BOUHA Mohamed

How many legal chess positions exist? A compression based technique

Abstract

Chess is a game of vast complexity and possibilities, yet the number of possible positions is finite, and therefore theoretically enumerable. This project tries to find an upper limit on that number. This report assumes the reader knows chess rules, see Bibliography.

Problem

Two chess positions are considered *distinct* if and only if they differ in their set of legal next moves. Under this definition, differences in castling rights, en passant rights, or move counters necessarily generate distinct positions even when the board configuration appears identical.

The classical counting problem asks: **how many such distinct legal positions exist?** From AIT, for any finite set S , every element

$x \in S$ satisfies:

$$K(x) \leq \log_2 |S| + O(1).$$

Thus, determining an upper bound on $|S|$ is equivalent to determining an upper bound on the Kolmogorov complexity of any chess position.

A direct combinatorial enumeration is tempting, but flawed: such estimates simultaneously (1) massively overcount by including illegal or physically impossible configurations (e.g. no king, too many pieces) while also (2) *undercounting* by failing to distinguish positions identical in appearance but different in move rights. Because of this double error, such combinatorial formulas cannot be guaranteed to give a valid upper bound.

To obtain a mathematically sound upper bound, we instead construct an *explicit lossless encoding* capable of representing every legal chess position uniquely. The length of this encoding immediately yields a guaranteed upper bound on both the number of positions and their Kolmogorov complexity.

Method

We begin with a naive combinatorial estimate to illustrate the difficulty of rigorously bounding the number of legal chess positions.

Each of the 64 squares can be either empty or occupied by one of 12 standard piece types:

- White: King, Queen, Rook, Bishop, Knight, Pawn
- Black: King, Queen, Rook, Bishop, Knight, Pawn

Including the empty square as a "piece type" gives a total of 13 possible states per square. Hence, the total number of possible board configurations is initially estimated as:

$$13^{64}$$

However, this number is far too large and cannot serve as a reliable upper bound on legal positions, because:

- It counts many illegal configurations, such as those missing kings, impossible pawn structures, or multiple promoted pieces.
- It does not distinguish positions that are identical in piece placement but differ in game-theoretic state. For example, a rook on its starting square may or may not have castling rights depending on prior moves. Two such positions have identical pieces on the board but differ because one rook can castle and the other cannot, thus the set of possible next moves is different.
- Similarly, en passant rights affect the uniqueness of positions even if the board looks the same visually.

To account for these subtleties, we extend the piece set to include new variants reflecting castling and en passant states:

- Two new rook types: *rook can castle* and *rook cannot castle*, for both white and black.
- Two new pawn types: *pawn can be taken en passant*, for both white and black.

This results in a total of 16 piece types + 1 empty square type, increasing the naive upper bound to:

$$17^{64}$$

which is even larger but better captures the distinctness of positions arising from special rules.

We can refine the counting by considering exactly n pieces on the board, where $n \leq 32$ (the maximum number of pieces at the start of the game). Assuming these n pieces are chosen from the 12 standard piece types (ignoring the special states for simplicity), the number of ways to choose which squares are occupied is:

$$\binom{64}{n}$$

We consider positions with exactly n pieces placed on 64 squares:

$$\binom{64}{n} \cdot 12^n,$$

where 12 refers to the standard piece types (6 per color). Summing over all $n \leq 32$ gives:

$$\sum_{n=0}^{32} \binom{64}{n} 12^n.$$

However, this number *cannot* be claimed as an upper bound:

- It counts many illegal configurations (missing kings, impossible pawn structures, multiple promoted pieces, etc.).
- It misses many *legal* positions because it does not encode castling rights, en passant rights, or move counters.
- It therefore does not define a unique mapping between physical board states and distinct game-theoretic positions.

Because of this, such combinatorial reasoning cannot be used to bound $|\mathcal{P}|$, the number of legal chess positions, nor to bound Kolmogorov complexity.

A constructive approach

To obtain a true upper bound, we construct a lossless encoding capable of representing *every* legal chess position uniquely. Such a mapping ensures that if a string of b bits suffices to describe any legal position, then:

$$|\mathcal{P}| \leq 2^b \quad \text{and} \quad K(P) \leq b + O(1).$$

Encoding Scheme

Our compression scheme draws inspiration from the concept of bitboards, a common chess programming technique where a 64-bit integer represents the presence or absence of pieces on each square of the chessboard. Each bit corresponds to a square, with 1 indicating occupancy and 0 indicating emptiness. In bitboards we need 12 64-bit integers to represent a position (1 64-bit integer for each piece type) + a couple bits for special info like castling rights and en passant.

Building on this idea, we improve efficiency by explicitly separating the information about where pieces are from what those pieces are. First, we create a 64-bit integer bitboard indicating which squares are occupied (1 for occupied, 0 for empty). Then, for each occupied square, we encode the piece type.

There are 12 distinct piece categories: king, queen, rook, bishop, knight, and pawn, each for black and white. We can encode these 12 types using 4 bits, which allows for up to 16 different values. Since a maximum of 32 pieces can be on the board, we allocate 4 bits per piece, totaling $4 \times 32 = 128$ bits to describe all pieces. Along with the initial 64-bit occupancy bitboard, this results in $64 + 128 = 192$ bits, or three 64-bit integers, to represent the entire position.

However, this initial scheme does not account for special move rights like castling or en passant. To address this, we use the extra 4 unused states available in the 4-bit piece encoding (since 16 possible values minus 12 used = 4 spare states).

For castling rights, we introduce two new piece variants: "rook can castle" and "rook cannot castle," for both black and white, adding 2 new piece types. This increases the total to 14 piece types. This approach effectively encodes castling rights by tracking rook states, which also implicitly captures whether the king has moved (since if the king moves, both rooks lose castling rights).

For en passant, we add two more variants: "pawn can be taken en passant," one for black and one for white. This brings the total to 16 piece types, perfectly fitting the 4-bit encoding.

With these adjustments, our scheme uses exactly three 64-bit integers (192 bits) to encode every detail of a legal chess position, including piece placement, castling rights, and en passant possibilities, with zero loss.

Results

Since every legal chess position maps to a unique 193-bit string:

$$|\mathcal{P}| \leq 2^{192} \approx 6.2771 \times 10^{57}.$$

By AIT, this immediately implies:

$$\forall P \in \mathcal{P}, \quad K(P) \leq 193 + O(1).$$

This is a rigorous, constructive, and verifiable upper bound. It does not rely on probabilistic or combinatorial approximations but simply uses insight from AIT about complexity and compression.

While we have come far from our initial rigorous upper bound of 17^{64} , modern upper bounds like the one from the chess position ranking project have us beat by a lot with an upper bound estimate of around 10^{46} , so our compression method still has some redundant information.

Discussion

Though 192 bits is likely far from the theoretical minimum, several avenues for improvement and alternative approaches were considered. One promising idea is to treat the starting position of chess as a fixed reference point and encode any given position as a sequence of moves from this start. Since each move can be represented compactly by specifying start and end squares along with updates to special state information, this approach can drastically reduce the bits needed for positions close to the opening. However, this move-based compression is less effective for arbitrary or deep positions far from the initial state because the number of moves needed to reach them may be large, causing the encoding size to grow significantly. An extension of this concept involves computing the shortest sequence of moves—legal or even illegal—that reaches the target position, potentially optimizing the move count and thus the encoding length. Yet this introduces challenges, particularly in dealing with ambiguities in castling and en passant rights when sequences include illegal moves. Additional bits would be necessary to resolve these ambiguities, complicating the encoding. Moreover, devising an algorithm to find minimal-length sequences (or minimal transpositions) from the starting position to any arbitrary position is non-trivial and remains an open problem. Proving bounds on the maximum number of moves required to reach any legal position would be essential to justify the efficiency and correctness of such an approach.

Bibliography

- Chess Rules:
<https://www.chess.com/learn-how-to-play-chess>
- Bitboards: <https://www.chessprogramming.org/Bitboards>
- The Chess Position Ranking Project:
<https://github.com/tromp/ChessPositionRanking>



November, 2025

CSC-5AI25-TP

Algorithmic Information & Artificial Intelligence

Micro-study

teaching.dessalles.fr/FCI

Name: Marcin Porwisz

MDL as an Optimization Criterion in Metaheuristics

Abstract

This work compares different model selection criteria in regression problems using a genetic algorithm as the optimization engine. Several statistical models and parametrizations are explored, and their complexity is evaluated using MDL-based criteria as well as a BIC-like baseline. Experiments on both synthetic and real-world data demonstrate how different criteria behave under varying levels of noise and dimensionality. The full implementation is available on GitHub [1].

Problem

There exist many different statistical models for regression. Given a particular task, some models may be too simple while others may overfit the data, harming generalization. There is no universal method for

determining which model offers the best trade-off between expressive power and model simplicity.

Method

I implemented a genetic algorithm that searches for a model that best approximates the data by minimizing a given selection criterion. The search space includes several statistical model families together with their corresponding hyperparameters (see Table 1).

Model type	Parameters / ranges
linear	no hyperparameters (ordinary least squares)
polynomial	polynomial degree: $2 \leq d \leq 8$
spline	number of knots: $3 \leq n_{\text{knots}} \leq 10$
tree	max depth: $1 \leq d_{\text{max}} \leq 6$
ridge	regularization: $10^{-3} \leq \alpha \leq 10^3$
random-forest	estimators: $10 \leq T \leq 100$, max depth: $1 \leq d_{\text{max}} \leq 6$
mlp	hidden size: $5 \leq h \leq 50$, regularization: $10^{-3} \leq \alpha \leq 10^3$

Table 1: Model types and hyperparameter ranges used in the genetic algorithm.

MDL definition

We begin with the following smooth approximation of the absolute value:

$$\text{abs_app}(x) = \sqrt{x^2 + \varepsilon}, \quad \varepsilon > 0$$

Using this, we define the scalar coding cost

$$c(x) = \log_2(1 + \text{abs_app}(x)) \approx \log_2(1 + |x|)$$

For a vector or matrix $M = (M_i)$, the total coding cost is

$$C_{\text{matrix}}(M) = \sum_i \log_2(1 + \text{abs_app}(M_i)) \approx \sum_i \log_2(1 + |M_i|)$$

Regression setting

We denote the model parameter vector and its length by:

$$\theta \in \mathbb{R}^k, \quad k = \dim(\theta)$$

For each data point x_i , the model prediction is:

$$\hat{y}_i = f_\theta(x_i)$$

and the residuals are

$$r_i = y_i - \hat{y}_i$$

The MDL model cost is defined as:

$$C_{\text{model}}(\theta) = C_{\text{matrix}}(\theta) + \log_2 k$$

The data cost (coding the residuals) is:

$$C_{\text{data}}(y \mid \theta) = C_{\text{matrix}}(r) = \sum_{i=1}^n \log_2(1 + \text{abs_app}(r_i))$$

Thus, the full MDL objective becomes:

$$\boxed{\text{MDL}_C(\theta) = C_{\text{model}}(\theta) + C_{\text{data}}(y \mid \theta) = C_{\text{matrix}}(\theta) + \log_2 k + C_{\text{matrix}}(r)}$$

Next, I implemented a modified MDL variant that introduces a residual threshold and assigns a minimum penalty of one bit to each significant error:

$$C_{\text{res}}(r) = \sum_{i=1}^n c(r_i), \quad c(r_i) = \begin{cases} 0, & |r_i| \leq \tau, \\ 1 + \log_2(1 + |r_i|), & |r_i| > \tau. \end{cases}$$

Similarly, each non-negligible parameter receives an additional bit:

$$C_{\text{par}}(\theta) = \sum_{i=1}^k (1 + \log_2(1 + |\theta_i|))$$

Note: The implemented MDL considers the complexity of parameters and error residuals only. Structural model complexity is not included. Furthermore, the coding length is not restricted to integers; fractional bits are allowed.

Baseline criterion

For comparison, we include a BIC-like criterion based on MSE:

$$\text{MSE}(\theta) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

$$\text{MDL}_{\text{BIC}}(\theta) = k \log n + n \log(\text{MSE}(\theta))$$

Because the criteria are based on different assumptions, only the resulting models can be meaningfully compared, not the absolute metric values.

Results

Experiment 1: Synthetic 1D Data

In this experiment we evaluated model selection on a simple one-dimensional synthetic dataset. All tested MDL/BIC criteria produced the same optimal model.

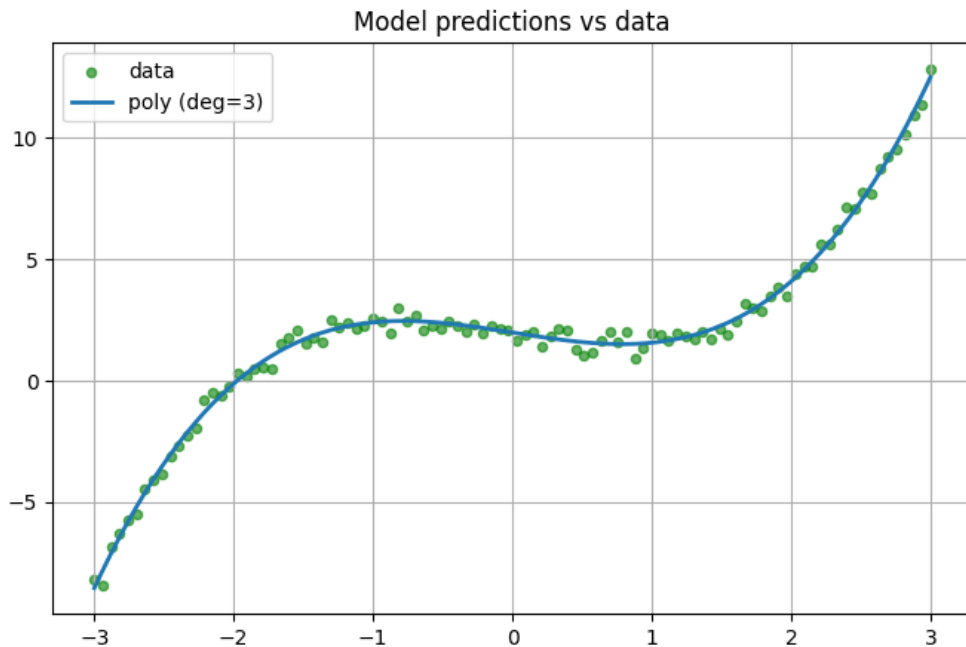


Figure 1: Results of Experiment 1 on synthetic 1D data. All criteria selected the same model.

Experiment 2: Marketing real 1D Data

In this experiment we tested the criteria on the marketing-campaign dataset from Kaggle [2]. This time MDL and BIC yielded different models: both MDL versions selected a polynomial regression of degree 4, whereas BIC selected a regression tree of depth 5 (Figure 2).

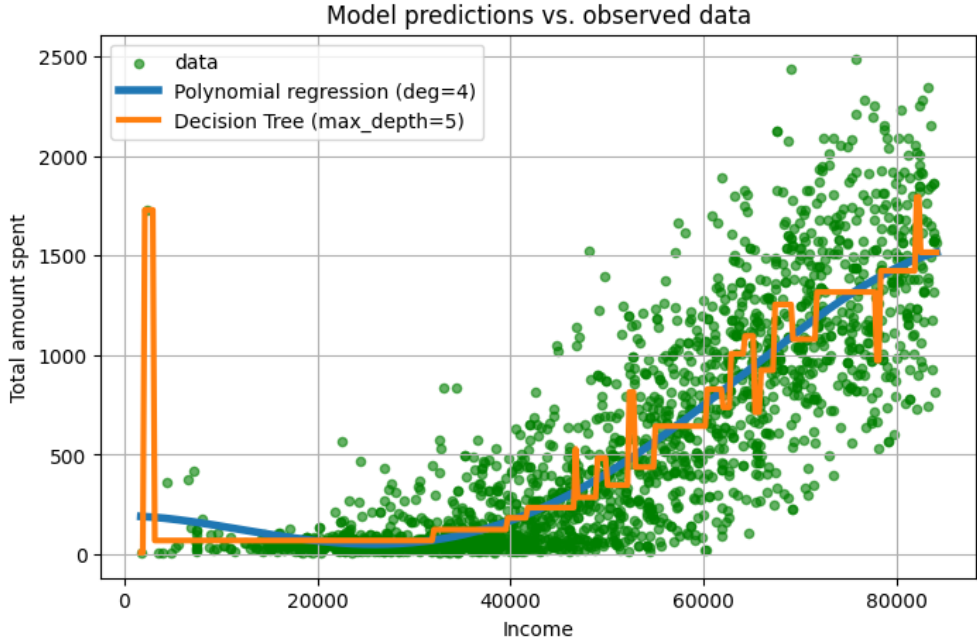


Figure 2: Experiment 2: MDL selected polynomial regression; BIC selected a regression tree.

When the MDL threshold was increased substantially (to $\tau = 100$), the modified MDL criterion selected a simpler regression tree of depth 3 (Figure 3).

Summary statistics are shown in Table 2.

Model	R^2	MSE	MAE	Params
Regression Tree (max depth = 5)	0.763	75,946.45	192.54	32
Polynomial Regression (degree = 4)	0.729	86,655.81	205.07	6
Regression Tree (max depth = 3)	0.734	85,059.48	203.71	8

Table 2: Comparison of selected models using BIC and MDL criteria.

Although BIC achieves better numerical scores, the plot indicates potential overfitting, especially around a low-income outlier.

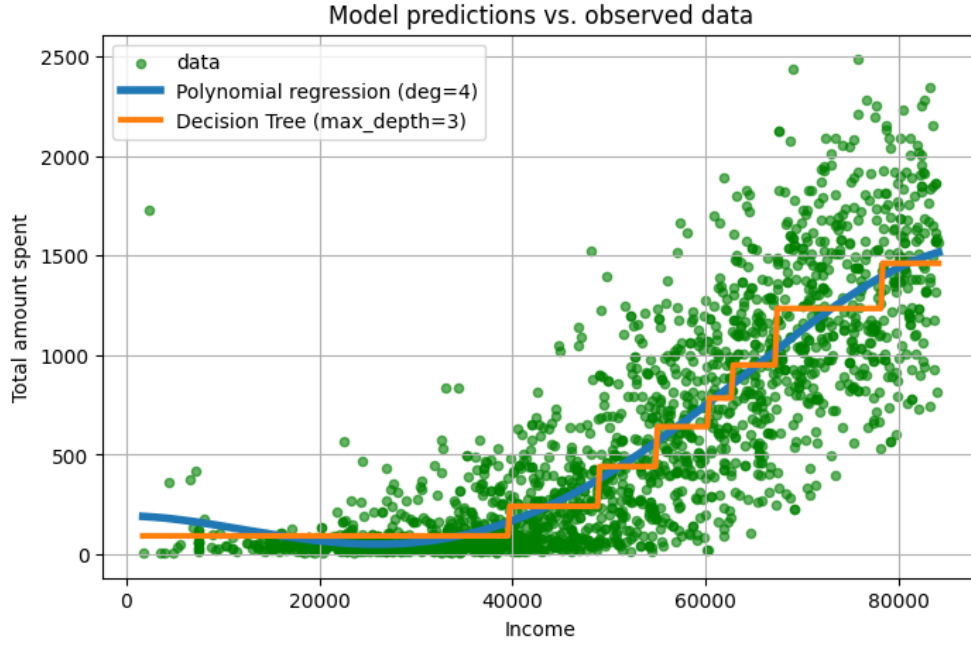


Figure 3: Experiment 2 (thresholded MDL): MDL-modified selects a regression tree of depth 3.

Experiment 3: Synthetic 5D Sphere Function

In this experiment we evaluated model selection on a synthetic 5-dimensional Sphere function.

Criterion	Best model	MSE	R^2	Params
BIC	poly (degree = 8)	1.03×10^{-22}	1.0000	1288
MDL	poly (degree = 8)	1.03×10^{-22}	1.0000	1288
MDL-modified	poly (degree = 2)	9.83×10^{-3}	0.99997	22

Table 3: Comparison of best models selected by three criteria.

The first two criteria (BIC and the original MDL variant) overfitted the data, while the modified MDL criterion selected a simpler model with slightly higher error but substantially fewer parameters.

Note: The Sphere function is essentially a sum of squared terms (with small noise), so the model selected by the modified MDL criterion is the intuitively correct one.

Discussion

Implementing MDL for regression turned out to be more challenging than expected. The original MDL formulation was susceptible to overfitting. The modified version appears to mitigate this, especially when the residual threshold is large enough. Overall, the experiments show that MDL can serve as a viable alternative to more standard criteria such as BIC. Future work may include refining the MDL definition, testing additional datasets, and extending the method to feature selection.

Bibliography

References

- [1] https://github.com/porwiii/Algorithmic_Information_project
- [2] <https://www.kaggle.com/datasets/rodsaldanha/marketing-campaign>



November, 2025

CSC-5AI25-TP

Algorithmic Information & Artificial Intelligence

Micro-study

teaching.dessalles.fr/FCI

Name: Yina Xia

MDL/BIC-based Customer Archetypes and Outlier Detection on the UCI Wholesale Customers Dataset

Abstract

I apply the Minimum Description Length principle, via BIC, to the UCI Wholesale Customers dataset to select a suitable number of clusters and to detect outliers. After comparing k-means and diagonal-covariance Gaussian Mixture Models, I focus on a five-cluster solution that yields interpretable customer archetypes. By defining a per-point compression rate, I identify about 5% of customers as “hard-to-compress” anomalies and visualize them in a low-dimensional space.

Problem

The study addresses two related questions in unsupervised learning from an information-theoretic viewpoint. First, given a real-world dataset of wholesale customers, how can we choose a reasonable number of clusters without relying on purely geometric heuristics such as

the elbow rule or silhouette? Second, once a clustering model is fixed, can we use code length—rather than just distance to a centroid—to define which customers are anomalous because they are poorly explained by the cluster structure?

Concretely, the data contain yearly spending for 440 customers across six product categories. Intuitively, we expect a few recurring customer types (e.g., fresh-food heavy, dry-goods heavy, large supermarkets) and a small minority of “weird” customers with very unusual spending patterns. The problem is to see whether MDL/BIC can both (i) point to a meaningful number of clusters and (ii) provide a principled outlier score that corresponds to these intuitive expectations.

Method

I use the UCI Wholesale Customers dataset, keeping only the six continuous annual spending features (Fresh, Milk, Grocery, Frozen, Detergents_Paper, Delicassen). Because the monetary amounts are highly skewed, I apply a \log_{1p} transform to each feature and then standardize them to zero mean and unit variance. All subsequent models are trained in this transformed space.

For model selection, I first implement k-means with a custom BIC computation under a spherical Gaussian noise model with cluster-specific variances. For each $K \in \{1, \dots, 15\}$, I run k-means, estimate per-cluster variance from the within-cluster sum of squares, compute the log-likelihood, count the number of free parameters, and obtain $\text{BIC}(K) = -2 \log L + p \log n$. In parallel, I fit diagonal-covariance Gaussian Mixture Models (GMMs) with the same range of K using sklearn’s GaussianMixture and record their built-in BIC values.

After selecting a working value of K , I analyse the resulting cluster centres as customer archetypes by mapping the centroids back to the original spending scale. For anomaly detection, I compute, for each point x_i , a negative log-likelihood under the K -cluster model, $\ell_i^{(K)}$,

and under a single global Gaussian model, $\ell_i^{(1)}$. The compression rate is defined as

$$\text{CR}_i = \frac{\ell_i^{(K)}}{\ell_i^{(1)}}.$$

Points with high compression rate are interpreted as hard to encode even when the cluster structure is known. I use the 95th percentile of the CR_i distribution as a threshold for outliers and visualize both clusters and outliers using a 2D PCA projection.

Results

The k-means-based BIC curve decreases steadily as K increases from 1 to 15, without any clear minimum in this range. This suggests that, under the spherical k-means model with cluster-specific variances, adding clusters always reduces the description length slightly, so BIC does not single out a unique “best” K in that family.

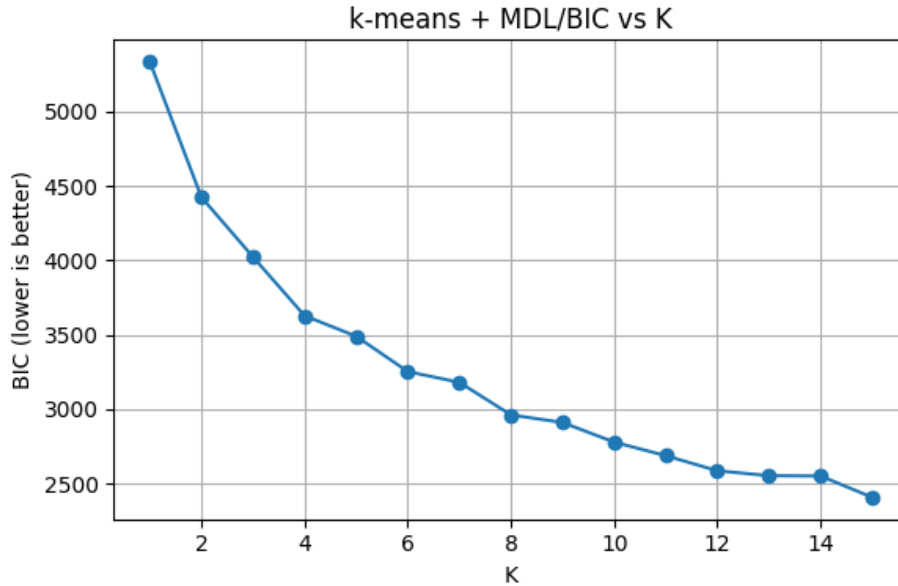


Figure 1: k-means MDL/BIC as a function of the number of clusters K .

In contrast, the silhouette score peaks at $K = 2$ (around 0.29) and then declines, indicating that, from a geometric perspective, a coarse two-cluster partition is the most compact and well-separated.

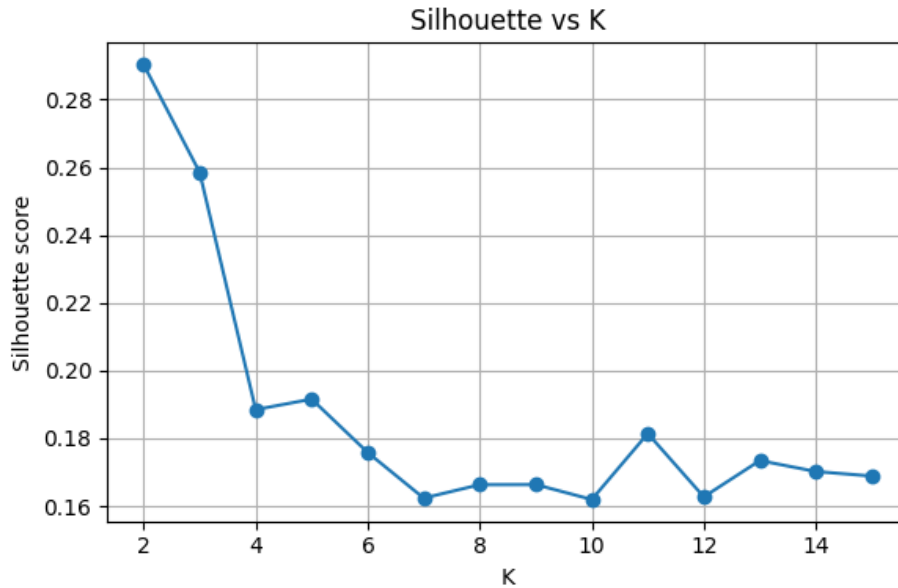


Figure 2: Silhouette score as a function of the number of clusters K .

For the diagonal-covariance GMMs, the behaviour is different: the GMM BIC curve shows a clear minimum at $K = 5$. Small values of K (1 or 2) fit the data too coarsely and result in high BIC, while very large K are penalised heavily by the increasing number of parameters. I therefore adopt $K = 5$ as a working compromise.

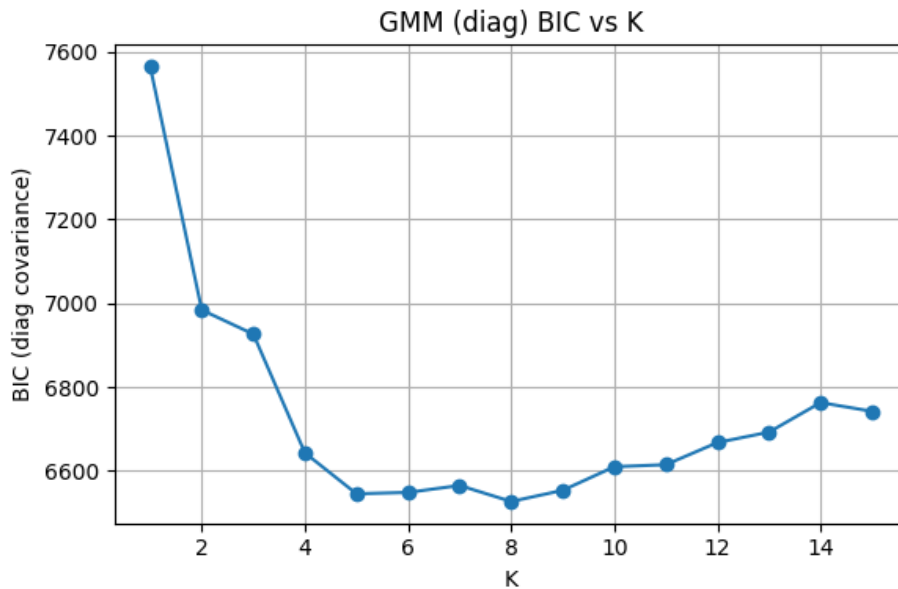


Figure 3: BIC for diagonal-covariance GMMs as a function of K .

In the five-cluster solution, the cluster centres reveal distinct spend-

ing profiles. One cluster spends very heavily on Milk, Grocery and Detergents_Paper but very little on Fresh and Frozen (a dry-goods / household-oriented profile), another cluster is extremely large in Fresh and Frozen (fresh + frozen heavy clients), and one cluster shows high spending in almost all categories (large mixed retailers). The remaining clusters correspond to more moderate or mixed profiles.

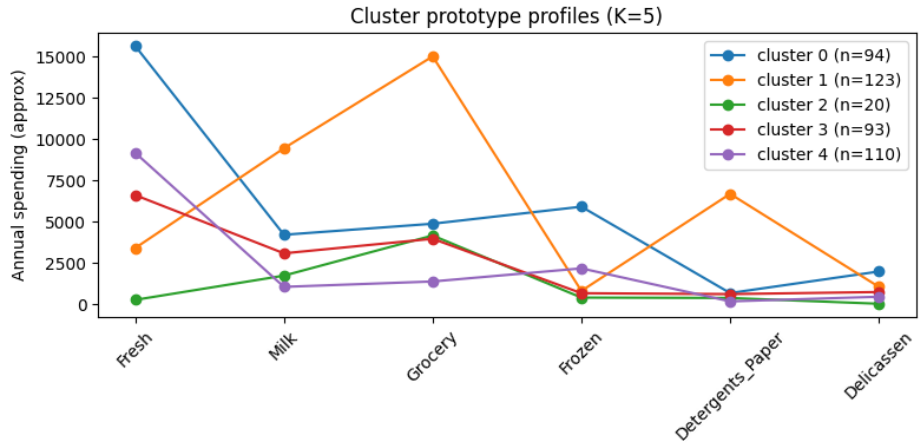


Figure 4: Approximate annual spending profiles of the five cluster prototypes.

The distribution of compression rates CR_i is concentrated between roughly 0.4 and 0.8 for most customers, with a small right tail above 1. The 95th percentile is about 1.05; using this as a threshold labels 22 customers (around 5% of the dataset) as outliers.

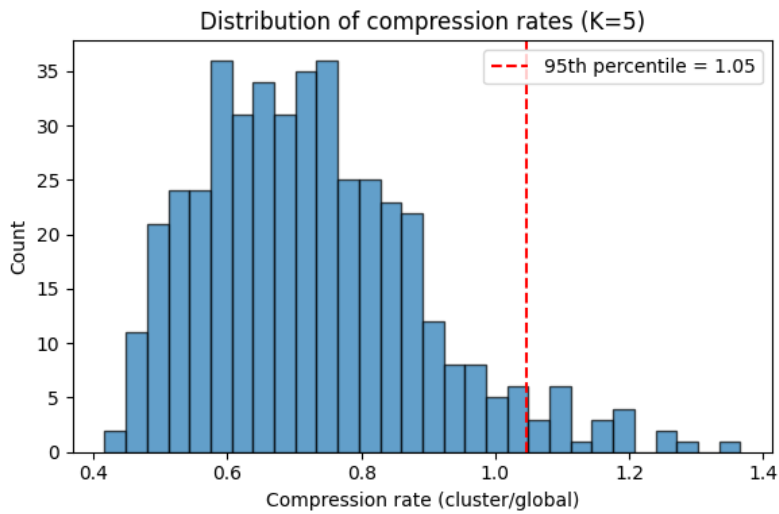


Figure 5: Distribution of compression rates for $K = 5$; the dashed line marks the 95th percentile.

In a PCA projection of the data, these hard-to-compress customers typically appear at cluster borders, in-between clusters, or relatively isolated, rather than inside dense cores.

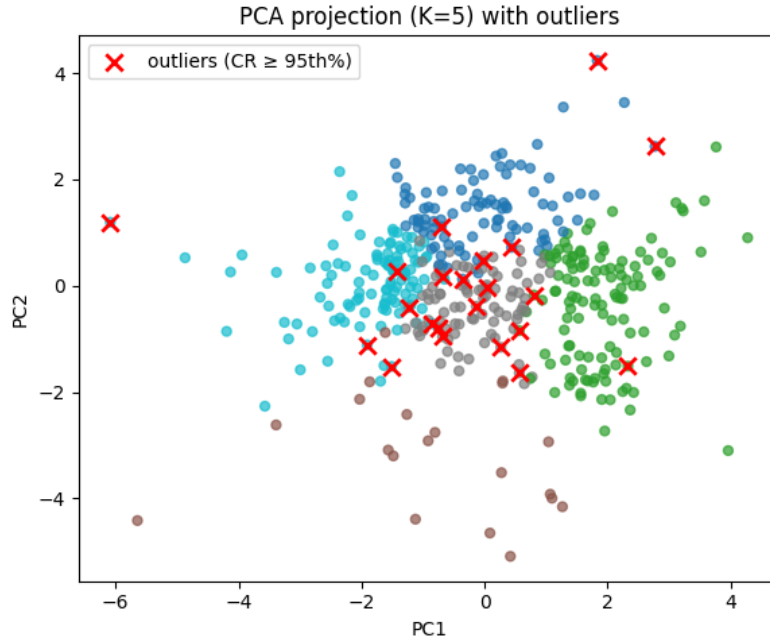


Figure 6: PCA projection for $K = 5$ with outliers (compression rate \geq 95th percentile) shown in red.

Discussion

The comparison between k-means and GMM highlights how strongly MDL/BIC-based model selection depends on the underlying model family. With k-means and spherical cluster-specific noise, BIC keeps decreasing on the explored range and does not indicate a clear optimum, whereas in the more expressive GMM(diag) family, BIC strongly prefers $K = 5$. This illustrates that “model complexity” and its optimal trade-off with data fit are not absolute, but relative to the chosen parametrisation and noise assumptions.

The compression-rate view of outliers provides a more unified perspective than geometric distance alone. Instead of defining anomalies as points far from a centroid, I define them as those that remain expensive to encode even after introducing a cluster structure. The fact that

these hard-to-compress customers also lie at the edges or in-between regions of the PCA plot suggests that the information-theoretic and geometric views are consistent here: the most “surprising” customers in terms of description length are also geometrically unusual.

The study has some limitations: the dataset is small (440 samples, 6 dimensions), and both k-means and GMM rely on Gaussian assumptions that may only roughly approximate real spending behaviour. With richer models (full-covariance GMMs or non-Gaussian mixtures), the preferred number of clusters and the set of outliers could change. Nevertheless, this micro-study shows that MDL/BIC can simultaneously guide cluster selection and anomaly detection in a real dataset, using the same coding-length framework.

Bibliography

- [1] UCI Machine Learning Repository – Wholesale customers Data Set. <https://archive.ics.uci.edu/ml/datasets/Wholesale+customers>
- [2] J. Rissanen. “Modeling by shortest data description.” *Automatica*, 14(5):465–471, 1978.
- [3] G. Schwarz. “Estimating the dimension of a model.” *Annals of Statistics*, 6(2):461–464, 1978.



November, 2025

CSC-5AI25-TP

Algorithmic Information & Artificial Intelligence

Micro-study

teaching.dessalles.fr/FCI

Name: Giuliani Astrid

Measuring perceived musical pleasure using complexity

Abstract

I explore how AIT complexity metrics can model listeners' subjective musical enjoyment. I compute predictive uncertainty and compressibility scores over diverse audio excerpts and use the notion of "unexpectedness" to estimate pleasure.

Problem

Understanding why certain pieces of music are perceived as pleasurable while others are not remains an open challenge in music cognition. Although many factors (cultural background, familiarity, emotional context) are known to influence musical enjoyment, we still lack a quantitative framework capable of linking the structure of the audio signal itself to subjective pleasure. Existing theories suggest that a balance between predictability and surprise plays a key role [4, 3], but this relationship has rarely been tested using direct, information-theoretic mea-

asures applied to real audio. The core problem is to determine whether higher compressibility in the audio signal correlates with greater musical enjoyment, while also acknowledging that excessive predictability may reduce perceived pleasure.

Method

To study the relationship between musical pleasure and complexity, I computed two complementary measures directly from audio signals. First, in `compute_cw.py`, I estimated the world complexity C_W , a proxy for surprise, defined as the negative log-likelihood of a musical piece under a large transformer model.

The audio was tokenized using a VQ-VAE encoder (vocabulary size 2127, one token ≈ 2.9 ms), and token probabilities were obtained from OpenAI Jukebox transformer, trained on 1.2 million songs [2]. I computed

$$C_W = \sum_t -\log_2 P_m(s_t | s_{<t})$$

where P_m is the model’s predicted probability for token s_t given prior tokens $s_{<t}$.

Second, I approximated in description complexity C by compressing the token sequence using a lossless LZMA algorithm. The compressed file size in bits served as an estimate of C .

Finally, I defined unexpectedness as the difference

$$U = C_W - C$$

as introduced in class [1] and following the idea that pleasurable stimuli are “complex to generate but simple to describe.”

Because raw complexity values scale with the number of tokens, and because the pieces in my dataset differed in length, I applied a final normalization step. All scores (C_W, C, U) were divided by the total number of tokens in the piece, ensuring that comparisons were not biased by audio duration.

All metrics were computed in `compute_all.py` on multiple genres (electro, reggae, classical piano, hard rock) as well as a baseline (white noise), which one can find in the `wav_files` directory.

Results

To better understand how world complexity evolves within each musical piece, I visualized the temporal profile of C_W averaged over 1-second intervals. In Figure 1, 2, 3, 4, and 5 I present the temporal C_W curves for each musical style analyzed.

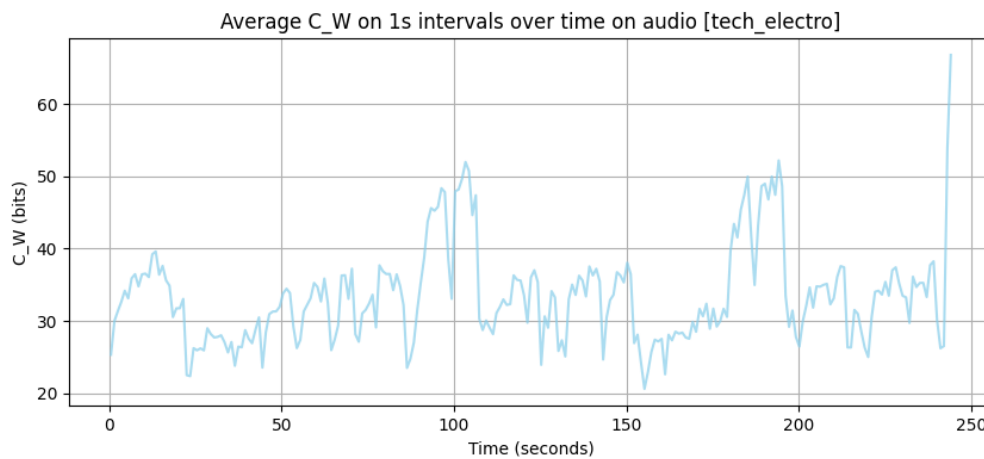


Figure 1: Tech electro

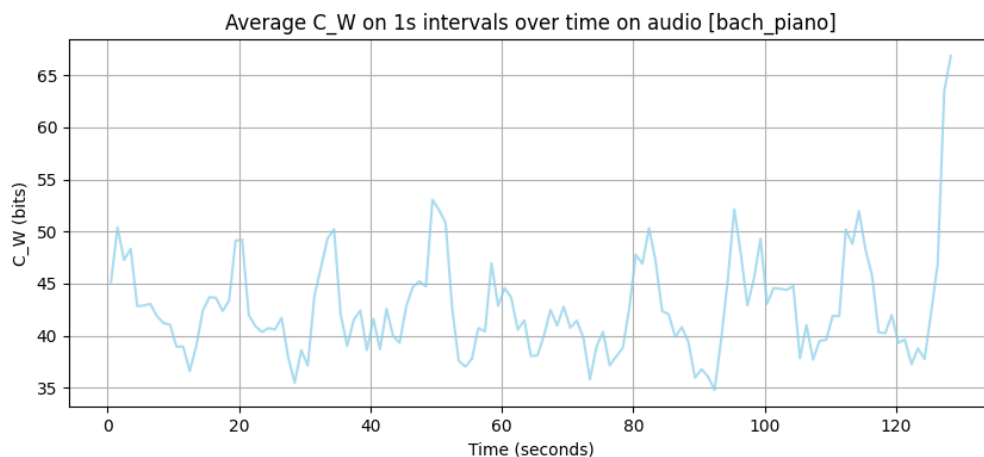


Figure 2: Bach piano

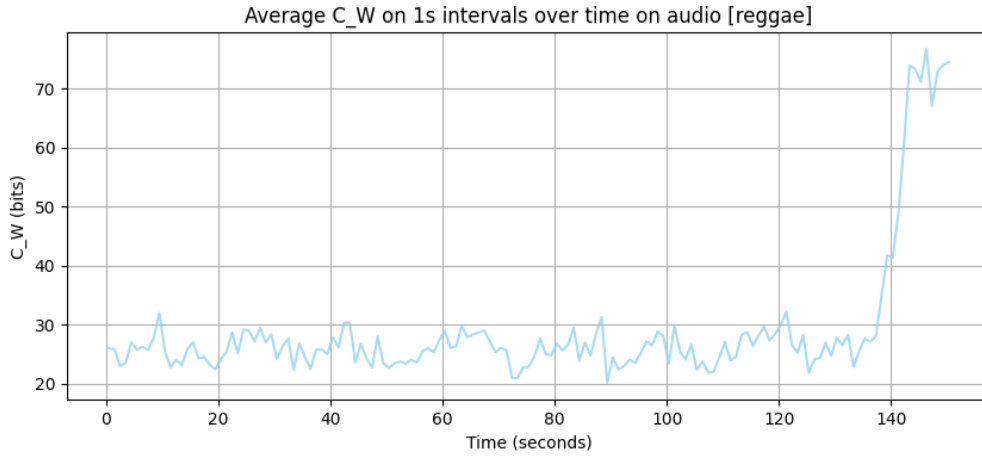


Figure 3: Reggae

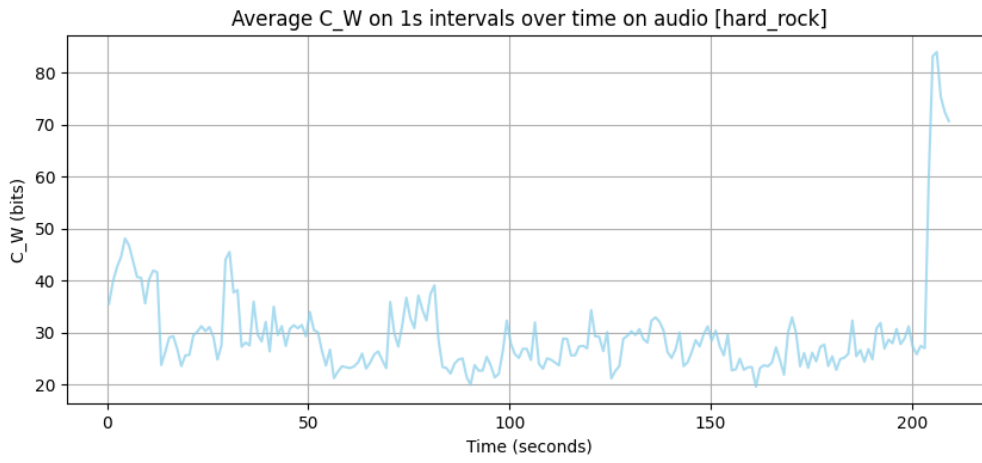


Figure 4: Hard rock

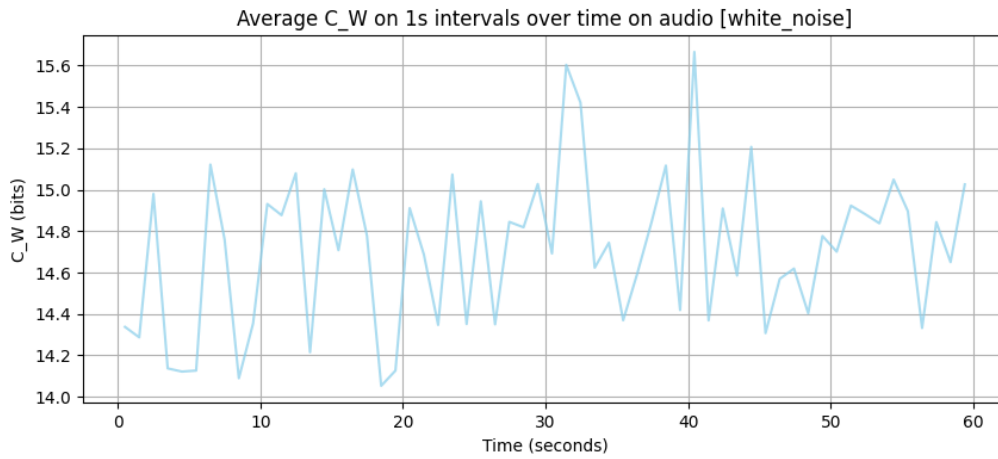


Figure 5: White noise

Table 1 summarizes the normalized values of world complexity C_W , description complexity C , and unexpectedness U computed for all au-

dio excerpts in the dataset.

Audio	C_W (bits/token)	C (bits/token)	U (bits/token)
Tech Electro	33.528	9.052	24.476
Bach Piano	42.777	8.598	34.179
Reggae	29.291	11.057	18.234
Hard Rock	29.755	10.236	19.518
White Noise	14.709	12.409	2.299

Table 1: Normalized complexity measures for different musical pieces

My results reveal that pieces generally considered musically rich, such as electronic, reggae, hard rock, or classical piano, exhibited high world complexity C_W , indicating that the model found them relatively rare or surprising in the musical space. At the same time, their description complexity C remained comparatively low, showing that despite being surprising to the model, their internal structure remained compressible.

White noise provided a useful contrast: it scored extremely low on C_W . This doesn't necessarily reflect maximal predictability for the model but can be explained by the fact that when the model is facing white noise, it is expecting everything with the same probability and therefore the distribution flattens and nothing is extremely unpredictable). It however scored high on C , meaning it is internally patternless and difficult to compress, aligning with its perceived lack of musical pleasure.

The derived unexpectedness measure $U = C_W - C$ was highest for genres typically judged enjoyable, and lowest for white noise (which is coherent with the fact that white noise is not considered pleasurable music), supporting the hypothesis that pleasurable music maximizes the gap between external unpredictability and internal compressibility. The ranking of U across pieces (Bach > electro > hard rock > reggae \gg white noise) is consistent with this interpretation.

Discussion

Overall, the results obtained align qualitatively with my initial expectations: musical pieces that are both surprising relative to the model’s learned distribution (high C_W) and internally structured (low C) tend to produce higher values of U , which I interpret as proxies for musical pleasure. White noise, as anticipated, scores poorly on this metric. However, these results remain preliminary and cannot be directly compared to human judgments, as no systematic validation was performed with reported listener pleasure. This constitutes a major limitation of the study.

The primary obstacle was the computational cost of the Jukebox model, which made it impossible to process a sufficiently large and diverse set of audio samples within Google Colab’s GPU restrictions. A proper validation would require running the method on a large dataset for which human enjoyment ratings are available, then correlating the model-derived pleasure estimates with empirical behavioral data.

A second limitation arises from the quantization step used for tokenization, which is inherently lossy. Some musical details are removed before complexity is computed, potentially distorting both C_W and C . While this tokenization is necessary for compatibility with the Jukebox architecture, exploring alternative or higher-fidelity encodings could improve the reliability of the complexity metrics.

Another issue concerns imperfect conditioning of the transformer model, especially with respect to the expected length of the pieces. As visible in the plots, Jukebox tends to remain highly uncertain near the end of each excerpt, artificially increasing C_W . Although cropping endings could reduce this bias, it might also remove musically meaningful material. More explicit length-conditioning or padding strategies would be required for accurate likelihood estimation.

Moreover, the model was not conditioned on genre, artist, or other contextual priors. Yet human expectations are strongly shaped by such

factors: a listener familiar with a genre may find predictable what the model considers surprising. Conditioning on metadata could therefore allow investigation of how “expectedness” varies with listener background. In this project, however, I remained in the neutral scenario of a listener with no prior assumptions.

Finally, the method depends heavily on the training distribution of Jukebox, which may not accurately reflect the diversity of human musical preferences. A model trained on a broader or more culturally representative dataset might produce complexities more aligned with real listeners’ expectations.

Overall, while the theoretical framework appears promising, addressing these methodological limitations is necessary before establishing any solid link between algorithmic complexity measures and perceived musical pleasure.

Bibliography

References

- [1] Jean-Louis Dessalles. Chapter 5 - subjective information and simplicity. In *Algorithmic Information and A.I. course*, 2025.
- [2] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. <https://openai.com/blog/jukebox/>, 2020.
- [3] Nicholas J. Hudson. Musical beauty and information compression: Complex to the ear but simple to the mind? *BMC Res Notes*, 4, 2011.
- [4] Jürgen Schmidhuber. Driven by compression progress: A simple principle explains essential aspects of subjective beauty, novelty, surprise, interestingness, attention, curiosity, creativity, art, science, music, jokes. *Journal of SICE*, 48:21–32, 2009.



November, 2025

CSC-5AI25-TP

Algorithmic Information & Artificial Intelligence

Micro-study

teaching.dessalles.fr/FCI

Name: Zeinab GHAMLOUCH & Alessa MAYER

"Interestingness" of News Headlines

Abstract

This project investigates what makes certain news headlines more “interesting” than others through the lens of Simplicity Theory and algorithmic information. We focus on a controlled subset of news events, death-related headlines, to reduce topical variability and study how linguistic unexpectedness interacts with real-world popularity. Using a large dataset of death-event headlines collected from Hacker News, we computed description complexity (C_d), world complexity (C_w), and derived interestingness scores $U = C_w - C_d$. We also estimated the public salience of the individuals mentioned using both Wikipedia pageview statistics and LLM-based popularity judgments. By comparing headline interestingness with measures of attention on social platforms, this study explores whether the informational properties of headlines and the prominence of the individuals involved systematically influence perceived newsworthiness. We developed and compared two distinct formulations of interestingness: The first (Method A) combines linguistics

tic unexpectedness with observed user engagement, while the second (Method B) integrates unexpectedness directly with independent measures of the mentioned individual’s real-world popularity. This dual approach allows us to study the relationship between informational structure, public prominence, and audience engagement.

Problem

Despite the abundance of digital news and the constant flow of online information, it remains unclear why certain headlines capture public attention more than others. Traditional explanations attribute newsworthiness to factors such as novelty, emotional impact, or social relevance, but these accounts often lack quantifiable measures grounded in information theory. Simplicity Theory proposes that an event is perceived as interesting when it is simple to describe yet surprising relative to prior expectations; however, empirical evaluations of this framework on real-world headlines remain limited.

A major challenge in testing such theories lies in the heterogeneity of news topics: headlines differ widely in subject matter, structure, and audience interest, making comparisons noisy and inconclusive. To address this, we restrict our analysis to a single, conceptually coherent class of events—death-related news headlines—allowing us to examine variation in interestingness within a controlled semantic domain.

Within this domain, an additional open question arises: how does linguistic unexpectedness interact with the real-world prominence of the individuals mentioned and user engagement? In other words, we seek to untangle a three-way relationship between a headline’s informational structure, the fame of its subject, and the audience’s behavioral response. This problem motivates two distinct investigations: we will examine both how engagement-based interestingness relates to the subject’s prominence, and conversely, how prominence-based interestingness relates to engagement.

Method

Our methodological pipeline consisted of four main stages: (1) data collection, (2) event-type filtering and person-name extraction, (3) computation of information-theoretic quantities (C_d , C_w , U), and (4) estimation of individual popularity and construction of composite interestingness scores. Each stage is described below.

Data Collection

To focus on a controlled class of news events, we extracted headlines related to death announcements from the Hacker News (HN) platform using the Algolia HN Search API [7]. We queried multiple death-related expressions (e.g., “dies”, “dead”, “passes away”, “death of”, “dies aged”) and retrieved up to 1000 results per query term. All results were deduplicated by `objectID` and sorted by HN points. We retained the top 1000 most popular headlines for subsequent analysis.

Filtering Death Events Involving Persons

Death-related expressions may also refer to objects (e.g., “dead battery”), companies, or abstract entities. To isolate events concerning human individuals, we applied Named Entity Recognition (NER) using the `spaCy en_core_web_sm` model. Headlines were retained only when the NER system detected at least one entity with label `PERSON`. Manual verification corrected misclassifications and ensured dataset validity. For each headline, we extracted the primary individual mentioned.

Computing Description Complexity C_d

Description complexity measures how compressible a headline is: the more regularity or predictability it contains, the shorter its compressed form. Because no single compressor perfectly approximates Kolmogorov

complexity [4], we used three widely-adopted, algorithmically distinct schemes:

- **gzip** – based on DEFLATE [2] (LZ77 + Huffman coding). Captures repetitive patterns via sliding-window matching.
- **bz2** – based on the Burrows–Wheeler Transform (BWT) [1] + Huffman coding [3]. Detects longer-distance structure and systematic symbol reordering.
- **lzma** – based on Lempel–Ziv Markov chains and range coding [8]. Models higher-order dependencies and yields stronger compression on short text.

For each headline, we computed:

$$C_{d,\text{method}} = \frac{\text{compressed bytes}}{\text{raw length}}$$

and averaged them:

$$C_d = \frac{C_{d,\text{gzip}} + C_{d,\text{bz2}} + C_{d,\text{lzma}}}{3}.$$

The rationale is that each compressor captures different structural regularities, and the mean provides a more stable approximation of underlying description complexity than any single scheme alone.

Computing World Complexity C_w

World complexity captures the rarity of the concepts appearing in a headline. We computed a simple approximation using inverse document frequency (IDF) over the headline dataset:

$$C_w = \sum_{w \in \text{title}} \text{IDF}(w), \quad \text{IDF}(w) = \log \frac{N}{\text{df}(w) + 1}.$$

This yields higher values for semantically rare or distinctive words, in accordance with Simplicity Theory.

Linguistic Unexpectedness

Following Simplicity Theory, linguistic unexpectedness is defined as:

$$U = C_w - C_d.$$

We then standardized U using z-scores to ensure comparability across headlines.

Measuring Behavioral Engagement

To incorporate real user responses, we used engagement indicators from Hacker News:

$$E = \log(1 + \text{points}) + 0.7 \log(1 + \text{comments}).$$

Two variants were computed:

1. **No-recency** – raw engagement.
2. **With-recency** – weighted by exponential time decay with half-life 180 days:

$$E_{\text{decay}} = E \cdot 2^{-\text{age}/180}.$$

All engagement values were z-normalized.

Estimating Real-World Popularity of Individuals

To quantify the prominence of the deceased individuals mentioned in headlines, we implemented two independent measures:

1. **LLM-based Popularity Measure ($M_{\text{LLM-pop}}$)**. A large language model (Llama 3.2 [5]) was prompted to assign each person a popularity score on a 1–10 scale, based on cultural influence and public recognition.
2. **Wikipedia Pageview Popularity Measure ($M_{\text{Wiki-pop}}$)**. Using the Wikimedia Pageviews API [6], we retrieved the past 30

days of daily pageviews for the Wikipedia article corresponding to each person. Total views were log-transformed and normalized to a 0–10 scale.

Final Interestingness Score

Method A: Unexpectedness + Engagement

We defined two composite interestingness measures using engagement:

$$I_{\text{no-recency}} = 0.6 U_z + 0.4 E_{\text{no-recency},z}, \quad I_{\text{recency}} = 0.6 U_z + 0.4 E_{\text{decay},z}.$$

Both were log-transformed and normalized to a 0–10 scale before producing final rankings. These represent our baseline formulation.

Method B: Unexpectedness + Popularity

As an additional method, we introduce a second set of composite interestingness scores that instead of engagement combine linguistic unexpectedness with real-world popularity measures of the person mentioned.

Specifically, we compute:

$$I_{\text{llm}} = 0.6 U_z + 0.4 M_{\text{LLM-pop}}, \quad I_{\text{wikipedia}} = 0.6 U_z + 0.4 M_{\text{Wiki-pop}}.$$

This second method allows us to reverse the comparison: instead of examining whether interestingness correlates with popularity, we build popularity directly into the interestingness formula and ask how these modified scores relate to engagement ($E_{\text{no-recency},z}$ and $E_{\text{decay},z}$).

Results

Comparing Unexpectedness+Engagement (Method A) with Popularity

We produced four quantitative comparisons between our computed interestingness scores and two independent measures of real-world popularity: Wikipedia pageview statistics and LLM-based popularity judg-

ments. For each comparison, we generated a scatter plot with a linear regression line and reported the Pearson correlation coefficient.

Interestingness vs. Wikipedia Popularity

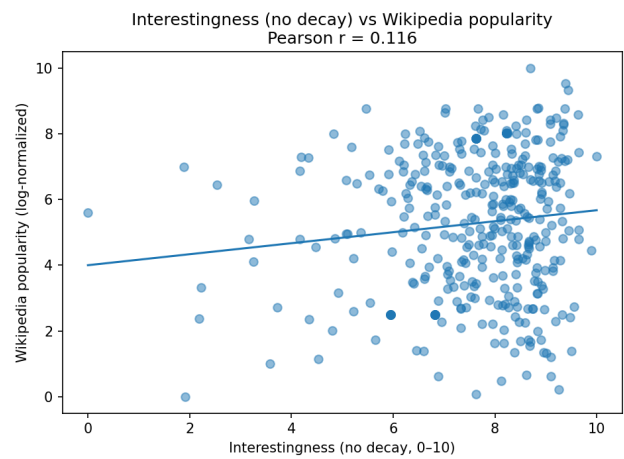


Figure 1: Interestingness (No decay) vs Wikipedia Popularity

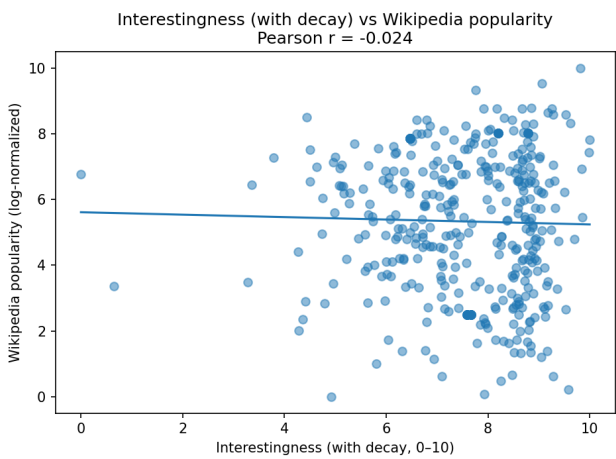


Figure 2: Interestingness (Decay) vs Wikipedia Popularity

Figure 1 compares the interestingness score computed without recency correction to the Wikipedia pageview-based popularity measure. The distribution shows a wide spread, with no strong linear dependence between the two quantities. The correlation is low, suggesting that linguistic unexpectedness and engagement-based interestingness capture a dimension distinct from general public salience of the person involved in the news event.

Figure 2 performs the same comparison using the recency-adjusted interestingness score. The correlation remains weak and of similar magnitude, indicating that temporal decay does not substantially affect the relationship between headline interestingness and global celebrity status. This supports the interpretation that the two concepts represent different kinds of informational signals.

Interestingness vs. LLM-based Popularity

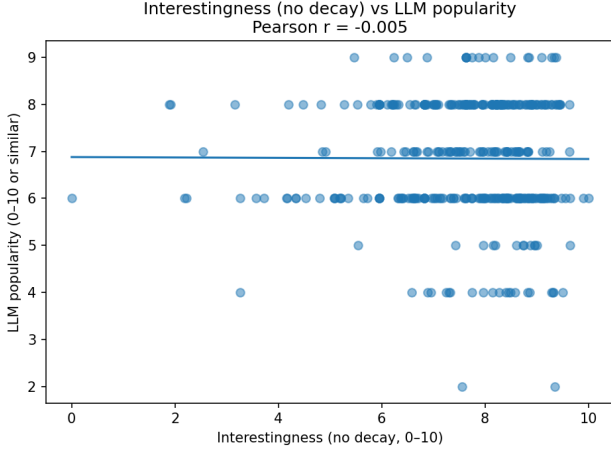


Figure 3: Interestingness (No decay) vs LLM Popularity

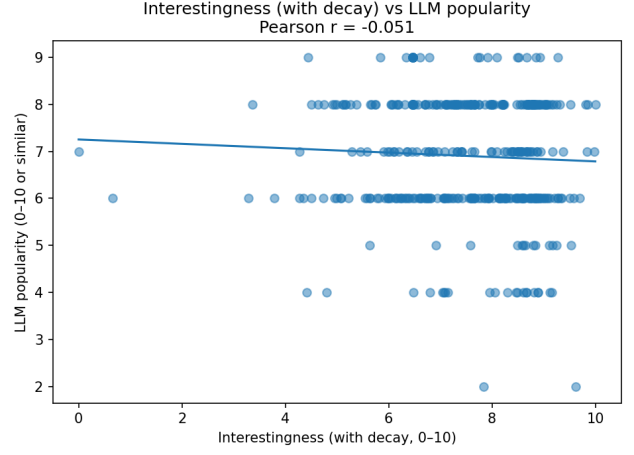


Figure 4: Interestingness (Decay) vs LLM Popularity

Figures 3 and 4 compare interestingness (with and without temporal decay) against the popularity judgments provided by a large language model. As with the Wikipedia-based measure, both scatter plots show low correlations. Although a few highly salient individuals receive both high popularity scores and moderately high interestingness scores, the general trend remains weak.

This suggests that LLM-based perceptions of public notoriety, despite aligning roughly with human cultural knowledge, do not strongly predict the structural or engagement-driven interestingness of the headlines. Instead, the two measures appear to capture distinct dimensions: linguistic complexity and behavioral engagement on the one hand, and perceived societal prominence on the other.

Comparing Unexpectedness+Popularity (Method B) with Engagement

We produced four additional comparisons examining whether the composite scores based on unexpectedness and real-world popularity ($I_{\text{wikipedia}}$ and I_{llm}) align with Hacker News behavioral engagement ($E_{\text{no-recency},z}$ and $E_{\text{decay},z}$).

Interestingness vs. Engagement Scores without temporal decay

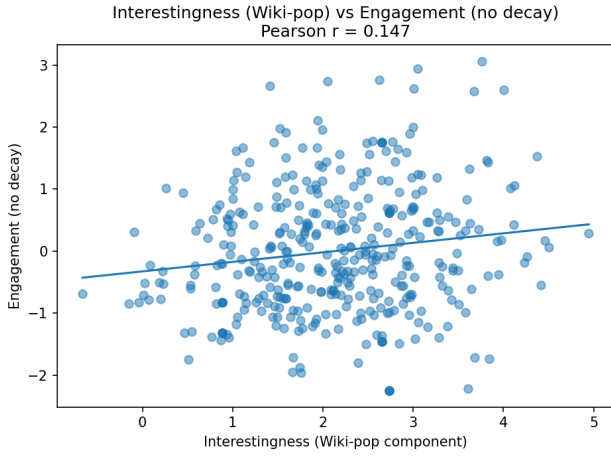


Figure 5: Interestingness (Wiki-pop) vs Engagement without decay

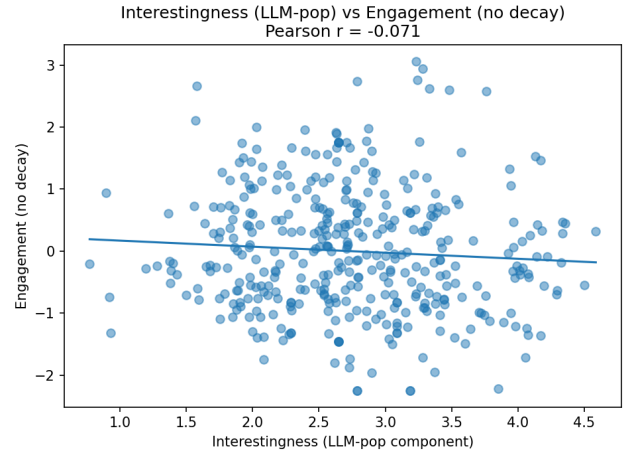


Figure 6: Interestingness (LLM-pop) vs Engagement without decay

In Figure 5 we observe that the correlation between the Wikipedia-based interestingness score and engagement is positive but weak, with a Pearson coefficient of $r = 0.147$. This represents the strongest correlation observed across all experiments in this study, suggesting that a combination of linguistic surprise and pageviews has a slight, non-negligible relationship with user engagement. In contrast, Figure 6 shows that when popularity is derived from LLM judgments, the correlation with engagement drops to $r = -0.071$. This indicates a near-zero or slightly negative relationship, implying that the LLM's perception of "fame" combined with unexpectedness does not align with what drives engagement on the platform.

Interestingness vs. Engagement Scores with temporal decay

Figure 7 displays a correlation of $r = 0.138$. This is consistent with the non-decayed results, confirming that the weak positive signal provided by Wikipedia pageviews also holds with the temporal weighting. Figure 8 yields a Pearson coefficient of $r = 0.049$. While positive, this value is statistically negligible, reinforcing the finding that LLM-estimated popularity adds little predictive power regarding audience

attention in this study.

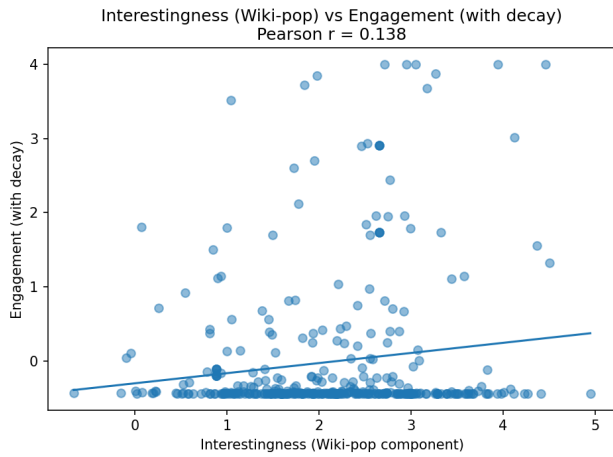


Figure 7: Interestingness (Wiki-pop) vs Engagement with decay

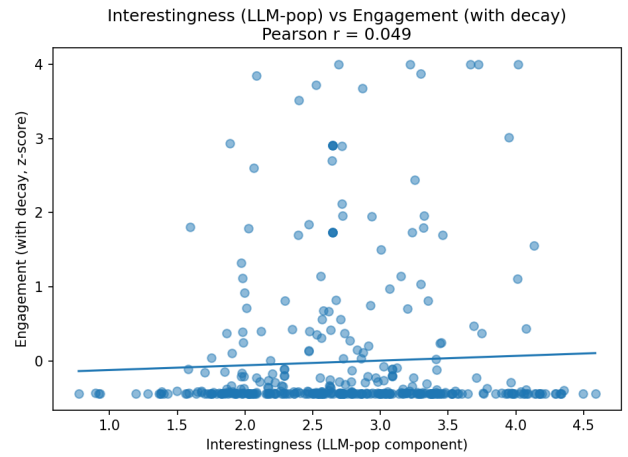


Figure 8: Interestingness (LLM-pop) vs Engagement with decay

Summary

Our analysis yielded consistently low correlations across all experiments. When using Method A (Interestingness = Unexpectedness + Engagement), the resulting scores showed weak relationships with both Wikipedia and LLM-based popularity measures. Similarly, when using Method B (Interestingness = Unexpectedness + Popularity), the composite scores showed weak correlations with actual user engagement, whether unadjusted or with temporal decay consideration. The strongest observed correlation was given by Method B, comparing the interestingness score combining unexpectedness and Wikipedia-based interestingness score with non-decay user engagement, yielding a Pearson coefficient of $r = 0.147$. Despite being the highest in our study, this value confirms only a very weak positive relationship.

Discussion

The goal of this project was to investigate whether the informational structure of news headlines—as captured by Simplicity Theory—is systematically related to how much attention these headlines receive, and

to what extent this relationship interacts with the real-world popularity of the individuals mentioned. By restricting our analysis to death-related headlines, we controlled for topic variability and allowed unexpectedness to be compared within a coherent category of events.

The results show that linguistic interestingness scores derived from Simplicity Theory (the difference $U = C_w - C_d$) exhibit only weak correlations with behavioral engagement measures (Hacker News points and comments) or external measures of a person’s cultural salience (Wikipedia pageviews and LLM-generated popularity judgments). This suggests that the death of a very famous individual does not automatically produce a linguistically “remarkable” headline, nor does a linguistically surprising headline necessarily correspond to high interest in the underlying news event.

These findings align with the expectation that newsworthiness is multifactorial. Simplicity Theory predicts that events with high unexpectedness should be perceived as more interesting, but in real-world settings this linguistic signal competes with many other variables: existing social narratives, emotional content, community interests, and platform dynamics. Deaths of highly prominent individuals often follow familiar headline templates (e.g., “X has died”), which reduces descriptive complexity C_d but does not necessarily increase world complexity C_w , resulting in relatively moderate U values despite strong public attention. Conversely, individuals with limited public visibility may produce headlines that are linguistically “unexpected” without generating significant engagement.

Several methodological limitations also influence these outcomes. First, our extraction of personal names, while verified manually, relies on NER tools that may miss edge cases or merge multiple entities. Second, Hacker News engagement reflects the interests of a specific technical community rather than the general population, which may skew the relationship between salience and attention. Third, Wikipedia pageviews, while a strong proxy for notoriety, fluctuate with indepen-

dent events and do not capture long-term fame consistently. The LLM-based popularity estimates provide an alternative perspective but depend on the implicit knowledge and biases of the language model.

Despite these limitations, the project demonstrates how algorithmic-complexity measures can be operationalized on real-world news data and combined with behavioral and semantic indicators. Promising directions for future work include analyzing other event categories (e.g., product launches, political resignations), exploring semantic embeddings to quantify conceptual “distance,” or modeling time dynamics more explicitly by examining engagement as it unfolds.

Overall, this study suggests that linguistic unexpectedness is one component of newsworthiness but interacts in complex ways with prior knowledge, cultural prominence, and audience-specific interests.

Bibliography

- [1] Michael Burrows, D J Wheeler D I G I T A L, Robert W. Taylor, David J. Wheeler, and David Wheeler. A block-sorting lossless data compression algorithm. 1994.
- [2] P. Deutsch. Rfc1951: Deflate compressed data format specification version 1.3, 1996.
- [3] David A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.
- [4] A. N. Kolmogorov. Three approaches to the quantitative definition of information. *International Journal of Computer Mathematics*, 2(1-4):157–168, 1968.
- [5] Meta AI. Llama 3.2: Revolutionizing edge ai and vision with open models, 9 2024.

- [6] Wikimedia Foundation. Wikimedia REST API: Pageviews Analytics. API Endpoint, 2025. Accessed: 2025-11-24. Endpoint: `/metrics/pageviews/`.
- [7] Y Combinator. Hacker News. <https://news.ycombinator.com>, 2025. Data retrieved via Algolia API. Last accessed: 10/11/2025.
- [8] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.



November, 2025

CSC-5AI25-TP

Algorithmic Information & Artificial Intelligence

Micro-study

teaching.dessalles.fr/FCI

Name: Ada Yetis

Categorization as Compression

Abstract

Categorization is one of the most widely used and natural applications of compression in our daily lives. This study aimed to explore compression in categorization from two different aspects; the relative simplification of a subcategory given a category and the relative simplification of a category given its attributes. It was discovered that subcategories were always less complex, although to varying degrees, when given the main category. Although the complexity of the main category was also generally reduced when given attributes, there were cases for which being given an attribute made the category more complex.

Problem

This study will focus on the compression achieved at various stages of categorization. Mainly it will focus on how much simpler subcategories of a category are when given the category and on how much simpler a

category is when given various attributes of it. It will aim to explore what might lead to these differences.

Background Information

Categorization is compression. Categorization is defined as the action of attributing categories to things. Categories are sets of things that all share common characteristics. Therefore, in that one word, the category name communicates that the object has every single characteristic common to said category. This means that it reduces however many bits would be necessary to encode that list of characteristics to the number of bits needed to encode the name of the category. This is a huge compression, especially given the fact that categories share many characteristics.

Recognizing a category requires detecting features that are relevant to the category and ignoring those that are irrelevant. Learning this abstraction requires paying more attention to the aforementioned relevant features compared to other attributes of the object. This means that these relevant features should carry more information about the object as we define our interactions with the object based on its category, which is communicated by the relevant feature.

Method

Tree was chosen as the main category. This choice was arbitrary. The only constraint was for the category to be general enough to allow sampling subcategories and attributes. People were asked to name a type of tree and to name a feature of a tree that was not the first that came to their mind. The phrasing for the feature was chosen specifically to create a variety of different types of attributes.

To quantify how much simpler a category was given the super category or attribute, an approximation of conditional complexity was used.

Conditional complexity was chosen since what is being quantified is the relative complexity of a term given another term so how many bits of information is necessary to encode the first term after the second term has already been encoded.

The Google frequency was used as an approximation for complexity. Specifically, approximate conditional complexity was calculated using the following formulae, where f stands for frequency:

$$f(x|y) = \frac{f(x + y)}{f(y)}$$

$$K(x|y) = \log_2\left(\frac{1}{f(x|y)}\right)$$

The results from the above formula were compared with the Kolmogorov complexity of the first term (the so called x) which was calculated as follows:

$$K(x) = \log_2\left(\frac{1}{f(x)}\right)$$

To be able to compare the change in complexity, the difference was then converted to a percentage of the complexity of the first term:

$$diff = \frac{K(x) - K(x|y)}{K(x)} \times 100$$

As there were two phenomena being studied, the study was done in two parts. In the first part, the complexity of various subcategories of the main category was calculated and compared with the conditional complexity given the main category. For the second part, the conditional complexity of the main category given various attributes was calculated and compared to the complexity of the category. In both cases, the Google total results data was extracted by hand due to changes in the Google API that prevented scraping and a Python program was used to perform the complexity calculations and calculate various basic metrics.

Results

As shown in Figure 1, the complexity of subcategories given the main category was lower than the complexity of the subcategories without any prior information. However, there were marked differences in how much simpler the subcategory was. Percentages were used to compare as a normalization measure to account for the differences in complexity of the subcategories. The lowest reduction, so the subcategory that benefitted the least from having the category defined, was still 9.26% simpler. The largest reduction was 51.46%. On average, 28.85% fewer bits were necessary to encode for a subcategory if the category was provided. The standard deviation of the values was 10.52, so there was a rather large variety in how much the complexity was reduced by.

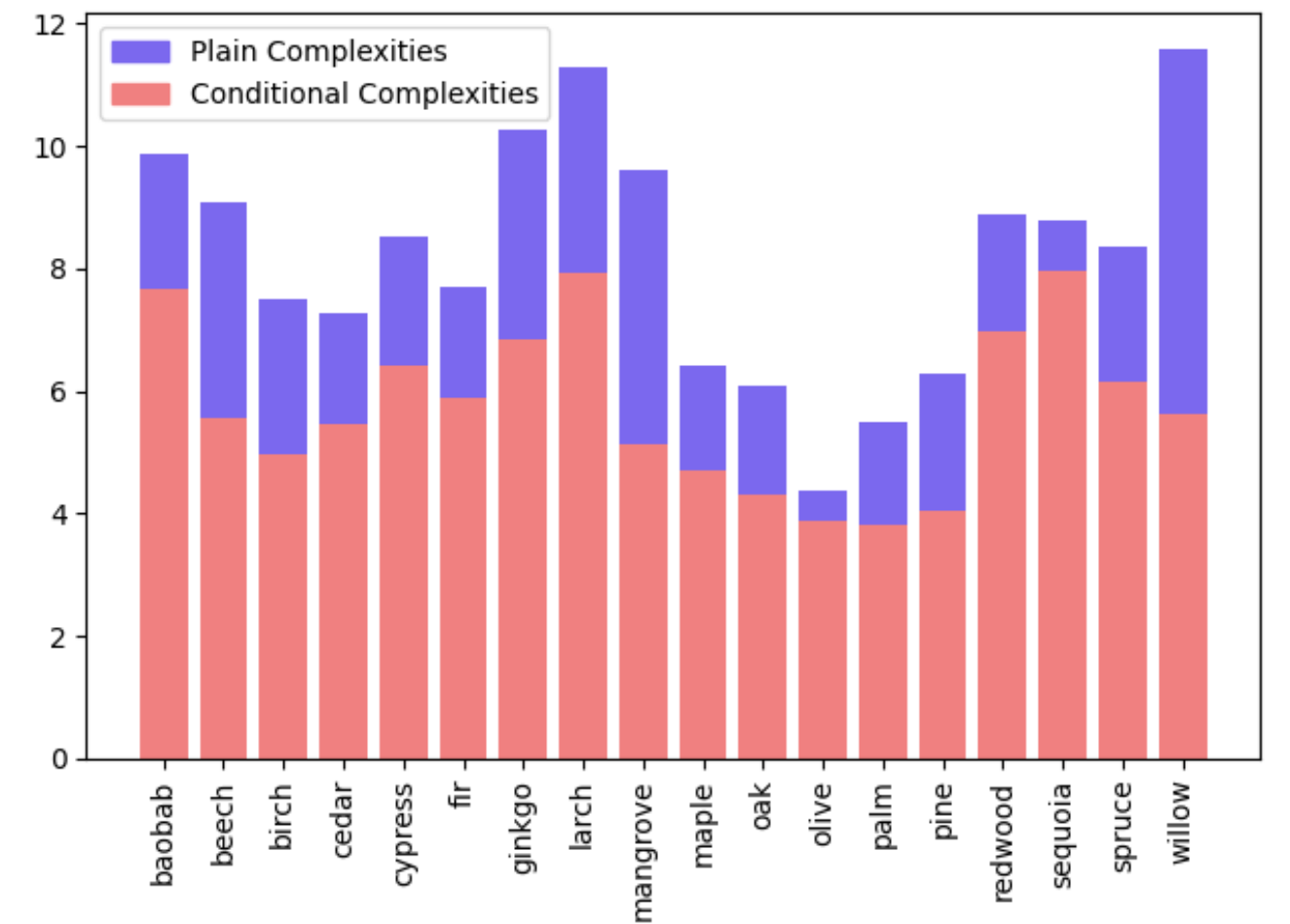


Figure 1: The conditional and plain complexity of various species of trees

As evident from Figure 2, attributes behaved less uniformly. For most attributes, the conditional complexity of the main category given the attribute was reduced, although the amount greatly varied. However, for a few of attributes, providing the term actually increased the complexity of the main category. The average change in complexity was 31.00% but increased to 39.44% if the attributes with negative complexity reduction were removed. The average of the negative impact these attributes had was -40.77% . The attribute with the maximum gain reduced the number of bits necessary to explain the main category by 84.75% while the minimum positive attribute reduced it by only 7.01%. The worst performing attribute increased the complexity by 45.66%. The standard deviation of the values was 31.32, which is a very large value and illustrates the wide variety of ways being given an attribute might impact the complexity of the main category.

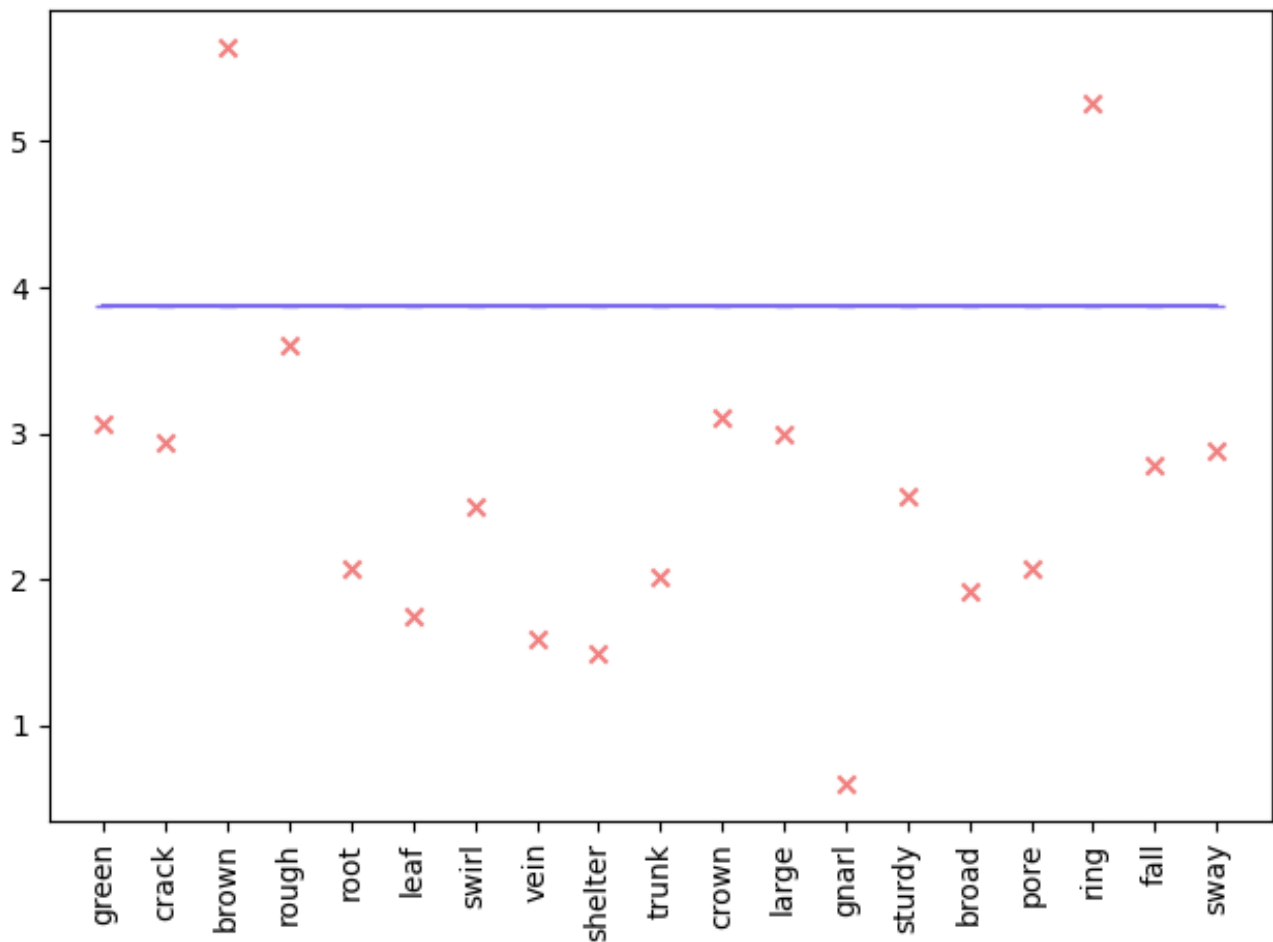


Figure 2: The conditional complexity of tree given various attributes

Discussion

The subcategory case performed as expected. The complexity of each subcategory was significantly reduced by providing the category name. The reduction in complexity was greater for subcategories for which there were few different possible meanings for the word. The smallest reduction was seen in subcategory *olive* and the largest in *willow*. This is counterintuitive since the specification of tree carries more information for something like *olive* that does not immediately come with the connotation of tree than for something like *willow* which does. However, given that conditional complexity measures the number of bits of information necessary to encode the first term given the second one, this

is to be expected. Since they are better known examples of the main category, subcategories that are more common species of tree show a greater reduction in complexity. Furthermore, for subcategories for which the word has multiple meanings and connotations, intuitively the main category provides more information because it limits the scope further. But limiting the scope also means that the explained concept is more complicated than what is communicated with just the subcategory. Specifically, because we are using Google frequency to approximate, for some concepts the limited scope can mean that their frequency is diminished significantly as the primary contexts they are seen in is different.

This was even more evident in the study of attributes. The expectation in studying the attributes was for the attributes that are primarily used to describe a category to lead to much larger reductions in complexity compared to those that were only peripheral characteristics. However, against this expectation, the attribute with the highest reduction in complexity was *gnarl* which is not a defining attribute of the category. It is, however, a word that is very rarely associated with anything other than the main category. In contrast, things that were considered defining attributes of the category such as *root* or *green* lead to much smaller reductions in complexity. This behaviour can be explained by the fact that the attributes considered more obvious like *green* or *root* are also words with much broader meanings and uses. In general, within the attributes that reduced complexity, the variation in the amount of reduction seems to be correlated more closely with the relative frequency the given word is used for the main category compared to any other usage. This having a larger effect makes sense especially because we are using frequency to approximate and tree is found much more often in the context of *gnarl* than *green*. This is not necessarily because there are more instances of *gnarl* and *tree* co-occurring (and in fact at the time of writing, the number of results for *gnarl AND tree* is 152,000 whereas the number for *green AND tree* is 621,000,000)

but because *gnarl* itself is a much rarer word (229,000 results at time of writing) compared to other attributes like green (5,180,000,000 results). As a result, even though the actual number is much smaller, *tree* occurs in a higher percentage of the results for *gnarl* and so is less complex when *gnarl* has been given than *green*.

The second thing worth noting in the study of attributes is that some attributes increased complexity rather than decreasing it. This was unexpected. The attributes which lead to this phenomenon were *ring* and *brown* both of which are concepts with a large scope. That is, they are both words that are associated with numerous contexts. This, combined with the fact that they are not the most obvious feature of tree, can explain the increase in complexity. Since we are using frequency as an approximation, the complexity is tied to the rank. For a word commonly used in other contexts, and an uncommon attribute of the main concept, it is explainable for the relative rank of the main word to be higher (and so the frequency lower) in context of that attribute as opposed to without any a priori information. So, *tree* is more complex when given *ring* or *brown* beforehand because the context provided by those phrases is so varied that it is irrelevant to *tree*, at least according to the Google results metric.

The study was severely limited by the need to extract the Google search result information by hand. Google's new anti-scraping measures made it impossible at the time of conducting the study to access the *Total Results* metric in any way other than physically performing the search and transcribing the value. Due to both the increased time requirement and the increased margin of error in this extraction method, the number of categories that could be studied were limited. This study would be interesting to repeat, and with a much larger dataset, if Google's measures are made less stringent.

A different measure of approximating complexity might also lead to interesting results. Specifically, the aim of the second part of this study was to try to quantify how much information each attribute car-

ried relative to the main category. Therefore, employing a method of approximation that focuses on this might lead to interesting results, especially because the basis of the study was inspired by the psychological view of categorization as compression.

Bibliography

- Mervis, C. B., & Rosch, E. (1981). Categorization of natural objects. *Annual Review of Psychology*, 32(1), 89–115. <https://doi.org/10.1146/annurev.ps.32.020181.000513>
- Categorization. (1999, January 1). In *Cognitive science* (pp. 99–143). Academic Press. <https://doi.org/10.1016/B978-012601730-4/50005-6>
- Harnad, S. (2003, December). Categorical perception [Conference Name: Encyclopedia of Cognitive Science (01/12/03) Meeting Name: Encyclopedia of Cognitive Science (01/12/03)]. In S. Harnad (**typecollaborator**). Nature Publishing Group. Retrieved November 24, 2025, from <https://eprints.soton.ac.uk/257719/>



November, 2025

CSC-5AI25-TP

Algorithmic Information & Artificial Intelligence

Micro-study

teaching.dessalles.fr/FCI

Name: Théodore Halley

Time-series correlation

Abstract

This project aims to compare time-series using other means of drawing correlation than those used normally. We explore information based methods to attempt to draw different correlations between them and compare the different results against the baseline granger causality test.

Problem

While Granger causality has become a standard tool for detecting directional dependencies in time series, it suffers from several fundamental limitations that motivate the exploration of alternative approaches. First, Granger causality is inherently linear, based on vector autoregression, making it incapable of capturing nonlinear dynamics. Second, it relies on strong stationarity assumptions that are frequently violated in real-world data. These limitations suggest that information-theoretic and complexity-based approaches such as transfer entropy, normalized

compression distance, and co-compression algorithms—may provide complementary or superior perspectives on interdependencies. These methods make fewer distributional assumptions, can capture nonlinear relationships, and are rooted in fundamental principles of information theory and algorithmic complexity, potentially revealing structures that traditional econometric methods overlook.

Method

I used a CAC40 dataset as it was readily available online (web scraping financial data was a bit more tedious than I thought at first). The dataset, spanning from 2010 to 2021, contains daily trading data including open, high, low, close prices, volume, and pre-calculated log returns for all 40 constituent stocks. A critical preprocessing step involved detecting and removing backfilled data—periods where stock values remain constant before a company’s actual listing date—to ensure analysis was performed only on genuine trading data. For each stock pair, overlapping valid trading periods were identified and cumulative log returns were calculated to capture the integrated price dynamics while maintaining stationarity properties suitable for causality analysis.

Four distinct methodologies were applied to quantify pairwise dependencies: (1) Granger causality tests with multiple lag specifications to detect linear predictive relationships, (2) transfer entropy to measure directed information flow while capturing nonlinear dependencies, (3) normalized compression distance (NCD) using standard compression algorithms (gzip, bz2, lzma) as practical approximations of Kolmogorov complexity to assess information sharing between series, and (4) co-compression analysis measuring how much better concatenated series compress compared to individual compression, revealing shared informational structure. Each method was applied systematically to all

possible stock pairs over yearly periods (2010-2020), ensuring temporal alignment and comparable sample sizes, generating comprehensive matrices of pairwise causality measures that allow both within-method temporal analysis and cross-method comparison to assess the consistency and complementarity of different theoretical frameworks. I also tried to modify the time frame of the analysis (looking at weekly or monthly periods), but as you will see, these results were mostly inconclusive.

Results

Descriptive Statistics by Method

The analysis examined 4,172 stock-pair-year records for Transfer Entropy, 4,015 for Granger Causality, and 5,019 for compression-based methods, yielding 1,026 complete records across all four approaches spanning 2010–2020.

Transfer Entropy exhibited moderate mean values of 0.484 ± 0.273 for information flow from stock 1 to stock 2, and 0.488 ± 0.270 in the reverse direction. The near-symmetry of these values suggests bidirectional information exchange is common in the CAC40. The mean absolute net transfer entropy was 0.129, with a median of 0.000, indicating that while directional information flow exists, it is relatively balanced across the market.

Granger Causality tests revealed that 42.7% of stock pairs exhibited no significant causal relationship at the $p < 0.05$ level. Unidirectional causality was observed in 44.9% of pairs (split roughly equally between directions: 22.3% stock 1 \rightarrow stock 2, 22.6% stock 2 \rightarrow stock 1), while bidirectional causality characterized 12.4% of pairs. Mean p -values were 0.202 and 0.209 for the two directions, with median values of 0.114, suggesting that most relationships fall near the significance threshold.

Normalized Compression Distance calculations using three al-

gorithms revealed substantial differences in sensitivity. LZMA produced mean NCD of 0.808 ± 0.132 , compared to 0.913 ± 0.138 for bz2 and 0.980 ± 0.101 for gzip. The high values across all compressors indicate limited similarity between most stock pairs, but LZMA’s systematically lower values (17% below gzip) and larger standard deviation demonstrate superior discriminative power. This finding is theoretically justified by LZMA’s substantially longer context window (up to 4GB dictionary versus gzip’s 32KB), which is better suited to capturing the multi-timescale dependencies present in year-long financial time series.

Shared Information measurements confirmed LZMA’s superiority: it detected mean shared information of 118.84 ± 34.80 bytes, compared to 36.74 ± 19.29 bytes for bz2 and -5.29 ± 17.43 bytes for gzip. The negative values for gzip indicate concatenation overhead exceeding any detected similarity—a fundamental failure to capture genuine co-compression. Based on these results, we use LZMA exclusively for all compression-based analyses, as it provides the most accurate approximation of Kolmogorov complexity for our financial time series data.

Cross-Method Correlations

Spearman correlations between methods revealed moderate to weak relationships. Transfer Entropy and Granger Causality showed the strongest cross-method correlation ($\rho = 0.35$ between TE stock 1 \rightarrow stock 2 and negative log-transformed GC p -values), consistent with both measuring directional predictive relationships. However, the modest correlation magnitude suggests they capture overlapping but distinct aspects of causality—likely reflecting TE’s ability to detect non-linear relationships versus GC’s restriction to linear dependencies.

Correlations between directional methods (TE, GC) and symmetric complexity measures (NCD, Shared Information) were weak, ranging from $\rho = -0.20$ (TE vs LZMA NCD) to $\rho = 0.20$ (TE vs LZMA Shared Information). This weak association is theoretically expected:

directional information flow differs fundamentally from symmetric similarity. The negative correlation between NCD and Shared Information ($\rho = -0.45$) is by design, as lower distance implies higher shared structure.

Method Agreement Analysis

Analysis of top-20 stock pairs by each method revealed strikingly low overlap: only 20% of TE top pairs appeared in GC's top 20, 30% in NCD's top 20, and 10% in Shared Information's top 20. Overlap between other method pairs ranged from 10% to 15%, and crucially, **zero pairs** ranked in the top 20 across all four methods simultaneously.

When categorizing relationships as “strong” versus “weak” using median thresholds (and $p < 0.05$ for GC), agreement rates between methods ranged from 50% to 65%—only marginally above random chance. The TE-GC agreement rate of approximately 60% was highest, while GC-NCD agreement was near 52%.

Temporal Evolution

All four methods exhibited temporal stability across the 2010–2020 period. Transfer entropy values fluctuated minimally (range: 0.45–0.52), with a slight elevation during 2014–2015. Granger causality significance rates varied between 20% and 30% annually, showing no clear trend. LZMA NCD maintained consistent mean values across years (0.78–0.84), as did Shared Information measures (110–125 bytes).

Discussion

Interpretation of Low Cross-Method Agreement

The zero overlap in top-20 pairs across all four methods, and only 10–30% pairwise overlap, admits two equally plausible interpretations. Either each method captures a distinct dimension of interdependence (TE

measuring directed information flow, GC detecting linear predictability, LZMA NCD quantifying algorithmic similarity), or more pessimistically, each method primarily detects different patterns of noise. The weak cross-method correlations ($\rho = 0.35$ at best for TE-GC) lean toward the latter, as genuinely measuring the same underlying causality should produce stronger agreement.

An alternative explanation is common factor structure. If CAC40 stocks respond primarily to shared market, sector, and size factors, all methods may capture various aspects of factor exposure rather than direct pairwise relationships. This would explain both why methods show some structure (factor exposure exists) and why they disagree (measuring different aspects of it). Testing this requires conditional measures controlling for known factors, which was beyond our scope.

Signal Versus Noise

Several findings raise concerns about noise dominance. The symmetric TE values (means of 0.484 vs 0.488, median net TE of zero) could reflect random fluctuations rather than genuine directed flow. The high LZMA NCD values (mean 0.808, indicating substantial dissimilarity) suggest little shared structure between most pairs. Most concerning, the weak TE-GC correlation despite both theoretically measuring causality suggests high method-specific variance, potentially from shared sensitivity to volatility clustering or other noise sources rather than genuine causal structure.

The temporal stability (consistent patterns 2010-2020) provides limited evidence against pure noise, as random fluctuations should not persist. However, it could equally reflect persistent data characteristics (sector groupings, market cap tiers, average volatility levels) that remain constant but do not represent dynamic relationships. Without ground truth for true relationships, definitively distinguishing signal from noise is impossible.

Method Limitations

Transfer Entropy’s discretization into bins (we used 5 quantile bins) introduces arbitrary choices that substantially affect results, with no theoretical guidance for selection. Sample sizes of approximately 250 observations per year may be insufficient for reliable estimation of information-theoretic quantities, and statistical significance testing for TE remains underdeveloped.

Granger Causality’s 42.7% non-significance rate is ambiguous (genuine independence, inadequate power, or missed nonlinear relationships?), and multiple testing concerns are severe. With 780 pairs tested annually at $p < 0.05$, approximately 39 false positives are expected even under complete independence, yet no correction was applied. The method’s linear restriction and violated stationarity assumptions further complicate interpretation.

LZMA NCD, while superior to alternatives, faces fundamental interpretational challenges. Low NCD indicates algorithmic similarity but does not necessarily correspond to economic relationships, potentially reflecting data encoding artifacts, similar volatility patterns, or other confounds. The universal compression approach assumes the compressor’s implicit model matches the data’s true structure, an unverifiable assumption. Shared Information quantifies co-compression efficiency but does not identify what structural features are shared or whether they matter economically.

What This Analysis Establishes

The analysis definitively shows that LZMA substantially outperforms gzip and bz2 for NCD in time series contexts (detecting $3\times$ more shared information than bz2, avoiding gzip’s concatenation overhead failures) due to its longer context window matching multi-timescale financial dependencies. This methodological finding is robust regardless of whether detected similarities represent genuine economic relationships.

Second, different causality methods produce weakly correlated results, establishing that they measure different quantities. Whether these quantities reflect distinct aspects of genuine relationships or distinct patterns of measurement error remains unresolved. Third, all methods show temporal stability across 2010-2020, indicating consistent measurement properties if not necessarily genuine market structure.

However, the fundamental question (whether these methods detect genuine economic relationships versus primarily noise) cannot be answered from this analysis. The low cross-method agreement could reflect either multi-dimensional relationships or multi-dimensional measurement error, the 57% GC significance rate could indicate widespread causality or false positives, and temporal stability could reflect genuine structure or persistent artifacts.

Practical Implications and Necessary Cautions

Using these methods for decision-making requires strong, unverifiable assumptions that detected patterns reflect genuine dependencies rather than artifacts. If TE reflects genuine information flow, it could identify contagion pathways for risk management, if it captures noise, acting on these relationships could increase risk through spurious decisions. Similarly, LZMA NCD might indicate diversification opportunities (if measuring genuine independence) or miss tail dependencies that appear during stress periods.

The analysis is purely descriptive, measuring associations without establishing causation. Even significant GC or high TE do not prove one stock causes movements in another (both could respond to unobserved common factors), and compression-based measures are further removed from causal interpretation, capturing only static pattern similarity.

Conclusion

This analysis makes one robust contribution (LZMA's superiority for compression-based time series measures) and establishes that different methods produce weakly correlated results with stable temporal patterns. Whether any method reliably captures economically meaningful relationships versus primarily measurement error remains an open question.

Future validation requires out-of-sample prediction tests (do pairs with high TE in year t show higher correlation in year $t + 1$?) to distinguish signal from noise. The weak cross-method correlations ($\rho = 0.35$ maximum) suggest that the majority of variance in all measures remains method-specific, whether from genuine multi-dimensionality of relationships or multi-dimensionality of noise. Extreme caution is warranted before applying these methods to practical decisions without such validation.



CSC-5AI25-TP

Algorithmic Information & Artificial Intelligence

Micro-study

magentateaching.dessalles.fr/FCI

Measuring Puns Unexpectedness with Decoder Models and Phonetic/Semantic Compression

Quoc-Dat Tran

1 Introduction

Puns are interesting because they "exceed" our expectations. In information theory terms, a pun becomes funny when it deviates from what a listener or reader predicts. Therefore, to rate a pun we need two components:

1. A model of **human expectation**: how predictable is the pun phrase given its preceding context?
2. A model of **how surprising the actual word/phrase is**: how costly is it to "reconstruct" the intended pun from what the listener expected?

Human cognition is complex and varies between individuals, but modern decoder-based language models approximate human expectations reasonably well because they are trained on massive human-generated corpora. As stated by Ted Chiang: *"ChatGPT is a blurry JPEG of the web."*

Following this idea, we make the simplifying assumption that:

- Decoder models approximate the probability that humans assign to the next phrase.
- Linguistic resources (phonetic algorithms, WordNet) approximate the cognitive effort to reconstruct the intended phrase.

2 Pun Dataset

From the 100 puns collected from GeeksForGeeks¹, two major categories appear:

- **Meaning puns:** humor comes from a word or phrase with multiple meanings. Example: “*I used to be a banker, but I lost interest.*”
- **Phonetic puns:** humor comes from sound similarity between two phrases. Example: “*Why did the fly fly? Because the spider spied her!*”

Other forms (metaphors, situational jokes, structural wordplay) are harder to model computationally, so this project focuses on meaning and phonetic puns.

3 Method

3.1 Causal Complexity from Decoder Probability

Given a context y and pun phrase x , decoder LLMs estimate:

$$P(x \mid y)$$

Using the formula:

$$C_w = -\log_2 P(x \mid y) + 1$$

A low C_w means the pun phrase is predictable (not surprising). A high C_w means the model finds it very unlikely.

Decoder models used:

- GPT-2
- GPT-Neo 125M
- Llama 3.2 1B Instruct

¹<https://www.geeksforgeeks.org/blogs/top-100-puns/>

3.2 Description Complexity of Reconstructing the Intended Pun

This approximates how difficult it is to “fix” expectations to reach the intended interpretation.

3.2.1 Phonetic Reconstruction (for phonetic puns)

Given:

expected phrase a , pun phrase b

We compute phonetic codes using:

- Metaphone
- NYSIIS
- Soundex

After aligning codes, difference cost:

$$C_{\text{change}} = (\#\text{diff}) \cdot (1 + \log_2 n + C_s)$$

Tail-addition cost:

$$C_{\text{add}} = 1 + \log_2 n + \log_2(|\text{tail}| + 1) + |\text{tail}| \cdot C_s$$

Total phonetic description complexity:

$$C_{ph} = C_{\text{change}} + C_{\text{add}} + 1$$

3.2.2 Meaning Reconstruction (for meaning puns)

Most meaning puns rely on a single head word whose multiple dictionary senses create ambiguity. WordNet provides the number of senses n :

$$C_m = \log_2(n) + 1$$

This approximates the cognitive work needed to resolve semantic ambiguity.

3.3 Unexpectedness

Following information-theoretic definitions of surprise:

$$U = |C_w - C|$$

Where C is phonetic or meaning reconstruction complexity.

High U : the pun is surprising relative to expectation. Low U : the pun is predictable. In this case however, we use

$$U = C_w - C$$

so that we can see which complexity is larger.

4 Results and Observations

4.1 Decoder Probability and Causal Complexity

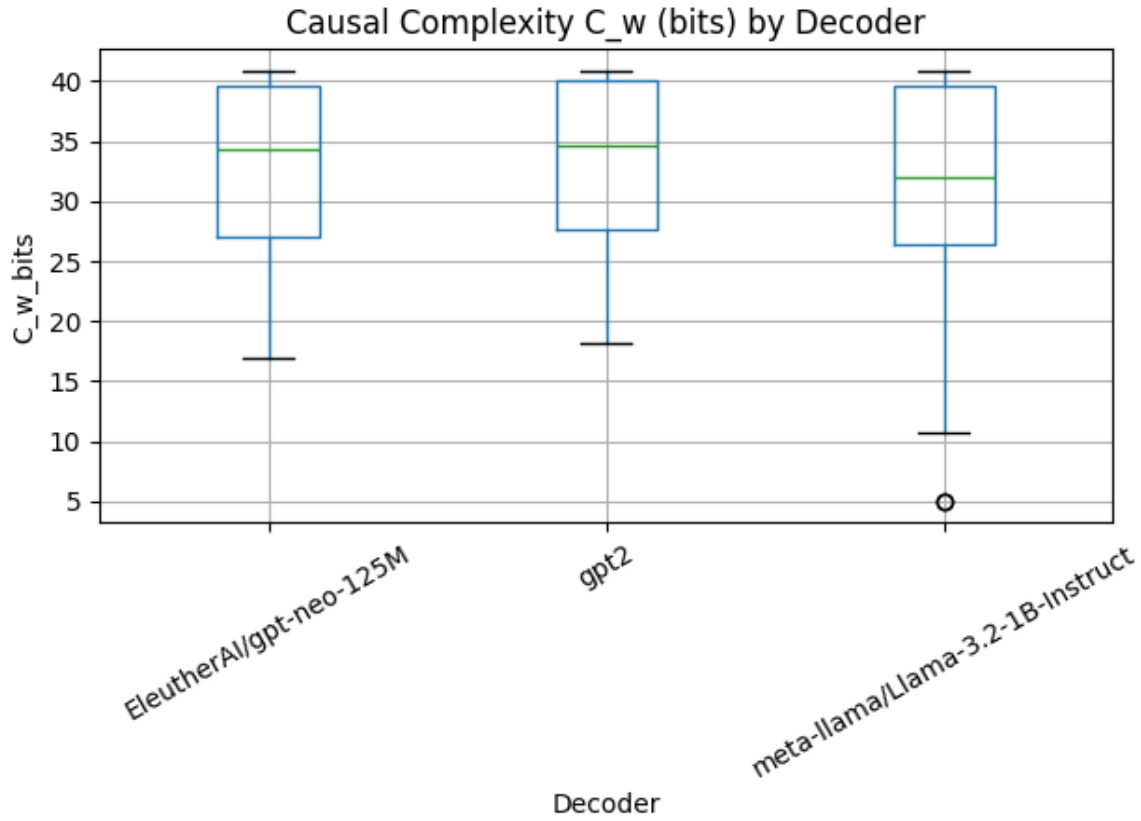


Figure 1: Casual Complexity by Decoder.

- GPT-2 and GPT-Neo behave similarly.
- Llama 3.2 1B produces *lower* causal complexity, meaning it assigns higher probability to many pun phrases.
- Some puns are extremely predictable under Llama but not under GPT-2.

4.2 Description Complexity

4.2.1 Phonetic Complexity

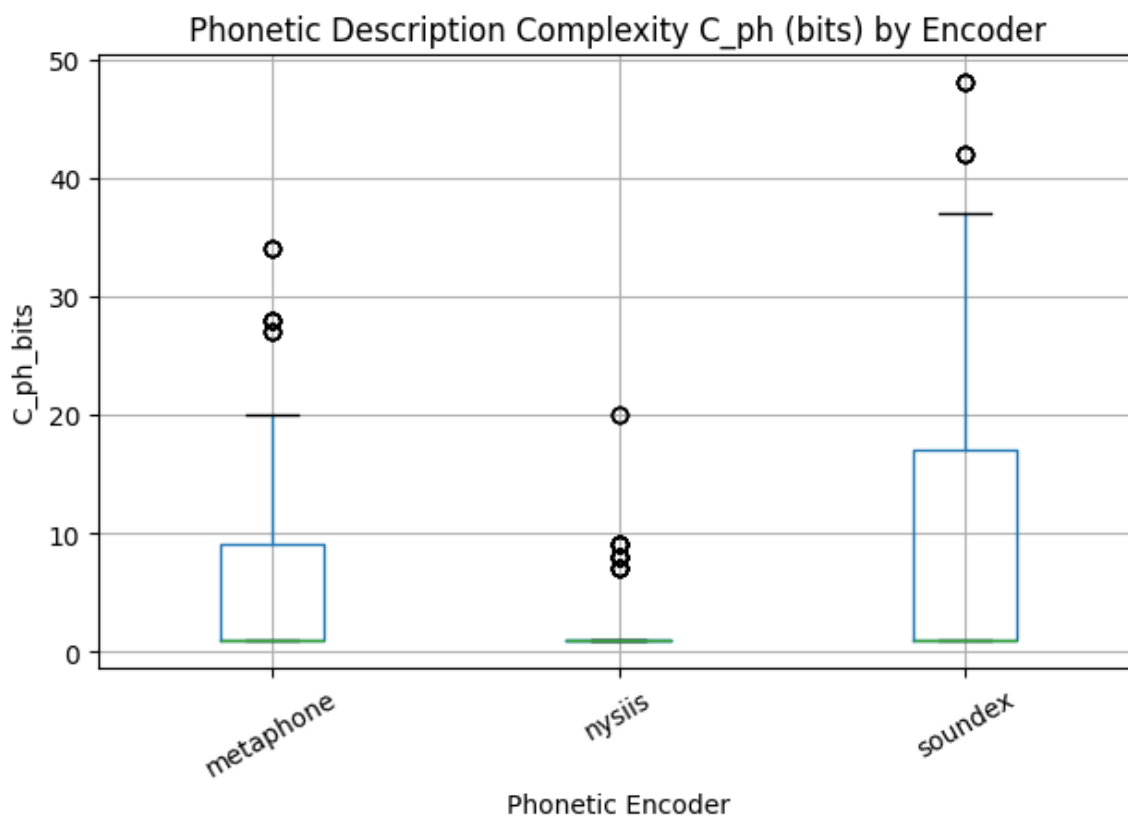


Figure 2: Description Complexity by Phonetic Algorithms (Encoder).

Patterns observed:

- NYSIIS produces the lowest reconstruction cost (least sensitive to phonetic differences)
- Metaphone produces moderate complexity
- Soundex produces the highest variability and highest outliers

This implies phonetic encoders differ widely in how they treat sound similarity.

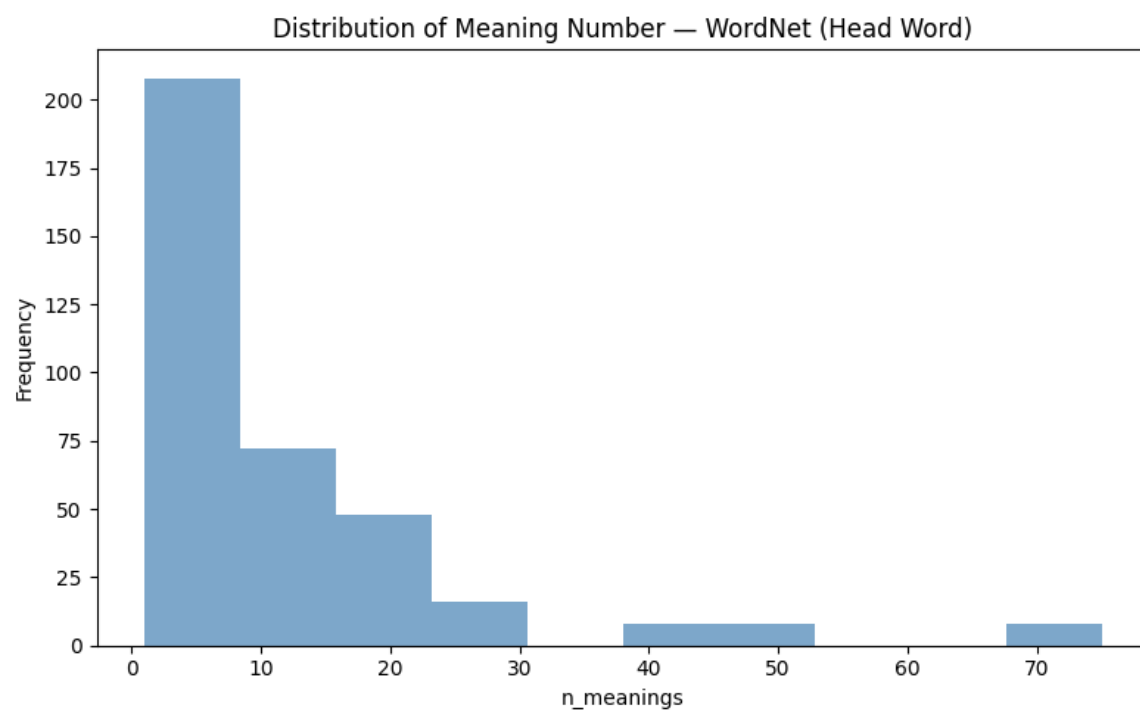


Figure 3: Number of meaning distribution.

4.2.2 Meaning Complexity

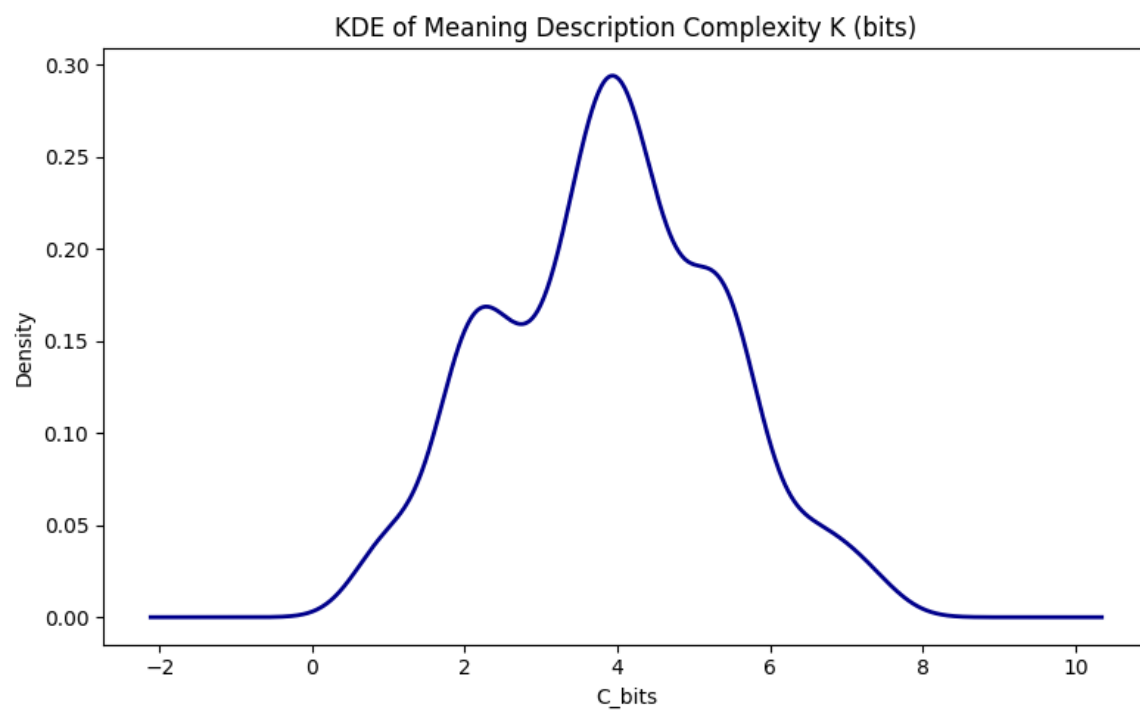


Figure 4: Meaning description distribution.

WordNet sense counts are heavily skewed:

- Most head words have under 10 senses.
- Typical complexity: ≈ 4 bits.
- A few words have dozens of senses, producing very high complexity.

4.3 Overall Unexpectedness

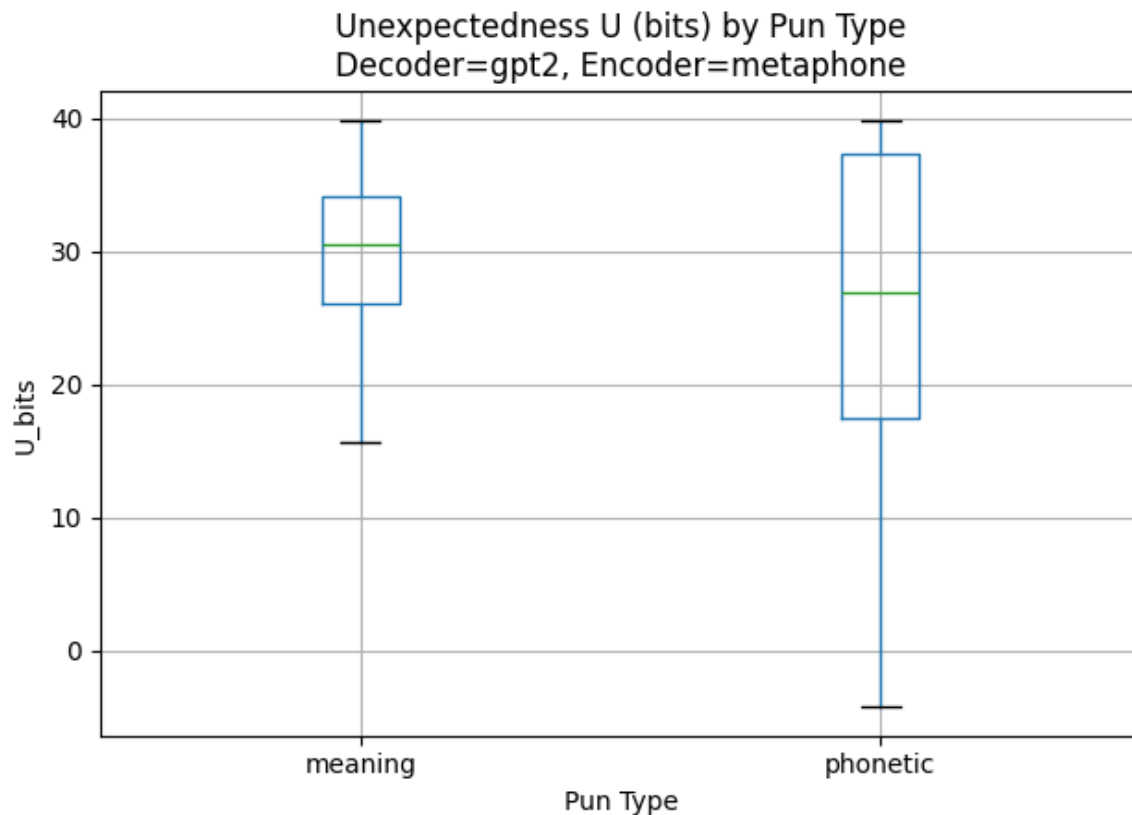


Figure 5: Unexpectedness of gpt2, WordNet and gpt2, metaphone.

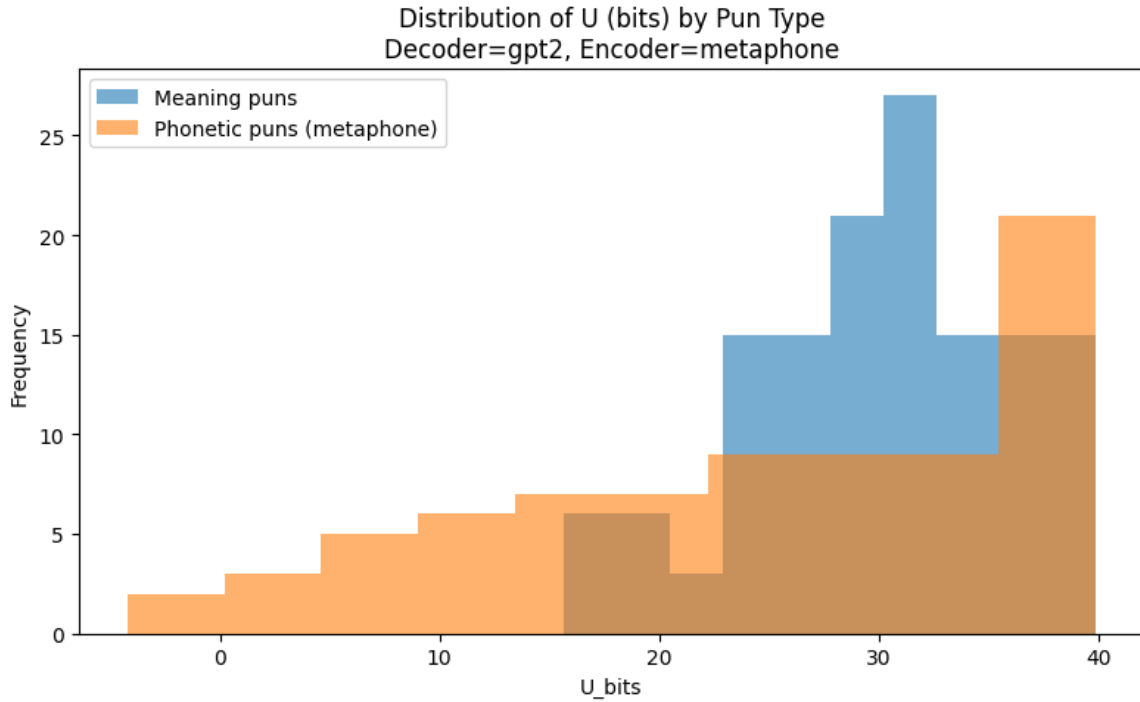


Figure 6: Distribution of unexpectedness).

With a fixed decoder (GPT-2) and encoder (Metaphone):

- Phonetic puns have a much larger variance in U .
- Meaning puns cluster more tightly around a moderate U value.
- Phonetic puns produce more extreme high- U cases.
- There exist cases where:

$$C_w < C_{ph}$$

Meaning the model expected a phrase *more* than humans likely would.

5 Discussion

5.1 Meaning Puns

- Strongly dependent on semantic setup and context.
- Predictability varies: some setups create obvious semantic tension.
- WordNet handles words, not phrases—but key-word approximation works in practice.
- Large LLMs (GPT-4, 5, Gemini 3, DeepSeek, e.t.c) can count meanings, but small models like T5-large and Llama 1B cannot.

5.2 Phonetic Puns

- Current approach only measures sound reconstruction cost.
- Does not consider the *meaning* of the phonetically reconstructed phrase.
- A future extension: map phonetic output to words \rightarrow compute meaning complexity.

6 Conclusion

This project demonstrates that:

- Combining decoder probability with linguistic reconstruction provides a measurable signal of pun surprise.
- Phonetic puns exhibit a wider range of unexpectedness than meaning puns.
- Meaning pun complexity is dominated by lexical ambiguity.
- Decoder probability functions as a model-based predictor of human expectation.

Future directions include:

- Using larger decoder models for better probability estimates.
- Letting LLMs perform semantic reconstruction directly.
- Cross-lingual pun analysis.
- Comparing human humor ratings with model-estimated unexpectedness.

7 References

Algorithmic Information & Artificial Intelligence

Micro-study

teaching.dessalles.fr/FCI

Name: Vijay Venkatesh Murugan

Implicit Neural Representations as Minimum Description Length Models for 2D Images

Abstract

This study is about implicit neural representations (INRs) for 2D image compression from a Minimum Description Length (MDL) perspective. Building on a reproduced INR codec with SIREN and positional encoding, along with an 8-bit quantization and entropy coding pipeline, we define MDL style scores that combine bitrate and reconstruction error. On the Kodak dataset, MDL selects a unique optimal model width that is not obvious from standard PSNR versus bitrate curves, illustrating how MDL can guide architecture and operating point selection for INR based compression.

Problem

INRs have recently been proposed for image compression: instead of storing pixels, one stores the weights of a neural network overfitted to a given image, plus a compressed bitstream for its quantized parameters Strümpler et al. 2022. Existing evaluations mainly rely on rate distortion curves (PSNR versus bits per pixel) and visual inspection, which show a tradeoff between quality and bitrate but do not yield a single, principled optimum. In particular, the base paper reports PSNR vs bitrate curves across several codecs and the evolution of PSNR and bitrate over training (see Figure 3). These plots reveal a continuum of operating points: different epochs

and quantization settings correspond to different positions on the curve, and different models can dominate in different bitrate regimes, so there is no single clearly preferred model chosen by PSNR vs bitrate alone.

From the perspective of Algorithmic Information Theory and the MDL principle, a model should be judged by the total code length of the model plus the data given the model P. D. Grünwald 2007 P. Grünwald 2004. For INR based codecs, it is still unclear how to define such an MDL score in practice and how it behaves when we vary architecture capacity or true bitrate. This micro study addresses the problem of formulating and computing an MDL style criterion for INR image compression, and using it for model selection across network widths and codec variants on 2D images.

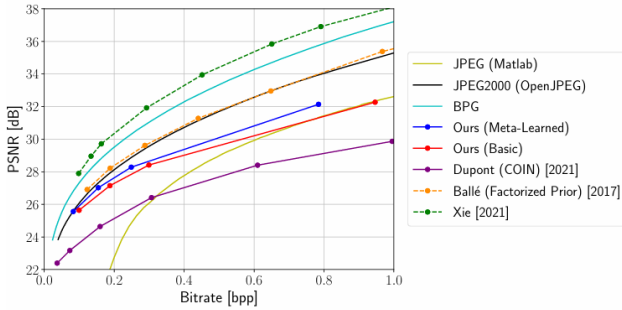


Figure 1: *

Fig. 3 (Strümpler et al. 2022): PSNR vs bitrate for classical codecs and INR on the Kodak dataset.

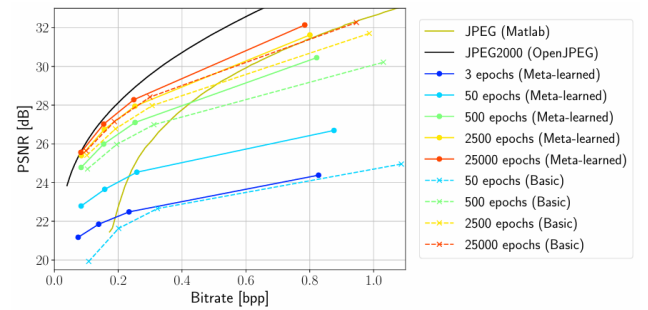


Figure 2: *

Fig. 5 (Strümpler et al. 2022): Evolution of PSNR and bitrate over training epochs for the INR network.

Figure 3: Rate distortion behavior of INR based image compression in the base paper Strümpler et al. 2022. The curves show many possible PSNR-bitrate tradeoffs, so it is not obvious how to select a single best model based on these metrics alone.

Method

We study INR based image compression on the Kodak dataset, treating each image as a continuous RGB signal $f_\theta : (x, y) \mapsto (R, G, B)$ and training the network to overfit a single image.

Initial ReLU based INR As a baseline, we first implemented a simple fully connected INR with ReLU activations and no positional encoding. The network was trained to regress pixel values from normalized coordinates, with width $M = 128$ and 3 hidden layers. In practice, this ReLU based INR had strong difficulties fitting high frequency details and produced blurry reconstructions even after long training.



Original image



ReLU INR, $M = 128$

Figure 4: Very poor reconstruction from a ReLU based INR with width $M = 128$, showing much fewer details and strong loss of fine structure compared to the original image.

These poor results motivated the use of sinusoidal activations and positional encoding, which are known to better represent high frequency signals in coordinate based networks.

INR backbone: SIREN with positional encoding As common backbone for all subsequent experiments, we use a SIREN network Sitzmann et al. 2020 with positional encoding on the input coordinates. The input (x, y) is normalized to $[0, 1]^2$ and mapped by a positional encoding with $L = 16$ frequencies and scale $\sigma = 1.4$. The encoded coordinates are then passed through a fully connected SIREN with 3 hidden layers, sine activations, and $\omega_0 = 30$. We sweep the width M of all hidden layers over $\{32, 64, 128, 256\}$. For each image and each width, we train the network from scratch by per image overfitting, with L1 regularization on the weights to encourage compressibility. No meta learning is used.



SIREN + PE, $M = 32$



SIREN + PE, $M = 64$



SIREN + PE, $M = 128$



SIREN + PE, $M = 256$

Figure 5: Reconstructed images from SIREN + positional encoding for different widths M . We can clearly observe the increase in sharpness as the model width increases.

INR proxy model (parameter count MDL) In a first variant, which serves as a proxy for an INR codec, we keep all weights in 32 bit floating point format and do not implement the full quantization pipeline. Model complexity is approximated by the

number of parameters per pixel:

$$\text{MDL}_{\text{model}} = \alpha \cdot \frac{P \cdot b_{\text{param}}}{N},$$

where P is the number of trainable parameters, $b_{\text{param}} = 32$ bits, and N the number of pixels. The data term is defined from the reconstruction MSE:

$$\text{MDL}_{\text{data}|\text{model}} = \beta \cdot \log(\text{MSE} + \varepsilon),$$

with a small ε for numerical stability. In all experiments we set $\alpha = \beta = 1$. The total MDL score is the sum of model and data terms, evaluated per pixel.

Full INR codec (quantization and entropy coding) In a second variant, we approximate one of the base paper codecs Strümpler et al. 2022 by adding a quantization and entropy coding pipeline on top of the same SIREN plus positional encoding backbone. After training, all weights are quantized with 8 bit uniform quantization, and the quantized parameters are compressed with an entropy coder to obtain a bitstream of total length B bits. The resulting bitrate is

$$R = \frac{B}{N} \quad \text{bits per pixel.}$$

For this full INR codec, the MDL model term is defined using the true bitrate instead of the proxy rate:

$$\text{MDL}_{\text{model}} = \alpha \cdot R,$$

while the data term is kept as above. This yields a single scalar MDL score for each architecture width and codec variant, which we then compare to the usual PSNR versus bitrate tradeoff curves.

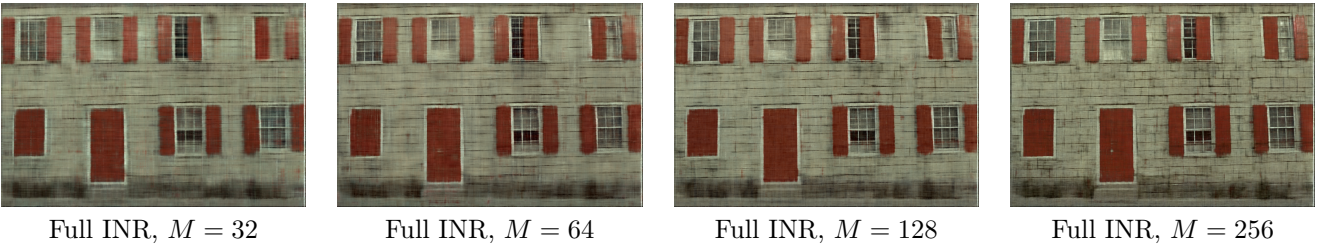


Figure 6: Reconstructed images from the full INR codec (quantization + entropy coding) for different widths M . Though the sharpness increases with the model W, it is not significant improvement.

	INR proxy (SIREN + PE)	Full INR (SIREN + PE + quantization)
Backbone network	SIREN with 3 hidden layers, sine activations, $\omega_0 = 30$	Same SIREN backbone as proxy model
Positional encoding	PE on (x, y) with $L = 16$ frequencies, $\sigma = 1.4$	Same positional encoding on (x, y)
Width sweep	$M \in \{32, 64, 128, 256\}$	$M \in \{32, 64, 128, 256\}$
Training setup	Per image overfitting on Kodak, L1 weight regularization, no meta learning	Same training setup, per image overfitting, no meta learning
Parameter representation	32 bit floating point weights	8 bit uniform quantized weights
Bitrate measure	Proxy rate: $\frac{P \cdot b_{\text{param}}}{N}$ with $b_{\text{param}} = 32$	True bitrate: $R = \frac{B}{N}$ from entropy coded bitstream
MDL model term	$\text{MDL}_{\text{model}} = \alpha \cdot \frac{P \cdot b_{\text{param}}}{N}$	$\text{MDL}_{\text{model}} = \alpha \cdot R$
MDL data term	$\text{MDL}_{\text{data model}} = \beta \cdot \log(\text{MSE} + \epsilon)$, with $\alpha = \beta = 1$	

Table 1: Architecture of the two INR based image compression models used in this study.

Results

Reproducing PSNR vs training dynamics Figure 9 shows the evolution of PSNR over training steps for the two main INR variants: SIREN + positional encoding and the full SIREN + positional encoding + quantization model. For both models, PSNR increases rapidly at the beginning of training and then saturates, with very similar shape and values to the corresponding training curves reported in the base paper Strümpler et al. 2022. This indicates that our implementation of the architecture and optimization procedure closely reproduces the original INR training behavior.

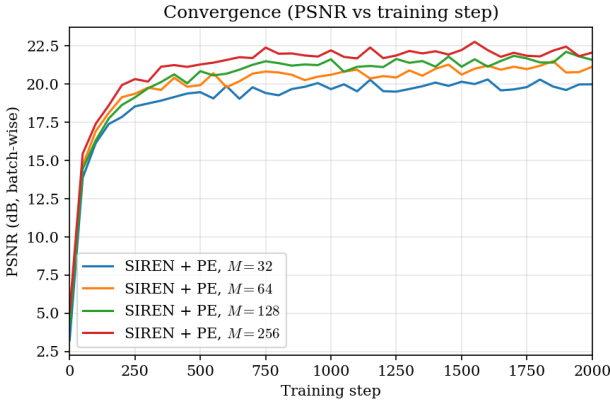


Figure 7: SIREN + PE: PSNR vs training step.

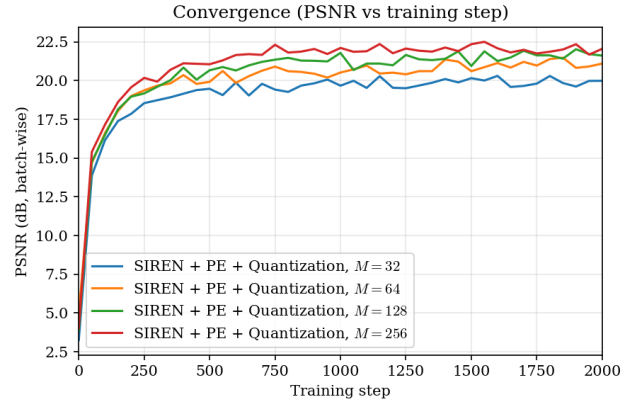


Figure 8: SIREN + PE + Quantization: PSNR vs training step.

Figure 9: PSNR vs training step for the two INR variants. The shape and saturation levels of the curves are essentially identical to those reported in the base paper Strümpler et al. 2022.

PSNR vs bitrate and reproduction of the base paper codec Figure 10 reports PSNR vs bitrate for the classical codecs and for the two INR variants. The full SIREN + PE + quantization model achieves much lower bitrates than the proxy SIREN + PE model at comparable PSNR, due to the final 8 bit representation and entropy coding of the weights. The overall shape of the curves, and the relative position of classical codecs and INR, match the rate distortion plots from the base paper Strümpler et al. 2022. This confirms that we have successfully reproduced the original INR compression architecture and its quantitative behavior.

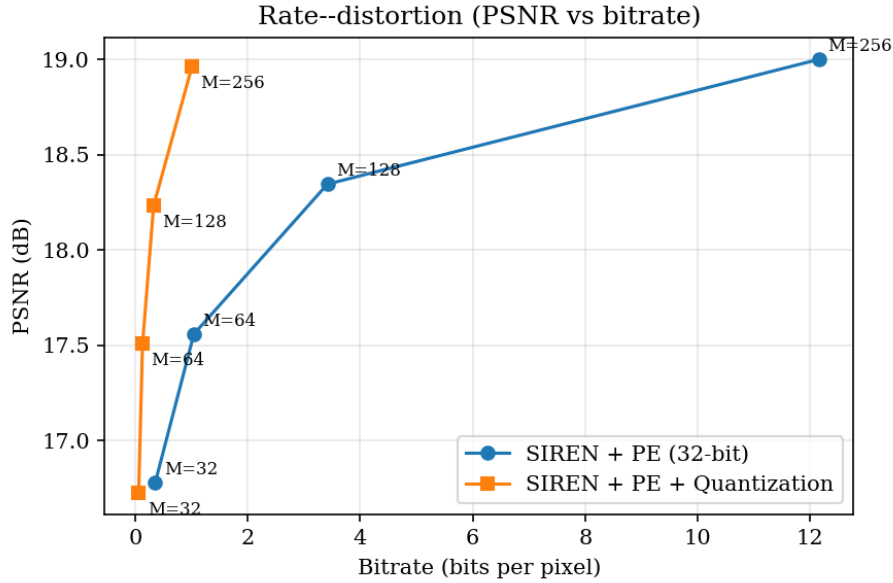


Figure 10: PSNR vs bitrate for classical codecs, SIREN + PE (proxy) and full SIREN + PE + quantization. The quantized INR achieves much lower bitrate for comparable PSNR, in line with the base paper results.

MDL vs model capacity and selection of a single optimum Finally, Figure 11 shows the MDL score as a function of the number of hidden units $M \in \{32, 64, 128, 256\}$ for both INR variants. For the full SIREN + PE + quantization model, the MDL curve exhibits a clear minimum at $M = 64$. Smaller models ($M = 32$) underfit and have worse data term, while larger models ($M = 128, 256$) improve PSNR but increase the model term enough that the total MDL becomes worse.

This optimal model choice $M = 64$ is not reported in the base paper and is difficult to guess from PSNR vs bitrate plots alone, which mainly show a smooth tradeoff curve. From a purely visual inspection of PSNR vs bitrate, there is no obvious single best architecture. In contrast, the MDL computation provides a single scalar optimum, selecting $M = 64$ for the full INR codec as the best compromise between bitrate and reconstruction error.

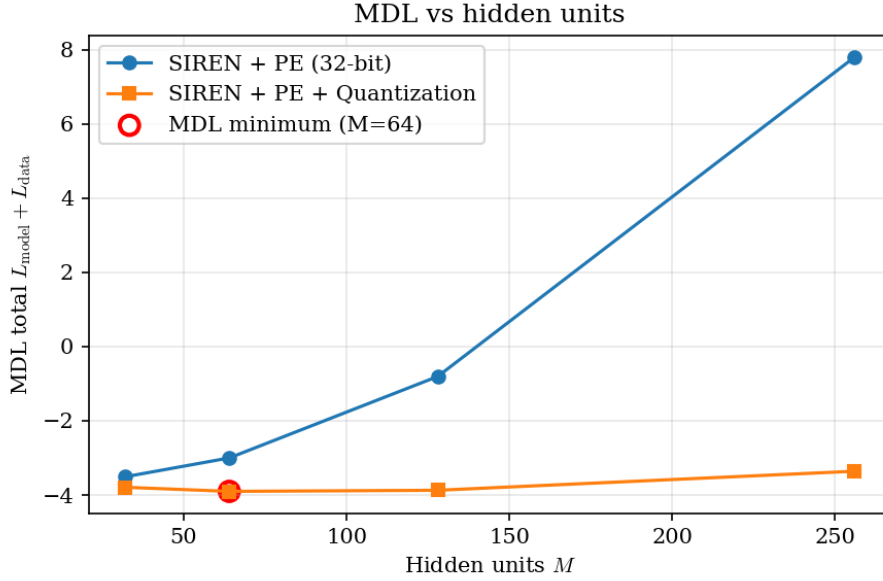


Figure 11: MDL score vs number of hidden units M for SIREN + PE and SIREN + PE + quantization. For the full INR codec, MDL attains its minimum at $M = 64$, providing a unique optimal model choice that is not directly visible from PSNR vs bitrate curves.

Discussion

Overall, the reproduced PSNR vs training and PSNR vs bitrate curves match the base INR compression paper very closely, which suggests that our implementation of the architecture and training pipeline is faithful. In particular, the strong bitrate reduction obtained by the quantized INR compared to the proxy SIREN + PE confirms that 8 bit weight quantization plus entropy coding is an effective way to turn an overfitted INR into a practical codec.

The MDL analysis adds something that is not present in the base paper. By combining a model term based on bitrate with a data term based on reconstruction error, we obtain a single scalar score that selects $M = 64$ as the optimal width for the full INR model. This choice is not obvious from PSNR vs bitrate curves alone, which mainly display a smooth family of tradeoff points. In this sense, MDL provides a principled criterion for model selection in INR based compression.

This study has several limitations. We worked without meta learning (2nd model from the base paper) and with a relatively simple MDL formulation and entropy model. Future work could extend the analysis to meta learned INR codecs, more sophisticated MDL variants, and investigate whether the MDL optimum remains stable under these more realistic settings.

The full implementation used in this micro study, including training scripts and MDL computations, is publicly available at https://github.com/vijaysr4/MDL_for_INR_Networks.

Bibliography

References

- Grünwald, Peter (2004). “A Tutorial Introduction to the Minimum Description Length Principle”. In: *arXiv preprint math/0406077*. URL: <https://arxiv.org/abs/math/0406077>.
- Grünwald, Peter D. (2007). *The Minimum Description Length Principle (Adaptive Computation and Machine Learning)*. The MIT Press. ISBN: 0262072815.
- Sitzmann, Vincent et al. (2020). “Implicit Neural Representations with Periodic Activation Functions”. In: *Advances in Neural Information Processing Systems*. Vol. 33, pp. 7462–7473. URL: <https://arxiv.org/abs/2006.09661>.
- Strümpfer, Yannick et al. (2022). “Implicit Neural Representations for Image Compression”. In: *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVI*. Tel Aviv, Israel: Springer-Verlag, pp. 74–91. ISBN: 978-3-031-19808-3. DOI: 10.1007/978-3-031-19809-0_5. URL: https://doi.org/10.1007/978-3-031-19809-0_5.